

**University of Central Florida
Department of Electrical & Computer Engineering**



**UNIVERSITY OF
CENTRAL FLORIDA**

Smart Parking System

EEL 4914 | Senior Design I | Spring 2022 | Group B

Senior Design 1 Documentation

Oscar Acuna
Computer Engineering
oacuna@knights.ucf.edu

Jordan Johnson
Electrical Engineering
jordan614407@knights.ucf.edu

M. Ridwan
Computer Engineering
mridwan@knights.ucf.edu

Kyle Carpenter
Computer Engineering
kylecarpenter@knights.ucf.edu

Table of Contents

| | |
|--|----|
| List of Figures | 7 |
| List of Tables | 9 |
| 1.0 Executive Summary | 10 |
| 2.0 Project Description | 11 |
| 2.1 Project Motivation | 11 |
| 2.2 Goals and Objectives | 11 |
| 2.3 Function of Project | 12 |
| 2.4 Project Block Diagrams | 12 |
| 3.0 Project Requirement Specifications, Constraints, and House of Quality | 14 |
| 3.1 Requirement Specifications | 14 |
| 3.1.1 General | 14 |
| 3.1.2 Hardware | 14 |
| 3.1.3 Web Application and User Interface Integration | 15 |
| 3.1.4 OpenCV | 15 |
| 3.1.5 Wifi Connectivity | 15 |
| 3.1.6 Power | 15 |
| 3.2 Constraints | 15 |
| 3.2.1 Standards | 16 |
| 3.3 House of Quality | 16 |
| 4.0 Existing Smart Parking Systems | 17 |
| 4.1 Existing implementation of Intelligent Parking Systems | 17 |
| 4.1.1 Indect | 18 |
| 4.1.2 ParkEagle | 19 |
| 4.1.3 CleverCiti | 20 |
| 4.1.4 Smart Parking Limited | 21 |
| 5.0 Technology Research | 23 |
| 5.1 Computer Vision | 23 |
| 5.1.1 OpenCV | 23 |
| 5.1.2 Computer Vision Techniques | 24 |
| 5.1.2.1 Edge Detection | 24 |
| 5.1.2.2 Hough Transform | 25 |
| 5.1.2.3 Regions of Interest | 26 |
| 5.1.2.4 Object Detection | 26 |
| 5.1.2.4.1 Cascade Classifiers | 26 |

| | |
|--|----|
| 5.1.2.4.2 You Only Look Once (YOLO) | 27 |
| 5.1.2.4.3 Object Detection Method Comparison | 28 |
| 5.2 Cameras | 29 |
| 5.2.1 OpenCV AI Kit (OAK) Cameras | 30 |
| 5.2.1.1 OAK-1 PoE | 30 |
| 5.2.1.2 OAK-D PoE | 30 |
| 5.2.1.3 OAK-D Pro PoE | 30 |
| 5.2.2 IP Cameras | 31 |
| 5.2.3 Camera Comparison | 31 |
| 5.2.4 Final Camera Selection | 32 |
| 5.2.4.1 Movidius Myriad X VPU | 33 |
| 5.2.4.2 DepthAI | 33 |
| 5.2.4.3 OAK-1 PoE Electrical Characteristics | 35 |
| 5.2.4.4 OAK-1 PoE Mechanical Information | 36 |
| 5.3 Microcontrollers | 36 |
| 5.3.1 Atmel (Microchip) | 38 |
| 5.3.1.1 ATSAM4E8CA-AN | 38 |
| 5.3.1.2 AT32UC3A1128-AUT | 38 |
| 5.3.2 Microchip | 38 |
| 5.3.2.1 ATSAME70J19A-AN | 38 |
| 5.3.2.2 PIC32MX664F064L-I/PF | 38 |
| 5.3.3 Infineon XMC4504F100F512ACXQMA1 | 39 |
| 5.3.4 Comparison Chart | 39 |
| 5.3.5 Microcontroller Final Selection | 39 |
| 5.4 LEDs | 40 |
| 5.4.1 LED Options | 40 |
| 5.4.2 LED Selection | 40 |
| 5.5 Local Server | 42 |
| 5.5.1 Odyssey X86J4125864 | 42 |
| 5.5.2 Raspberry Pi 4 | 43 |
| 5.5.3 UDOO X86 II | 44 |
| 5.5.4 Local Server Choice | 46 |
| 5.6 Ethernet PoE Switch and Local Network Internet Access | 46 |
| 5.6.1 PoE Switches | 46 |
| 5.6.1.1 PoE Switch Choice | 47 |
| 5.6.2 Local Network Internet Access | 47 |
| 5.6.2.1 Cellular Modem | 47 |
| 5.6.2.2 Smartphone's Hotspot and Tethering. | 48 |

| | |
|--|----|
| 5.6.2.3 Wireless Router | 48 |
| 5.6.2.4 Internet Access Choice | 49 |
| 5.7 Web Application Research | 49 |
| 5.7.1 Web Application Types | 49 |
| 5.7.1.1 Static Web Applications | 50 |
| 5.7.1.2 Dynamic Web Applications | 50 |
| 5.7.1.3 Single Page Web Applications | 50 |
| 5.7.1.4 Multi-Page Web Applications | 50 |
| 5.7.1.5 Animated Web Applications | 51 |
| 5.7.1.6 E-Commerce Web Applications | 51 |
| 5.7.1.7 Portal Web Applications | 52 |
| 5.7.1.8 Rich Internet Applications | 52 |
| 5.7.1.9 Progressive Web Applications | 52 |
| 5.7.1.10 Decided Web Application | 53 |
| 5.7.2 Web Development Stacks | 53 |
| 5.7.2.1 LAMP Stack | 53 |
| 5.7.2.2 MEAN Stack | 53 |
| 5.7.2.3 MERN Stack | 54 |
| 5.7.2.4 MEVN Stack | 54 |
| 5.7.2.5 Python - Django Stack | 55 |
| 5.7.2.6 Ruby on Rails Stack | 55 |
| 5.7.2.7 Decided Web Stack | 55 |
| 5.8 Mobile Application Research | 56 |
| 5.8.1 Mobile Application Types | 56 |
| 5.8.1.1 Native Apps | 57 |
| 5.8.1.2 Web-Based Apps | 57 |
| 5.8.1.3 Hybrid Apps | 57 |
| 5.8.1.4 Mobile App Choice | 57 |
| 5.8.2 Cross-Platform (Android/iOS) App Development Framework Options | 57 |
| 5.8.2.1 Ionic | 58 |
| 5.8.2.2 Flutter | 58 |
| 5.8.2.3 React Native | 58 |
| 5.8.2.4 Xamarin | 59 |
| 5.8.2.5 Cross-Platform App Development Framework Choice | 59 |
| 5.9 Web Server Research | 59 |
| 5.9.1 PaaS vs. IaaS | 60 |
| 5.9.2 Web Server and Database Hosting Providers | 60 |
| 5.9.2.1 Digital Ocean | 60 |

| | |
|--|-----------|
| 5.9.2.2 Heroku | 61 |
| 5.9.2.3 MongoDB Atlas | 61 |
| 5.9.2.4 Microsoft Azure App Services | 62 |
| 5.9.2.5 Google Cloud Platform | 62 |
| 5.9.2.6 Amazon Web Services (AWS) | 62 |
| 5.9.2.7 Web Server and Database Hosting Provider Summary | 63 |
| 5.9.2.8 Web Server Hosting Provider Selection | 64 |
| 6.0 Related Standards | 65 |
| 6.1 OSHA Standards | 65 |
| 6.2 Data Communication Standards | 66 |
| 6.2.1 Ethernet Standards | 66 |
| 6.3 Programming Standards | 69 |
| 6.3.1 Programming Naming Standards | 69 |
| 6.3.2 Programming Syntax Standards | 69 |
| 6.3.3 Programming Indentation and Bracing Standards | 70 |
| 6.4 Ingress Protection Code (IP Rating) | 70 |
| 6.5 Voltage and Power testing Standards | 71 |
| 6.6 NEMA Ratings for Enclosure Standards | 73 |
| 6.7 CPSC Standards | 74 |
| 6.8 Soldering Standards | 75 |
| 7.0 Design Constraints | 78 |
| 7.1 Economic Constraints | 78 |
| 7.2 Environmental Constraints | 78 |
| 7.3 Social Constraints | 79 |
| 7.4 Political Constraints | 79 |
| 7.5 Ethical Constraints | 79 |
| 7.6 Health and Safety Constraints | 80 |
| 7.7 Manufacturability Constraints | 80 |
| 7.8 Sustainability Constraints | 81 |
| 7.9 Time Constraints | 81 |
| 7.10 Testing and Presentation Constraints | 81 |
| 8.0 System Design | 83 |
| 8.1 Computer Vision System Design | 83 |
| 8.1.1 Computer Vision System Overview | 83 |
| 8.1.2 Software Tools | 83 |
| 8.1.3 Software Design | 84 |
| 8.1.4 Software Flowchart | 85 |

| | |
|--|-----|
| 8.1.5 Hardware Design | 86 |
| 8.2 LED Display System Design | 87 |
| 8.2.1 Display Images | 87 |
| 8.2.2 Hardware | 88 |
| 8.2.3 Software | 89 |
| 8.3 Mobile App Design | 91 |
| 8.3.1 Mobile App Block Diagram | 92 |
| 8.3.2 Mobile App User Interface Design | 92 |
| 8.4 Web App Design | 93 |
| 8.4.1 Web App Use Case Diagram | 93 |
| 8.4.2 Database Entity Relationship Diagram (ERD) | 94 |
| 8.4.3 Web App User Interface Design | 95 |
| 8.5 Control Unit Design | 98 |
| 8.5.1 Control Unit's Hardware | 98 |
| 8.5.2 Control Unit's Software | 99 |
| 8.6 PCB Components | 102 |
| 8.6.1 Ethernet Components | 102 |
| 8.6.1.1 RJ45 Ethernet Port | 102 |
| 8.6.1.2 Ethernet PHY | 103 |
| 8.6.1.3 External Clock Source | 104 |
| 8.6.2 MCU LED Interface | 105 |
| 8.6.3 Step-Down Voltage Converter Circuit | 105 |
| 8.6.4 IDC Connectors | 106 |
| 8.6.5 Push Button | 106 |
| 9.0 Prototyping | 107 |
| 9.1 PCB Schematic Capture | 107 |
| 9.2 Bill of Materials | 108 |
| 10.0 Testing | 111 |
| 10.1 Hardware Testing | 111 |
| 10.1.1 Computer Vision System Hardware Testing | 111 |
| 10.1.2 Microcontroller Hardware Testing | 112 |
| 10.1.3 PCB Hardware Testing | 113 |
| 10.1.3.1 Ethernet Port and Cable Testing | 114 |
| 10.1.3.2 Ethernet PHY Testing | 114 |
| 10.1.3.3 Step-Down Voltage Converter Circuit Testing | 114 |
| 10.2 Software Testing | 115 |
| 10.2.1 Computer Vision System Software Testing | 115 |

| | |
|--|-----|
| 10.2.2 Local Server Software Testing | 116 |
| 10.2.3 Microcontroller Software Testing | 117 |
| 10.2.4 Web App Testing | 118 |
| 10.2.5 Mobile App Testing | 120 |
| 11.0 Mounting and Installation Procedure | 122 |
| 12.0 Project Budgeting and Financing | 125 |
| 13.0 Project Milestones for Each Semester | 126 |
| 13.1 Semester 1 (Senior Design 1) | 126 |
| 13.2 Semester 2 (Senior Design 2) | 127 |
| 14.0 Project Management | 128 |
| 15.0 Conclusion | 129 |
| References | 130 |

List of Figures

| | |
|---|-----|
| Figure 1: LED Display - Software Overview..... | 12 |
| Figure 2: Hardware Block Diagram Overview..... | 13 |
| Figure 3: Web and Mobile App - Software Overview..... | 13 |
| Figure 4: Parking Space Detection Diagram - Software Overview..... | 14 |
| Figure 5: Example of Canny Edge Detector..... | 25 |
| Figure 6: Hough Transform on a Parking Lot..... | 26 |
| Figure 7: Cascade Classifier for Face Detection using Eyes as Feature..... | 27 |
| Figure 8: Example of YOLO Object Detection | 28 |
| Figure 9: Functional Block of Belago 1.1..... | 31 |
| Figure 10: OAK-1 PoE | 33 |
| Figure 11: High-Level DepthAI Software Architecture..... | 34 |
| Figure 12: OAK-1 PoE Mechanical Measurements..... | 36 |
| Figure 13: RGB LED Matrix Panel - 32x64 | 41 |
| Figure 14: PoE Connection Standards | 68 |
| Figure 15: Acceptable vs Rework Needed Soldering Process | 76 |
| Figure 16: Acceptable vs Not Acceptable Part Mount | 77 |
| Figure 17: Flowchart of the Computer Vision System Workflow | 85 |
| Figure 18: Basic Block Diagram of the OAK-1 PoE Hardware..... | 86 |
| Figure 19: LED Example Images (Double Digits) | 87 |
| Figure 20: LED Example Images (Single Digit) | 88 |
| Figure 21: LED Display IDC Connection | 89 |
| Figure 22: Corner Alley Example | 90 |
| Figure 23: LED Display Program Flow | 91 |
| Figure 24: Mobile App Block Diagram | 92 |
| Figure 25: Mobile App GUI Prototype..... | 93 |
| Figure 26: Web App User Case Diagram | 94 |
| Figure 27: Database Entity Relationship Diagram (ERD) | 95 |
| Figure 28: Front Page Design | 96 |
| Figure 29: Video Fee Page Design | 96 |
| Figure 30: User Administration Page Design | 97 |
| Figure 31: Parking Administration Page Design | 97 |
| Figure 32: Parking System Control Unit | 98 |
| Figure 33: Local Server Java Program GUI Design | 100 |
| Figure 34: Camera Text File Data Format | 100 |
| Figure 35: Local Server MySQL Database Design | 101 |
| Figure 36: Simple Schematic of Interface between Ethernet Cable and MCU | 102 |
| Figure 37: Simple Schematic of LEDs on RJ45 Ethernet Port | 103 |
| Figure 38: Simplified Schematic for the TPS563201 | 105 |
| Figure 39.1: PCB Schematics | 108 |
| Figure 39.2: PCB Schematics | 109 |
| Figure 40: Luxonis documentation example result of depthai_demo.py running | 110 |
| Figure 41: Point of View from Camera..... | 122 |
| Figure 42: Secondary Point of View from Camera | 122 |

List of Tables

| | |
|---|-----|
| Table 1: House of Quality | 16 |
| Table 2: Engineering Trade Off Matrix..... | 17 |
| Table 3: Object Detection Method Comparison | 29 |
| Table 4: Camera Comparison..... | 32 |
| Table 5: DepthAI SDK Classes and Functions | 35 |
| Table 6: Absolute Maximum Ratings of the OAK-1 PoE..... | 35 |
| Table 7: Recommended Operating Conditions of the OAK-1 PoE..... | 36 |
| Table 8: Microcontroller Comparison..... | 39 |
| Table 9: Specifications for Odyssey Mini PC..... | 42 |
| Table 10: Raspberry Pi 4 Specifications..... | 43 |
| Table 11: Specifications for UDOO X86 II..... | 44 |
| Table 12: Summary of PoE Switches..... | 46 |
| Table 13: Summary of Internet Solutions..... | 48 |
| Table 14: Web Server and Database hosting plan and Pricing..... | 63 |
| Table 15: Data rate based on IEEE 802.11..... | 66 |
| Table 16: Ethernet Wiring Standards..... | 67 |
| Table 17: IP Code First Digit Meaning | 70 |
| Table 18: IP Code Second Digit Meaning | 71 |
| Table 19: Software Development Tools | 83 |
| Table 20: Ethernet PHY Port Interface/MCU MAC Interface Pins | 104 |
| Table 21: Ethernet PHY Clock Interface Pins | 105 |
| Table 22: MCU LED Interface Pins | 106 |
| Table 23: PCB Bill of Materials | 109 |
| Table 24: Budget Breakdown | 125 |
| Table 25: Senior Design 1 Milestones | 126 |
| Table 26: Senior Design 2 Milestone | 127 |

1.0 Executive Summary

With the large number of students attending the University of Central Florida, parking congestion is a recurring issue that has caused a variety of problems for students and faculty. The current methods of dealing with this problem being, an LED sign outside of each garage indicating “open” or “full” and a website that displays the percentage of open parking spots in each garage are not reliable, and they are simply not enough to handle the volume of people that the parking garages at UCF endure. In this document, we suggest a different way of handling this problem by introducing a smart parking system that quickly communicates to students where open parking spots are within a garage through the use of small LED signs.

This project takes a unique approach compared to most available parking management systems, as we will not be using proximity sensors to monitor each individual parking spot. The combination of needing a large number of sensors to monitor every parking spot, power supply requirements for each sensor, complexities relating to installation, and maintenance requirements for all of the sensors makes for a solution that could become quite costly, especially when looking at large scale applications such as a UCF parking garage. Instead, we will be using cameras and computer vision in order to detect where multiple parking spots are and then make the decision of whether a parking spot is open or occupied. This information will be relayed to an LED sign situated at the end of a row of parking spots which will then indicate the exact number of open spots within that row.

While our group would like the opportunity to design a parking management system for an entire garage at UCF, we do have time and budget constraints that we need to adhere to. Therefore, we will be designing a system that is just a proof of concept that could be scaled to fit a garage of any size. This is an essential factor of our project as the entire motivation for doing this project comes from UCF’s parking garage issues. With this said, we will market this project as an efficient, accurate, and cost-effective solution for dealing with congested parking garages.

The physical deliverables for our project will include a camera, an ethernet switch, a local server, a PCB, and an LED sign. The camera we are using is a PoE device embedded with OpenCV that will feed live data on parking availability into the ethernet switch. From the ethernet switch, both data and power will be fed into our PCB through PoE, and the results will be displayed on a LED sign which will be driven by a microcontroller. The data coming into the switch by the camera will also be fed into a local server which will then send the live video feed and data on parking availability into the cloud.

In addition to these physical deliverables, we will also have a website and a mobile application which will receive the data from the local server. These applications will provide information on parking, including the number of available spots and where they are located, data analytics on the best times to find parking, and how long certain spots have been occupied for.

2.0 Project Description

2.1 Project Motivation

While the number of students that attend UCF's campus increases every year, the number of parking garages around campus does not. With a growing student body, naturally, there will be an increase in the number of vehicles flowing in and out of campus throughout the day. During times of the day when there is a lot of overlap between class sessions that are either starting or concluding, a large number of students and faculty are entering and exiting the garages around campus, which creates a bottleneck problem. This causes the garages to become quite congested as people navigate them. This congestion inside of the garages leads to a number of issues, including long lines inside of and outside of parking garages, backed up traffic around the perimeter of the campus, and late arrivals to class due to difficulties finding a parking spot.

Our group has decided that UCF's current solution to these issues is ineffective. A website that indicates the percentage of open parking spots inside each garage, along with signs outside of the garages indicating whether it is open or full, is simply not enough to ease the large flow of traffic that UCF's parking garages endure. Therefore, we will develop a smart parking system that aims to mitigate the problems described above.

2.2 Goals and Objectives

UCF is a growing school with more than 70406 students currently enrolled. With such a large number of students in attendance in addition to faculty, this leads to the parking garage issues that students and the faculty face every year.

With our project, the goal is to aid the UCF population by having a budget-friendly smart parking system that reduces the time it takes to get in the garage, the time it takes to get out of the garage, and the time it takes to find a parking spot. The objectives we have to help reach this goal are listed below:

- Use video cameras embedded with computer vision concepts via open-cv to detect open and occupied parking spots.
- Develop a website and mobile application where UCF students and faculty will be able to get a detailed description of the available parking spot in a timely manner.
- Use LED signs to communicate open parking spot locations for people navigating the garage.
- Develop our system in a way such that power consumption is low.
- Develop an ethernet network for transmitting data between the different components in our project.

With a successful implementation of our smart parking system, we hope to be able to conquer the problem of crowded parking garages with great results. Our smart parking system would allow UCF's students and faculty to be able to visualize and access parking data in a more efficient manner and not have to waste their valuable time and energy

focusing on parking issues anymore. At the conclusion of our project, we aim to be successful in building a realistic, budget-friendly smart parking system that uses low power consumption and is embedded with IoT.

2.3 Function of Project

The smart parking system should be capable of recognizing all vehicles entering and exiting the parking garages on campus and making real-time analysis of available parking spots in order to direct vehicles to new open locations. Through the utilization of a camera system and computer vision, the vehicles can be tracked through portions of the garage. A notification system built into the garage that is composed of LED signs that indicate open parking spots and how many are available on a specific section or level will be the primary guidance for the drivers.

Depending on whether stretch goals are achieved, the LED could vary in complexity. One of the target capabilities is showing available parking spot counts at each level of the garage, with a final stretch goal of having a complex LED system that guides drivers to individual parking spots. The LEDs would point to specific parking spots and make for a very unambiguous directing system.

In addition to physical signage in the garage, there will also be a mobile app component for the parking status updates. The UCF parking app will be overhauled to feature higher accuracy of how many spots are available as well as additional details, like the rate of vehicles entering the garage and which levels of the garage are full.

2.4 Project Block Diagrams

The preliminary project system is summarized in the block diagrams shown in Figures 1 through 4.

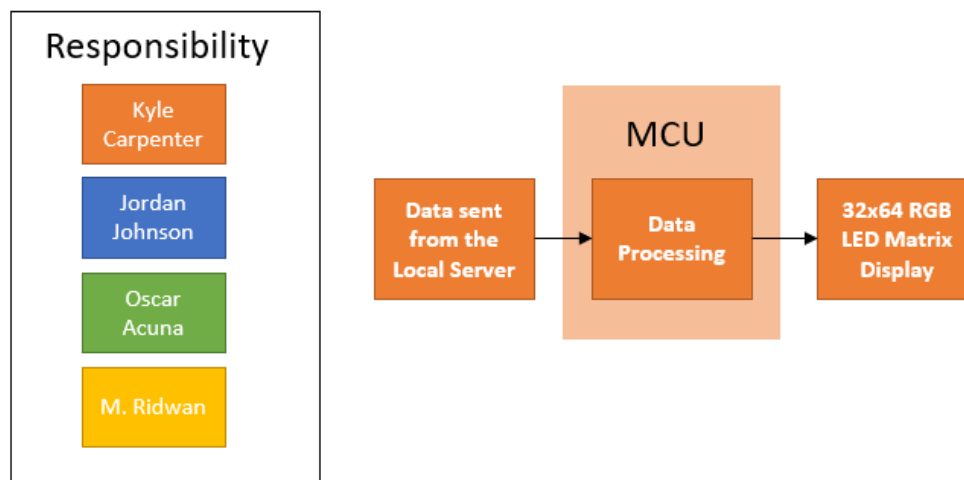


Figure 1: LED Display - Software Overview

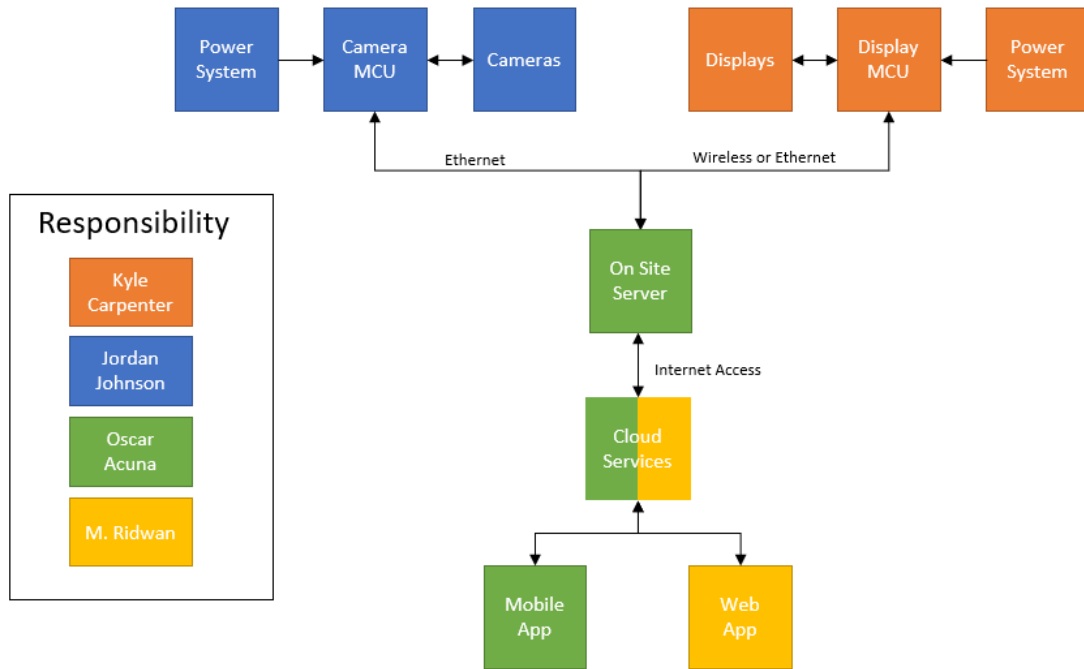


Figure 2: Hardware Block Diagram Overview

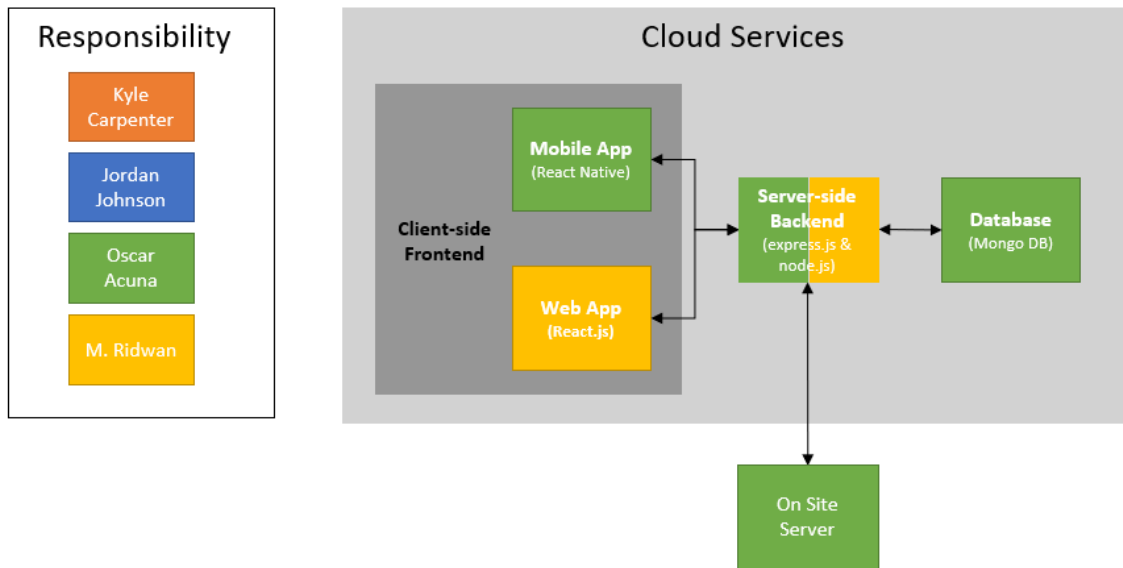


Figure 3: Web and Mobile App - Software Overview

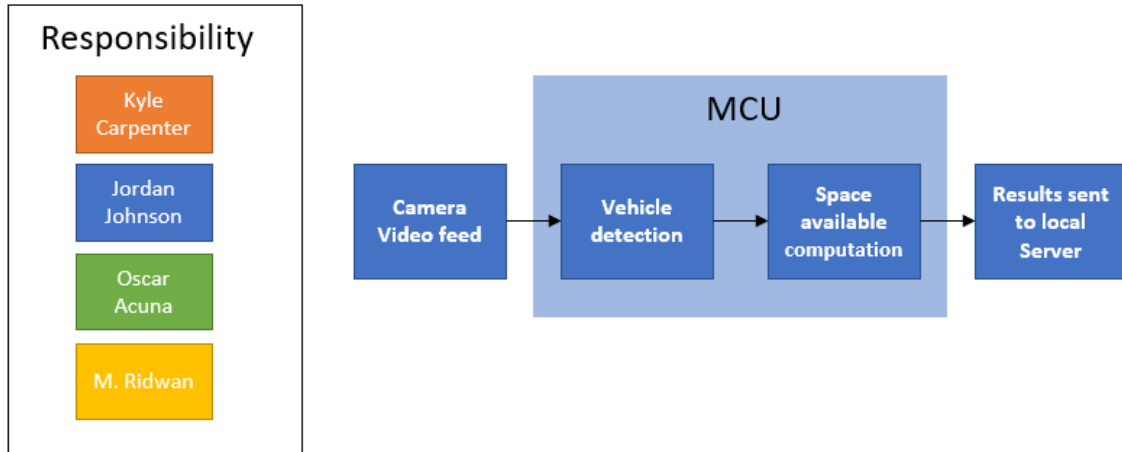


Figure 4: Parking Space Detection Diagram - Software Overview

3.0 Project Requirement Specifications, Constraints, and House of Quality

3.1 Requirement Specifications

3.1.1 General

- Our system should be able to monitor at least 15 parking spots.
- The system must detect available spaces using OpenCV in less than 2 seconds.
- The system must inform drivers of the number of available parking spots using an LED sign
- The system should be hoisted a minimum of 14 feet above the ground for more accurate video capture.
- Our system should be 100% accurate within 60 seconds of a change in # of open spots
- Our system should be 90% accurate within 30 seconds of a change in # of open spots

3.1.2 Hardware

- LED signs must have at least a 5,000 nit rating.
- LED signs must be at least 24 inches by 32 inches.
- A Microcontroller should have Wi-Fi and a communication module to drive the LED signs and transfer data to it.
- Cameras must be OpenCV compatible.
- Cameras must be able to see the required number of parking spots at the height of 14 feet.

3.1.3 Web Application and User Interface Integration

- Easily maneuverable Graphics interface to help end-user navigate and connect with the implemented video transmission technology.
- Stream real-time video footage of parking spots for better navigation aid.
- Display the total number of vacant parking spots within the designated area.
- Allow end-users to be able to reserve parking spots.

3.1.4 OpenCV

- OpenCV must detect when a parking spot is free or occupied.
- Train OpenCV over time to differentiate between the vacant and non-vacant parking spots successfully.
- GIS must be implemented to monitor the parking spot outlines and keep a count of the total parking spots.

3.1.5 Wifi Connectivity

- Cameras, PCB, and local server will all need to connect to a local network via Wi-Fi with a speed of at least 5 Mbps using the 2.4GHz frequency.
- A Wi-Fi router/access point will provide a local network for the cameras, microcontrollers, custom PCBs, and LED signs to connect to the local server. It should provide at least 10 Mbps upload/download speed in the 2.4GHz frequency.
- An ethernet switch with PoE-enabled ports or PoE injectors will be implemented to provide power to the cameras.

3.1.6 Power

- Cameras must be able to be powered by ethernet cable (Power over Ethernet)
- PCB must be battery powered

3.2 Constraints

With less than a year to deliver this project, time is the primary constraint that our group must adhere to. To ensure we are making considerable progress in a reasonable amount of time, we have created a list of project milestones that we will follow over the next year. Additionally, since our project will consist of some expensive components, we must ensure that our budget does not get out of control. Our group has agreed that we will spend no more than \$2,000, and an initial budget is shown in section 4.0. Additional constraints that we must adhere to in order for our system to work correctly are listed below.

- The system must be able to work in the daytime as well as nighttime. For nighttime, the camera must be able to work with at least 0.001 lux.
- The local Server must be able to be accessed remotely. If Windows OS or a Linux OS is used, software such as TeamViewer allows controlling the server remotely.
- The site must provide Internet access of at least 5Mbps.

3.2.1 Standards

- All electronics must be enclosed in cases designed following the IP65 standard to protect the equipment from dust and water.
- Cameras will be powered using Power over Ethernet standard 802.3af.
- Wi-Fi network must be standard 802.11ax to provide 2.4GHz network connectivity.
- The site must provide an energy source of 120V/60Hz
- Network communication of cameras and LED signs should use TCP/IP standards.

3.3 House of Quality

The house of quality, shown in Table 1, shows the positive and negative correlations between engineering and marketing requirements. In addition, the table helped in determining the engineering-marketing trade-off. In Table 2, the engineering trade-offs are shown, which clearly indicates how one category affects others.

Table 1: House of Quality

| | | | Engineering Requirements | | | | |
|--|------------------------|---|--------------------------|------------|-------------------------|----------------------------|----------|
| | | | Power | Waterproof | Day/Night Functionality | Accuracy | Cost |
| | | | - | + | + | + | - |
| M | Speed | + | ↑ | | | ↓ | ↓ |
| | Outdoor Functionality | + | ↓ | ↑↑ | ↑↑ | ↑ | ↓↓ |
| | Visual Indicators | + | | | ↑ | | ↓ |
| | Area Coverage | + | ↓ | | ↓ | ↑ | ↓↓ |
| | Mobile/Web Application | + | | | | ↑ | ↓ |
| | Cost | - | ↑ | ↓ | ↓↓ | ↓ | ↑↑ |
| Target Engineering Requirements | | | | ≥ IP65 | ≤ 0.001 lux | 90% in 30s, 100% in 60s | ≤ \$2000 |

Table 2: Engineering Trade Off Matrix

| | | Power | Waterproof | Day/Night Functionality | Accuracy | Cost |
|----------------------------|---|-------|------------|----------------------------|----------|------|
| | | - | + | + | + | - |
| Power | - | | | ↓ | ↓ | ↑ |
| Waterproof | + | | | | | ↓ |
| Day/Night Functionality | + | | | | | ↓ |
| Accuracy | + | | | | | ↓ |
| Cost | - | | | | | |

4.0 Existing Smart Parking Systems

Over the years, many companies have implemented different solutions to solve the global parking issue. Each of the implementations has its own innovative and optimized solutions to aid the population and introduce the intelligent parking concept in day-to-day lives. These solutions have steered the project's motivation to improve some of the existing solutions to improve parking problems. This section will cover the different systems already in place to solve the growing issue of parking congestion.

4.1 Existing implementation of Intelligent Parking Systems

The project we have chosen to pursue gives us quite a bit of freedom with the type of design we can develop with the fact that there are many ways to go about making parking garages more efficient. We have decided that we want to use cameras and computer vision techniques in our design, though it is common for existing smart parking systems to use video cameras, various sensors, or a combination of both. With that said, there are some existing smart parking systems that have served as motivation for our project. Brainstorming ideas gathered from existing solutions and improving upon those ideas and implementations is a great way of solving global problems. The majority of the big companies bounce ideas off of each other and create better solutions to support the previous implementations. Following the same strategy, for this Smart Parking project, a couple of other companies and their innovations were utilized. Some of their system providers have already created ample opportunities for growth within their systems. Like Indect's parking system allows camera sensors, state of the art casing that would withstand any natural calamities. Cleverciti has taken customer experience to another step with a subscription-based system. Companies like Parkeagle helped the parking garages and corporate building

parking solutions and took the initiative to handle large implementations of urban city parking issues. Then companies like Smart Parking Limited, with their globally scaled solutions, have incorporated the usage of cloud-based platforms along with customizable applications to cater to the audiences. The immense range of innovative solutions has paved the way for further improvement of parking dilemmas and the implementation of smart city concepts with technological advancements.

4.1.1 Indect

One of the biggest companies in the United States, Disney, have implemented the concept of Smart Parking in one of their newer addition of theme parks. Disney Springs at the heart of Orlando has become one of the most prominent tourist spots in Central Florida. So to tackle the parking problem for the massive number of people visiting the park every day, Disney Springs has adopted the new smart parking system provided by Indect. Indect achieves the solution to the parking issue by monitoring every one of the parking spots in Disney Springs parking garage and indicating their status, whether empty or not. This signal is shown on top of each parking spot with a Led light flashing. The LED light flashes green if the parking spot is vacant, and in the case of the spot being occupied, the LED light flashes a Red light. In addition, Indect provides endless level-counting and wayfinding solutions for Disney Springs by integrating LED signs at different locations (entrance, exit, turns) inside the garage for convenience. It also allows license plate recognition and locating the whereabouts of a car as a feature if someone mistakenly forgets the location of the parked vehicle.

Indect's Single Space Ultrasonic sensors provide the ability to integrate RGB color (red, green, blue) LEDs through parking spaces to be clearly visible to consumers. These also have the ability of IVIS interface, which immensely helps with Day/Night lighting modules to reserve energy. They also allowed space to be customizable to different colors depending on the need. Since the embedded light system is communicating with a framework via a wireless network, a simple change from a computer can trigger any necessary changes in the light's colors or mode within a matter of seconds. The Ultrasonic Mini Sensor is EMC and CE certified and comes with ultrasonic transducers for better size efficiency. With the ultra-flat design, this module also has an ultrasonic sound sensor for wayfinding capabilities that are inaudible to humans. One more variant of Ultrasonic Diagonal Sensor uses sound sensors to make decisions on parking spot availability. It achieves this feat with the emission of ultrasonic waves and echoes to calculate the distance to receive data and make decisions based on that. This version is also CE, and UMC certified.

Indect's camera-based Sensors are the most sophisticated and state-of-the-art implementation of smart parking. This system has been fully automated and does not wait or require manual human inputs for any of its functionalities. This line of camera-based solutions is called UPSOLUT, and one of these sensors has the capability of detecting up to six parking spots at a time. Their cameras are also waterproof and dust resistant to keep up with weather calamities. The IP67 sensor, which is the only implementation of such sensors in commercial production, allows the UPSOLUT to even function submerged in water and airtight sealing, making the maintenance of the sensors to a very minimum. With video recording and sound alert features, this system allows wayfinding for end-users,

which is really convenient. This particular line of products also comes with kiosks installed in the parking garages for car-finding activities. Since this particular system comes with the license plate recognition, utilizing the UPSLOUT app, consumers can quickly locate their cars. Any sort of activity in the parking lot gets sensed by the sensors within the cameras that trigger the video recording process, which is later streamed. The fisheye camera allows video capture of up to QXSGA resolution at 25 frames per second. The PTZ support in the video streams is what makes license detection and recognition possible. The stream is also defaulted to a black and white format with 100 seconds max intervals to maximize storage efficiency. The advanced machine learning algorithms for vehicle detection within the system make UPSOLUT the most accurate camera-based system available in the current market.

Other than this, Indect also has a Surface Mounted Sensor embedded product line. This particular product line utilizes induction pads that are embedded with electromagnetic vehicle detection sensors. This enables extra features like vehicle count calculation at the toll booths. These magnetic field vehicle detectors use X, Y, and Z-axis detections of the earth's magnetic field. Each sensor is continually on the lookout for any abnormalities within the components of the magnetic field. Any new data entry within the X, Y, and Z components suggests vehicle movement and makes decisions on parking space detections.

With the many different implementations of different sensors as well as the usage of camera sensors in order to implement a smart parking environment makes Indect one of the pioneers in the smart parking industry. Usage of mobile applications to stream video in real-time and car way-finder feature also sets the company apart from other companies in the innovation aspects. The Smart Parking project implements a lot of theories and technologies that Indect has already utilized in a more commercial manner. This infrastructure of technologies can be an immediate motivation for the Smart Parking project to base off of.

4.1.2 ParkEagle

Parkeagle is one of the up-and-coming companies researching smart parking and broadening its focus to smart city communications. Parkeagle enables their smart parking system and caters the solution to the end-users through a real-time information update through mobile apps. Parkeagle utilizes ultra-low-power smart parking sensors to detect the parking spot status. These sensors communicate via server to display the data calculated on an app and even digital road signs. A lot of cities in Europe are utilizing ParkEagle's smart parking system. Parkeagle also implemented a StreetEagle cloud system to take the data from the sensors as input and send it to a digital server, where algorithms are run to make decisions on parking occupancy, traffic flows, and also vehicle types. Later these data are presented in the Park Eagle management system for the end-users. The app also has a geofencing feature that allows the drivers to know traffic flow and parking conditions at a given spot ahead of time. The dashboard is catered to fulfill the needs of parking condition information and traffic data for end-users.

Parkeagle has also created technological advancements in modern parking systems and vehicular communications flow in general. The usage of sensors and their capability of

communicating directly with cloud systems can indeed revolutionize the concept of smart parking. Since all the algorithmic computation is being done on the cloud, real time communication between cloud and end - user side is much more efficient. Their mobile parking app and digital parking signs also add to the convenience to the catered audiences. This project's smart parking app could use the cloud methodologies of Parkeagle for better communication between webapps and the camera sensors. The machine learning algorithm taking place in the cloud instead of the web server would enable quick run time of real time video streaming as well as quick data update on consumer user end.

4.1.3 CleverCiti

CleverCiti adds to the advancements in the parking system with a turn-by-turn parking guide for their end-users. CleverCiti is working with municipal companies to reduce local traffic and unnecessary emissions and solve the drivers' current parking issues. They are helping local companies to streamline and boost parking revenues with the advancements in parking technology. They have introduced the usage of battery-powered sensors on lampposts to monitor traffic and parking availability in city areas. The company is improving the on-street parking situation in urban areas such as parking lots at shopping centers, corporate offices/campuses, and train stations.

CleverCiti utilizes overhead sensors made in Germany that allows detection of parked cars and vacant parking spots. However, for smoother communication and to optimize run time, these sensors only send the GPS coordinate data sets to the server of the available spaces only. This helps the system gather relevant data and also maximizes the memory storage uses. The sensors on the lamp posts also are able to support up to four smart sensors and power them around the clock. This aids in smart city deployment with the ability to send traffic congestion data, vehicle mode data and of course street side parking data. These sensors also can communicate via an LTE network as well as the existing power line connected to the posts.

The smart parking system of CleverCiti has a unique feature where it accurately guides the drivers to the parking spots that are available. They have implemented this GPS-like feature only within specific areas of municipal parking areas. This saves a lot of time for the end-users as they know exactly which spot to go towards to attain a vacant parking spot. Along with these, CleverCiti has commercialized a subscription-based mobile app system that allows their customers to purchase or subscribe to their interface and solution set. This way, the permit holder and monthly subscribers get optimum parking notifications and facilities through the app. They also have taken extra steps for their customers to opt-in for a dynamic reservation feature, where customers may look ahead to reserve a vacant parking spot ahead of time and avoid all the hassles.

CleverCiti has taken smart parking to another level when it comes to commercializing a subscription-based system. They have implemented and focused on the customer service aspect of parking and really put emphasis on catering their goals and vision to the end users' comfort. The subscription-based system and the dynamic reservation set the entirety of the system apart from any other smart parking technology. The current smart parking

project can implement the innovative features of CleverCiti's mobile app to incorporate further assistance and improve the overall user experience.

4.1.4 Smart Parking Limited

Award-winning company Smart Parking has been a world leader in technological design and advancement in the development and management of parking systems. With their intelligent parking structure, they have improved parking experience in shopping and retail, supermarkets, airports, hospitals, and even universities. The company is emphasizing the concept of a smart city through the development and improvement of smart parking systems. The connection of IoT devices across the city through their SmartCloud system would aid in the global implementation of the intelligent parking concept of the company. SmartCloud, developed by Smart Parking Limited, is a globally scaled, real-time, internet of things platform. This cloud platform was deployed parallelly with the deployment of the Google cloud. This platform enables the company to optimize in-house data collection and management.

The company's smart sensors and smart devices communicate directly with this platform. The integration of the SmartCloudAPI enables complete connection with the assets to provide real-time support. Even the firmware support for the smart sensors and remote management has been made possible and widespread with the usage of the integration of SmartCloud. With these advantages, this company has revolutionized the parking experience with a robust mobile app called Tessera. This application is developed to aid urban city areas and municipalities to be efficient with their parking management. This end-to-end compliance management system incorporates all the advantages provided by the company to provide an efficient solution suited for any parking management system. This app provides dashboards that are supported in real-time to showcase the live visual activity of the parking site in a visual map form.

Google Maps is greatly incorporated within the app, which provides detailed real-time information on traffic tariffs and parking citations for specific areas at any given time. It helps users find vacant bays for parking and offers a feature to give directional support to the users to avoid any parking discomfort. The app goes as far as detecting disabled parking availability, limited-duration parking as well as pay booths for paid parking. Within the app, users can locate the history of their parking experiences and review specific invoices or location experiences. This app also can be modified depending on industry use. The company allows a fully customizable parking app for its client according to their own needs. Certain companies in certain cities may have particular needs that one generic application may not be able to provide. Thus with the implementation of SmartCloud, the company has implemented the opportunity of customizing the mobile app according to the consumer needs.

The network gateways implemented within the parking system require multiple sensors across areas. Thus for any parking system, multiple sensor units called SmartSpots are implemented. These individual units are scattered to gather precise data on the parking area status from the sensors embedded within. Some of the SmartSpots use sensors that can communicate with the cloud system in wireless form, which is important in areas where

devices do not need to communicate with each other. However, there are also sensors deployed that communicate with each other via fiber optic or ethernet.

Along with all these web-based technologies implemented, there are also real-time supported display signs that the company has issued. These signs are called signage and are used to communicate with the users and show the traffic flow status, available parking spot count, and much more. Using digital network communication, these signs provide live updates with the data gathered directly from SmartCloud. One neat feature these LED signs offer is the directions. According to the number of available spots, these signs provide directional data flashed on the LED. There are multiple uses for these signs. For level-by-level implementation, signage comes with the total vehicle count along with the directional data view. Other than these overhead displays, some signs can be mounted on walls to provide support for outdoor parking areas as well as the parking entrances and exits. The color is dynamically customizable by the administrator with different daytime and nighttime modes for energy conservation. The sensors implemented have their customizations as well. Inground sensors have been developed to better communicate with SmartSpot gateways using ANPR. These sensors are water-resistant and casings are designed to withstand natural calamities. Mainly for outside parking spots, these sensors are utilized. The surface-mount sensors aid in the easy installation aspect. Their sensors work perfectly with exposed sites with not fully supported surfaces. Roof, cable support - where further construction work is impossible, these surface mount sensors work perfectly.

Embedded within the floor, these sensors are designed to provide better visibility with LED settings to support visibility for increased traffic and visual challenges on the surfaces. Lastly, the most popular overhead indicator sensors are used for parking garages with many complex routes for parking. These sensors are supported with high visibility LED lights to make it easy for motorists to locate the desired parking spot. These parking sensor lights can be controlled dynamically according to the business needs of the administrator. The RGB LED colors are utilized for every one of the sensors, which also communicate directly with the SmartCloud API to provide data in real-time to the digital application at the end-users; discretion. Along with all these innovative technologies already implemented, the Automatic Number Plate Recognition feature aids in the police and traffic management facilities. Otherwise known as License Plate Recognition, the technology allows monitorization of vehicle registration plate recognition features to better secure and manage the car parking infrastructures. For toll agencies and government pay-per-use highways, this ANPR technology has aided. High-capacity cameras with advanced machine learning algorithms incorporating the perspective of lighting conditions and angles allow extremely accurate ability to scan and decipher each license plate. Depending on mirrored plates and lighting conditions, these camera sensors also use IR optimization to work with different lighting modes, allowing efficient and accurate picture capturing at any time. This technology has been used for paid parking lots as well, where the sensors send the data directly to SmartCloud, where an automated payment script is in charge of payment collection and billing the users based on license number.

Smart Parking Limited indeed provides one of the most developed and robust support for day-to-day parking complexities. With the implementation of the company's cloud

platform, this system has the potential of growing and advancing vastly along with the tide of time. The business perspective of this system has a lot of potential with the ability to cater to the consumers based on metrics like locations, terrains, and other necessary needs. Implementation of a global mobile application keeps the implementation of the system up to date and easy to maneuver for the client. The different sensor applications also cater to different needs based on infrastructure and location. License plate recognition brings another realm of possibilities in the automated billing opportunities for municipal bug areas and government facilities. Smart Parking Limited is a pioneer in changing the path of parking technological advancements. Even though the implementation of such a large-scale system is not possible for the current smart parking project, the abundant opportunities for growth and improvement within the intelligent parking business are without a doubt a primary motivator for the project.

5.0 Technology Research

In this section, we will introduce the potential technologies to be used in our Smart Parking System. The areas that will be discussed include computer vision, cameras, microcontrollers, LED matrix displays, and servers.

5.1 Computer Vision

Computer vision is an area of artificial intelligence that allows computers and systems to derive meaningful information from images, videos, or other visual inputs and make recommendations based on that information. Ultimately, computer vision aims to give computers and systems the ability to see and perceive the world in the same way that humans do. For the purpose of our project, we will need to use computer vision in order for our system to accomplish five tasks:

1. Detects the position of all available parking spaces.
2. Identify whether that parking space is open or occupied.
3. Uniquely identify cars by color or license plate.
4. Differentiate between a parking space being occupied by a human or a vehicle and react accordingly.
5. Calculate how long a car occupied a spot for.

We will need to use a computer vision library to achieve these tasks. While there are a variety of libraries that could meet our needs, including OpenCV, Tensorflow, and SimpleCV, our group has decided to work with OpenCV. This is mainly due to OpenCV having more algorithms to work with than any other library, a large support community that will help us become proficient with this library, and our group already has some experience with using OpenCV from classes we have taken at UCF.

5.1.1 OpenCV

OpenCV is an open-source computer vision and machine learning software library. It has a broad area of applications and provides thousands of computer vision algorithms relating to image processing, video analysis, object detection, and object identification

segmentation, just to name a few. This library was written in C++, and its primary interface is in C++; however, it supports a wide variety of other programming languages, including Java, Python, and Matlab. In the case of Python, this is due to the use of wrappers where the computationally intensive code in C++ is working in the background while the user is able to use customizable pre-defined functions. For our project, we would like to use Python due to it being a general-purpose programming language that is simple, easy to learn, and more readable than other languages.

What separates OpenCV from other computer vision libraries is the fact that it was designed for real-time vision applications. This will be useful for our smart parking system since we will always be working with a live dataset rather than a pre-recorded dataset. Additionally, we would like the correct number of open parking spots to be reflected on our LED sign within 30 seconds of a change, and working with a computer vision library that is designed for real-time applications could help out with this.

5.1.2 Computer Vision Techniques

To achieve the goals described in previous sections, we will need to implement a combination of computer vision techniques. This section will describe the techniques that we could expect to apply in our system.

5.1.2.1 Edge Detection

Edge detection is a computer vision technique used for identifying the points in a digital image where there is a sharp change in image brightness. These points where the image brightness changes sharply are called edges or boundaries. This technique is effective as it can be used to extract useful structural information from an image and drastically reduce the amount of data to be processed. This will be a likely technique to be used in our system as we will need our camera to be able to identify what a parking space is from a digital video feed. Since there is a sharp change in brightness between the pavement and the lines painted on the ground to designate a parking spot, these would show up as edges in the image, as shown in Figure 5.

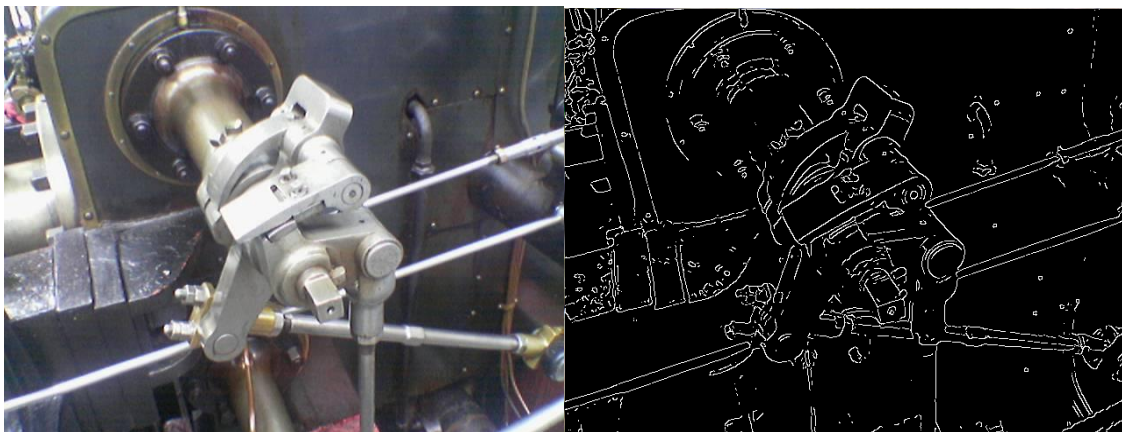


Figure 5: Example of Canny Edge Detector [15]

There are a variety of edge detection techniques, including the Prewitt edge detector, Laplacian edge detector, and the Sobel edge detector; however, the most commonly used edge detection technique is the Canny edge detector which is shown in the figure above. While this method is more complex than other edge detection methods, it is also the most effective. This technique uses a multi-stage algorithm which can be defined by the six steps below:

1. Convert digital image or video to grayscale
2. Apply a Gaussian filter in order to smooth the image and reduce noise
3. Find the intensity gradients of the image to help identify edge intensity and direction
4. Apply non-maximum suppression to thin the edges in the image and eliminate false responses to the edge detector
5. Apply a double threshold to determine strong, weak, and irrelevant edges in the image
6. Use hysteresis edge tracking to convert weak edges into strong ones only if there is a strong edge close to it

While edge detection is an effective method for extracting useful structural information from an image, there are a couple of drawbacks that should be mentioned. The first is that the size of the output image will be shrunken compared to the input image due to the fact that the filters used for edge detection shrink the size of the image. Since the size of the image is shrunken, this creates another drawback where there is a loss of a lot of valuable information, especially from the outer perimeters of the image. With this said, there is a technique to combat these drawbacks called image padding where additional pixels are added to the perimeters of the image to avoid losing the valuable information from the input image.

5.1.2.2 Hough Transform

The Hough transform is a feature extraction technique that can be used to detect any shape in an image as long as that shape can be represented in a mathematical form. Since a straight line can be represented as $y = mx + b$ in the Cartesian coordinate system or as $\rho = x\cos\theta + y\sin\theta$ in the polar coordinate system, this method can be used to detect the straight lines that define parking spaces. With this being said, the polar coordinate system is primarily used in the Hough transform as the Cartesian coordinate system is not able to define vertical lines.

This method is most effective after edge detection has been applied to the image as the useful information that will be needed for the Hough transform has already been extracted. The Hough transform looks at each individual edge point defined by the edge detector and then looks at other edge points within the image to see if there is an intersection. Once the Hough transform sees that there is a high amount of intersections between a set of edge points, it can make the conclusion that a line can be marked through those points as shown in Figure 6.

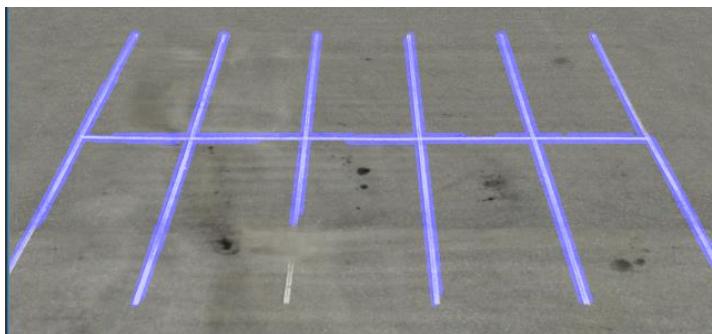


Figure 6: Hough Transform on a Parking Lot [17]

This is effective because the output image from an edge detector is still an image described by its pixels; however after the Hough transform has been applied, we now have an image that is defined by the characteristic equations representing a parking spot, further refining the amount of data to be analyzed from the image. With this said, there is one thing to mention about using this method being that in order for it to perform best, we must begin from a set of empty parking spaces. Using parking spaces that are already occupied could create some inaccuracies from the output of the hough transform.

5.1.2.3 Regions of Interest

After the lines defining a parking spot have been drawn onto the image by the hough transform, we would then need to define the specific areas of interest, i.e., each individual parking spot. One method of doing this is using the intersections between the vertical and horizontal lines defined by the hough transform in addition to the endpoints to extract a list of x and y coordinates. From this list of 2d points, we could then map out the corners of each parking spot and group them into sets of 4 such that a unique box is created for every spot.

An alternate method of mapping out the regions of interest would be to use the OpenCV Mouse as a Paint-Brush function. This would give us the freedom to manually draw the bounding boxes for each area of interest, however, our system would require human input before it is able to work properly. This means that our system would not be completely autonomous and we believe it is a method that should only be pursued if the autonomous method described above does not work for us.

5.1.2.4 Object Detection

After we obtain a layout of the parking spots to be monitored, we will then need our system to be able to identify whether a parking spot is occupied or not. This task relates to object detection and there are quite a few ways we could go about accomplishing this. This section will describe a few methods that are available, some initial thoughts to consider for each of the methods, as well as compare each of them and present a final selection.

5.1.2.4.1 Cascade Classifiers

Object detection using Haar feature-based cascade classifiers is a machine learning based approach to this concept. With this method, a cascade function is trained from a lot of

positive and negative versions of images and then used to detect objects in other images. Essentially, this method tells OpenCV exactly what features to look for in an image and then makes the decision of whether a certain object exists or not through the use of convolutional matrices. For example, cascade classifiers can be used for face detection as shown in the Figure 7 below. The cascade classifier looks for eyes and then uses this information to conclude if there is a face or not.



Figure 7: Cascade Classifier for Face Detection using Eyes as Feature [3]

For our case of needing to detect cars in an image, pictures with cars occupying parking spots would be the positive images, and pictures with empty parking spots would be the negative images needed to train the cascade function. The program would then need to extract features from each of these images and see if there are any similarities between images in order to conclude if a car is in the image or not. With this said, there are already cascade classifiers that exist for car detection, however, we could probably make this method more accurate by taking our own sample pictures of parking spaces at UCF. While our current method of counting the cars in an image involves actually detecting a car, haar cascades could also be effectively used for other strategies of counting cars as explained in Section 10.2.1.

5.1.2.4.2 You Only Look Once (YOLO)

The You Only Look Once (YOLO) algorithm is a deep learning-based approach to real-time object detection. This method differs from other deep learning methods as the YOLO detector can be defined as a one-stage detector. This means that YOLO will outperform other deep learning methods such as the Region-Based Convolutional Neural Network (R-CNN), which is a two stage detector, however, what it gains in speed it loses in accuracy. Though, this is not to say that YOLO is not an accurate algorithm, it is just not as accurate as two stage detector algorithms.

Functioning as a one stage-detector means that YOLO requires an image or video feed to pass through its network only one time, hence the name. This is possible because YOLO reasons with an entire image rather than successively examining several regions in an image to detect the objects that are present in it.

This algorithm begins by dividing an image into ‘S’ x ‘S’ cells. It then uses a unique neural network using the characteristics of an entire image or video to predict multiple boxes with every one containing a specific object. If the center of an object is in one of these cells, this cell will now be responsible for detecting the object. Depending on if an object exists in a cell or not, the algorithm will now assign a score to each cell representing the level of confidence for the object present in the box. If an object does not exist in the cell, the score

should be zero. From here a convolutional neural network based on the GoogLeNet neural network is used to provide a class specific confidence score for each box. The output of the YOLO detection network will be the original input image or video with localized object detection and their respective classes attached to it as shown in the Figure below.

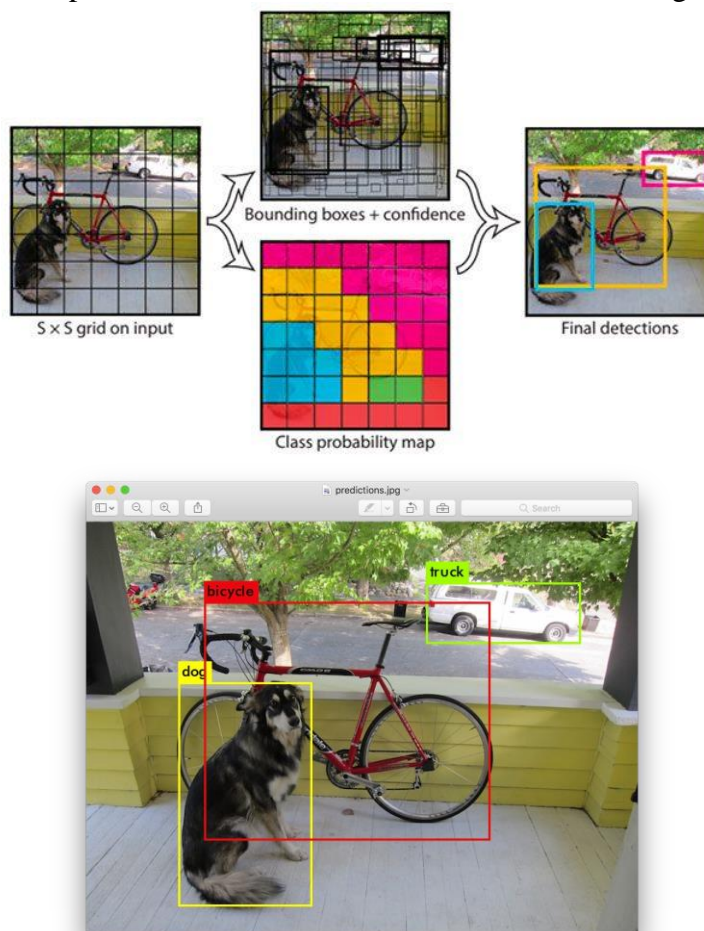


Figure 8: Example of YOLO Object Detection [33]

There is a variation of the YOLO object detector called Tiny-YOLO, which is a smaller version of YOLO; however, it is less accurate than YOLO. We will need to verify just how accurate each of these methods is through testing.

5.1.2.4.3 Object Detection Method Comparison

To summarize the object detection methods described above, Cascade Classifiers are a straightforward method to go about object detection as you are simply using convolution matrices to extract the features from an image. This is a simple, quick, and easy method; however, we expect it to not be as accurate as other methods. On the other hand, the YOLO detector is a highly accurate method for object detection; however, the implementation of this method would be very complex. A method that could prove to be a bit less complex but still share many similarities with the YOLO method is the Tiny-YOLO method, though this will need to be verified.

After considering each of these methods, we realized that the one that suits us best would depend on the strategy that we use to count the number of available parking spots. If we go with our original method described in previous sections of detecting the car itself, then YOLO or Tiny-YOLO would be the best option for us. If we decide to pursue another strategy to count available parking spots, such as the ones described in section 10.2.1, then Haar Cascades may be the more suitable option to fit our needs. With this said, we will not know which method best suits us until it is verified through testing. The table below shows a comparison of the object detection methods we are considering. It is important to mention that this table is not 100% accurate and is only based on speculation from the research we have done.

Table 3: Object Detection Method Comparison

| Method | Complexity | Speed | Accuracy | Power Need |
|---------------|------------|-------|----------|------------|
| Haar Cascades | Low | High | Low | Low |
| YOLO | High | High | High | High |
| Tiny-YOLO | Medium | High | Medium | Medium |

5.2 Cameras

The camera is a vital part of our Smart Parking System because it is the sensing element that is capturing the data to be used by the other components within our system. Our system will use two cameras mounted on the two adjacent sides of a row of parking spaces. The cameras will be facing each other such that they are keeping track of the unoccupied and occupied parking spaces on the side of the row opposite which they are mounted.

With there being such a large variety of cameras that could be implemented in our project, our group created a list of specifications that we would like to see from the camera. First, we expect the camera to have at least an IP65 weatherproof rating such that the system is still functional in rainy conditions. Second, we expect the camera to be designed to handle lowlight conditions such that the system is still functional at night time. Third, we expect the live video feed coming from the camera to be easily extracted such that we can easily implement computer vision technologies in real-time. Fourth, we expect the camera to be Power over Ethernet (PoE) compatible in order to easily transmit data coming from the video camera, increase the distance we are able to wire our cameras and decrease the number of wires that need to be used in our system. Lastly, we expect our camera to have a field of view between 70 and 105 degrees such that the amount of required parking spaces in our system is captured while preserving video quality.

With this said, our group understands that we do not want to spend any more than \$1,500 on our project and are willing to change some of the specifications for our camera in order

to adhere to this constraint. The types of cameras that we have decided to consider for our project include the OpenCV AI Kit PoE cameras and IP cameras.

5.2.1 OpenCV AI Kit (OAK) Cameras

The OpenCV AI Kit (OAK) is a family of cameras that embed performant spatial sensing, neural inference, and computer vision functionality. They are driven by Intel's Movidius Myriad X Vision Processing Unit (VPU), which is a type of microprocessor that allows all computations to be done at the camera level, completely offloading the robotics perception of our system to the camera itself. All of the OAK cameras employ Sony's IMX378 RGB image sensor, which allows for a maximum frame rate of 60 fps at a resolution of 12 megapixels, autofocus, and a display field of view of 81 degrees. Additionally, each of these cameras is situated with a ¼ - 20 tripod mount on the bottom of the unit, which will ease the process of lifting our cameras to a high enough level to view the parking spaces.

The OAK camera family is separated into three groups including the USB line, PoE line, and the IoT line. We have decided to work with the PoE cameras as using ethernet cords will provide us with sufficient cable length between the camera, our ethernet switch, and our PCB. Additionally, the PoE cameras come with IP67-rated housing. The PoE line consists of three cameras, including the OAK-1 PoE, OAK-D PoE, and the OAK-D Pro PoE. Each of these cameras is further described below.

5.2.1.1 OAK-1 PoE

The OAK-1 PoE is the baseline camera of the OAK PoE line. It employs just a single RGB image sensor camera which pipes its video feed directly into the Myriad X VPU and DepthAI for AI processing. Since the OAK-1 PoE utilizes onboard python scripting, it provides a variety of functionalities at the camera level, including object tracking, corner detection, feature tracking, custom computer vision functions, and neural interference. Additionally, if we did want to extract a recorded video or live video stream, this camera also provides H.264, H.265, and MJPEG encoding. With all of these functionalities, the unit price of this camera is \$249.

5.2.1.2 OAK-D PoE

The OAK-D PoE differs from the OAK-1 PoE in the fact that it uses three onboard cameras rather than just one. In addition to the single RGB image sensor, it also includes two OmniVision OV9282 grayscale image sensors capable of producing video at a max frame rate of 120 FPS with a resolution of 1 MP. Using these two additional cameras gives the OAK-D PoE the capabilities of 3D object localization, 3D object tracking, and depth perception on top of the functions already available with the OAK-1 PoE. The addition of the capabilities brings the unit price of this camera from \$249 with the OAK-1 PoE to \$299.

5.2.1.3 OAK-D Pro PoE

The OAK-D Pro PoE is essentially the same exact camera as the OAK-D PoE; however, the OAK-D Pro PoE is designed to handle low-light situations. It accomplishes this through active infrared illumination in the form of the Belago 1.1 Laser Dot Projector. Referring to the Figure 9 below, the functional block of the Belago 1.1 is shown.

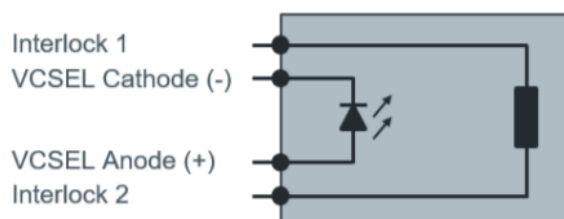


Figure 9: Functional Block of Belago 1.1 [2]

Vertical cavity surface-emitting lasers (VCSEL) are a type of laser diode where laser beam emission happens perpendicular to the top surface. This laser dot projector illuminates the area in the camera's field of view using 4700 laser dots and pretty much acts as a flashlight for the camera. This helps with disparity matching in low light situations, especially for blank surfaces with little to no texture, such as a wall or a floor. It is also important to mention that this laser dot projector meets the class one specification for lasers meaning that it will cause no harm to the human skin or eyes. In addition to the laser dot projector on this camera, it also uses an infrared LED floodlight to assist in low-light situations. The addition of these night vision abilities brings the unit price of the OAK-D PoE from \$299 to \$399.

5.2.2 IP Cameras

The second type of camera we are considering for our project is an IP camera. While using an IP camera would provide a very budget-friendly option, it would also introduce some unique challenges that could have a negative impact on our system.

One of these challenges is that no computer vision computations would happen at the camera level. Since these types of cameras do not include a microprocessor, we would need to extract the live video feed into a Python program via IP address using the OpenCV library and then break the video into individual frames before being able to apply computer vision techniques. Not only would this make our system have to handle large amounts of data being transmitted between the camera and the local server, but there would also run the risk of us having to work with a python program that could become very complicated. Additionally, IP cameras typically only come with two mounting options: a wall mount or a ceiling mount, in contrast to the OAK cameras, which come with a tripod mount. This would add complexities relating to what we would mount the cameras on since we can't mount the cameras wherever we see fit with the fact we will be testing our system on public parking lots and garages. After looking at multiple IP cameras, the two that best fit the needs described in section 7.1 were the ANNKE C800 and the RLC-810A.

5.2.3 Camera Comparison

Table 4 shows the criteria that we used to compare each of these cameras.

Table 4: Camera Comparison

| Model | OAK-1 PoE | OAK-D PoE | OAK-D Pro | ANNKE | RC-810A |
|-------|-----------|-----------|-----------|-------|---------|
|-------|-----------|-----------|-----------|-------|---------|

| | | | PoE | C800 | |
|-------------------------------|------------------------|--------------------------|--------------------------|----------------|----------------|
| Type | Microprocessor | Microprocessor | Microprocessor | IP Camera | IP Camera |
| Unit Price | \$249.00 | \$299.00 | \$399.00 | \$79.99 | \$84.99 |
| Size (WxHxD) | 81.9 mm x 81.9 x 31 mm | 130 mm x 65 mm x 29.9 mm | 111 mm x 47 mm x 31.1 mm | 155 mm x 70 mm | 192 mm x 66 mm |
| Video Resolution | 12 mp | 12 mp | 12 mp | 8 mp | 8 mp |
| Speed | 60 fps | 60 fps | 60 fps | 30 fps | 25 fps |
| FOV (degrees) | 81 | 81 | 81 | 102 | 87 |
| Power | PoE 802.3af | PoE 802.3af | PoE 802.3af | PoE 802.3af | PoE 802.3af |
| Designed for Low Light | No | No | Yes | Yes | Yes |
| Depth Perception | No | Yes | Yes | No | No |
| Weatherproof Rating | IP67 | IP67 | IP67 | IP67 | IP66 |
| Easy Implementation | Yes | Yes | Yes | No | No |

5.2.4 Final Camera Selection

After considering the cameras analyzed in the sections above, we have decided that the OAK-1 PoE would be the best camera to use for our project. Although this camera is not designed to work in lowlight conditions, it still satisfies all of our other requirements while remaining friendly to our \$1,500 budget, as we would be paying \$249 per camera. The OAK-D Pro PoE is the only OAK camera designed to function in low light situations; however, this camera also offers a variety of features that would be overkill for the system we are designing, and we are not willing to spend \$399 per camera.

The OAK-1 PoE, shown in Figure 10, offers full IEEE 802.3af, Class 3 PoE compliance with 1000BASE-T speeds (1gbps) for both communication and power. As mentioned in previous sections, this camera is driven by the Movidius Myriad X VPU, and DepthAI is what we will be using to communicate with this VPU and complete all AI and computer vision computations.



Figure 10: OAK-1 PoE [1]

5.2.4.1 Movidius Myriad X VPU

The Movidius Myriad X VPU is the component of the OAK-1 PoE that will allow all AI and computer vision computations to be completed from the moment video is captured before any type of data communication happens through ethernet. This VPU will contribute heavily to making our system fast as it promises 4 Trillion Operations Per Second (TOPS) of processing power (1.4 TOPS for AI applications).

This version of the Myriad X VPU used in the OAK-1 PoE makes use of a new deep neural network developed by Intel called The Neural Compute Engine, which is specifically designed to run deep neural networks at high speed and low power. This will be effective for our project as deep neural network inferences is a possible method we could use to detect available parking spaces and make the decision of whether it is open or occupied. With this said, the Myriad X VPU offers features including the ability to run any custom-built AI models, video encoding, computer vision, and object tracking. We will have access to these features through DepthAI, which is further explained in the next section.

5.2.4.2 DepthAI

DepthAI is an open-source hardware, software, and AI-training platform built around the Movidius Myriad X VPU. It focuses on the combination of 5 key features being artificial intelligence, computer vision, depth perception, performance (high resolution and FPS, multiple sensors), and being an embedded, low power solution. Together, these features allow DepthAI to be a spatial AI + computer vision platform that gives robots and computers the ability to perceive the world as a human can. DepthAI uses a Python Application Programming Interface (API) to connect to, configure, and communicate with the OAK devices within a system as shown in Figure 11.

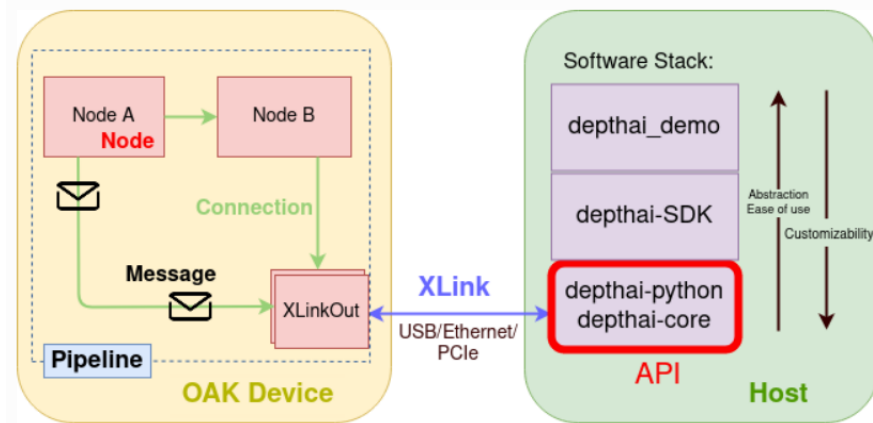


Figure 11: High-Level DepthAI Software Architecture [1]

The following list highlights the important aspects of Figure 11:

- The **Host Side** is the computer that the OAK device is connected to. This can be a Raspberry Pi, Windows PC, or any other compatible computer. Multiple OAK devices are able to be connected to one host and uniquely identified.
- The **Device Side** is the OAK device itself. If anything happens on this side, it means that it is running on the Movidius Myriad X VPU.
- The **Pipeline** is a complete workflow that consists of nodes and the connections between them. When we receive our OAK cameras, we would first need to create this pipeline, populate it with nodes, configure the nodes and the connections between them, and then load it onto the OAK device.
- The **Nodes** are the building blocks of the pipeline. Each node provides a specific functionality, a set of configurable properties, and inputs/outputs. An example of a node would be the EdgeDetector which receives two inputs, an input image and an input configuration, and then produces one output, the output image. Once a node is created and configured as desired, it can then be linked to other nodes.
- The **Connection** is the link between one node's input and another node's output. These connections define the pipeline data flow and establish where messages should be sent in order to achieve an expected result.
- **Messages** are the communication that happens between linked nodes. Messages sent between linked nodes are the only way they are able to communicate with one another.
- **XLink** is a middleware capable of exchanging data between an OAK device and the host. XLinkIn is used to send data from the host to the OAK device, and XLinkOut is used to send data from the OAK device to the host.

It is also important to note the DepthAI Software Development Kit (SDK) which was built on top of the Python API library. The DepthAI SDK is a Python package containing classes and functions that will help in performing common tasks while using the Python API. The package primarily consists of managers which handle different aspects of development as shown in Table 5.

Table 5: DepthAI SDK Classes and Functions

| Classes and Functions | Description |
|--------------------------------------|---|
| depthai_sdk.managers.PipelineManager | Helps in setting up processing pipeline |
| depthai_sdk.managers.NNetManager | Helps in setting up neural networks |
| depthai_sdk.managers.PreviewManager | Helps in displaying video previews from OAK cameras |
| depthai_sdk.managers.EncodingManager | Helps in creating videos from OAK cameras |
| depthai_sdk.managers.BlobManager | Helps in downloading neural networks as MyriadX blobs |
| depthai_sdk.fps | For FPS calculations |
| depthai_sdk.previews | For frame handling |
| depthai_sdk.utils | For various most-common tasks |

5.2.4.3 OAK-1 PoE Electrical Characteristics

Protecting the OAK-1 PoE from any electrical damage is vital as this would have a negative impact on its reliability, therefore, it is important to consider the electrical characteristics of the camera. Below are two tables, Table 6 and Table 7, showing the absolute maximum ratings and the recommended operating conditions of the OAK-1 PoE.

Table 6: Absolute Maximum Ratings of the OAK-1 PoE

| Symbol | Ratings | Min | Max | Unit |
|------------------|--|-----|-----|------|
| V _{PoE} | 802.3af, Class3 input supply voltage range | 37 | 57 | V |
| I _{PoE} | Maximum Input Current Requirement | | 350 | mA |
| T _{stq} | Ambient Temperature | 0 | 60 | C |

Table 7: Recommended Operating Conditions of the OAK-1 PoE

| Symbol | Ratings | Min | Typ | Max | Unit |
|------------------|--|-----|-----|-----|------|
| V _{PoE} | 802.3af, Class3 input supply voltage range | 37 | | 57 | V |
| P | Power Consumption Requirement | 4 | 5 | 7.5 | W |

| | | | | | |
|------------|---|--|-----|----|---|
| P_{IDLE} | V_{BUS} Idle Power Draw (Myriad X Booted) | | 2.5 | | W |
| T_A | Ambient Operating Temperature | | | 50 | C |

According to Luxonis, the power usage for the OAK-1-PoE tends to range between 1.94 W (at standby) and 4.56 W (at max power consumption). During normal operation, we can expect the camera to pull about 4.2 MW. With this said, the ethernet cable used in our system should have a CAT5E rating or higher. This will allow us to achieve the 1 gigabit per second of data transfer promised by this camera as well as meet PoE requirements. Additionally, on the device end, the ethernet cable should use a shielded 8-pin RJ45 connector which should not be an issue since this is standardized.

5.2.4.4 OAK-1 PoE Mechanical Information

The OAK-1 PoE is a fairly small camera with a height of 81.9 mm (114 mm if you include the tripod mount) and a width of 81.9 mm, a length of 31 mm, and a weight of 294 grams. Figure 12 shows the physical dimensions of this camera.

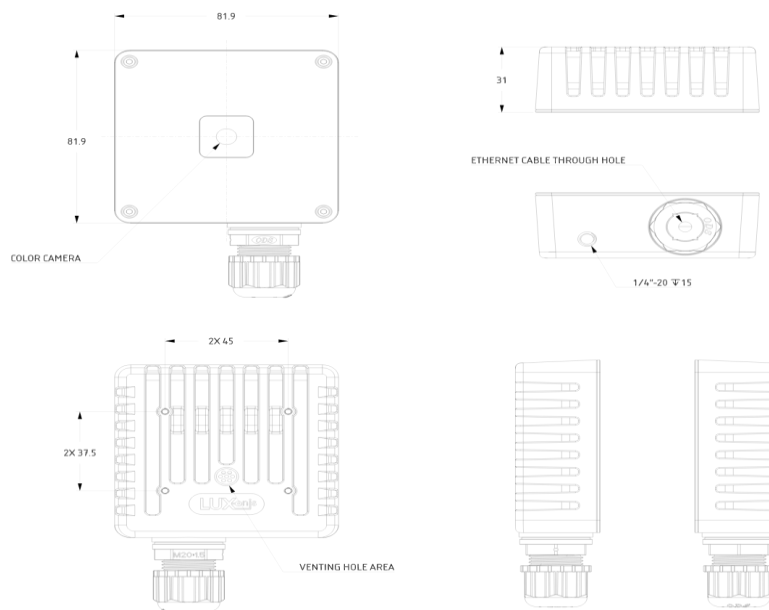


Figure 12: OAK-1 PoE Mechanical Measurements [1]

5.3 Microcontrollers

The microcontroller is responsible for driving the LEDs to display where available parking is. Since the local server will handle the computation of available parking, the microcontroller is relieved of performing any space computations of its own. Instead, the MCU will solely display messages to drivers indicating how many spaces are available in a region and which way to drive to the available space. Through daisy-chaining, multiple LED matrix displays will be driven by the same microcontroller. This relieves the need for multiple boards, each driving individual displays.

The PCB on which the MCU shall be integrated will be connected directly to an external power source. Therefore, no particular power constraints will need to be considered for the microcontroller selection. An additional factor is whether to choose a microcontroller with an integrated Ethernet controller, which could greatly reduce the complexity of the PCB design. The tradeoff is that the maximum data rate supported is 100 Mbps compared with the 1 Gbps supported data rate from a separate Ethernet controller. This tradeoff is a total non-factor, though, in a practical sense because the amount of data to be transmitted to the microcontroller is so minuscule compared to the allowable bandwidth. Therefore, the supported 100 Mbps data rate for most integrated Ethernet controllers will suffice.

In order to drive the LED display(s), there must be enough available pins from the MCU to address all of the LEDs. Normally, the number of pins needed could be simply calculated from the below equation.

$$Pins = Rows + Columns$$

Since the 32 x 64 LED display is 2048 LEDs in total; the MCU must have enough pins to address the LEDs via multiplexing. Charlieplexing was considered, but it has limitations to its abilities when it comes to displaying RGB and changing brightness. But these pin considerations are not relevant in our case since the selected display makes use of shift registers and allows for much fewer pins (16) in order to address a greater number of LEDs. Since each LED has 3 colors (RGB), they take approximately 3 bytes in RAM. This results in 6144 bytes. When accommodating for a second display to be daisy-chained, the byte count doubles to 12288 bytes. This puts the bare minimum RAM size at around 12 KB. According to SparkFun's product description, for a similar LED display at 32 x 32, a minimum clock frequency of 16 MHz is needed to display an image with tolerable flicker, so a microcontroller capable of running beyond that frequency is needed to drive the LEDs. Most microcontrollers that have integrated Ethernet controllers have a minimum frequency of 66 MHz, so that issue is avoided.

The microcontroller additionally needs to communicate with the WiFi module to be integrated into or with the PCB (if WiFi is chosen). Most modules support SPI and I²C, so any MCU that can meet the Ethernet controller, RAM, frequency, and interface requirements will do.

Aside from technical requirements, the only factors left to consider are the recency, availability, and price of the microcontrollers. Additionally, the form factor is an element to consider since the board design will have to be prototyped, most likely on a breadboard. With this in mind, a DIP package MCU would make the most sense for prototyping, with the potential to pivot to more traditional form factors for the final PCB design. However, there are currently no existing MCUs in the DIP form factor that also has an integrated Ethernet controller, so our hand is forced in that area. That leaves us with standard form factors like SOP (Small Outline Package) and QFP (Quad Flat Package) which are both surface-mount packages.

5.3.1 Atmel (Microchip)

The selected Atmel processors date themselves with the name of Atmel still attached to the product. Atmel was bought by Microchip in 2016 but has continued releasing their product lines, only under the new Microchip name.

5.3.1.1 ATSAM4E8CA-AN

The ATSAM4E8CA-AN is an older microcontroller released in 2016 that fulfills all of the requirements stated earlier. It has a 32-bit bus width and is built on the ARM Cortex M4 core. The MCU has 512 KB of flash memory, 128 KB of SRAM, and a 120 MHz clock. The form factor is the low-profile quad flat package (LQFP). It currently is very cheap, \$2.04 per MCU. Despite all these positives, it has the one downside of being a 100-pin chip. This is an excess of pins that we do not need for our design and could create unnecessary complications in the PCB design phase.

5.3.1.2 AT32UC3A1128-AUT

The AT32UC3A1128-AUT is one of the older microcontrollers, having been released in 2012. It has similar specifications to the above MCU except for a reduced frequency of 66 MHz, 128 KB of flash memory, 32 KB of SRAM, and the AVR architecture. The form factor is the thin quad flat pack (TQFP). With this MCU also being 100-pin and more expensive at \$8.39 per chip, this MCU quickly became the first choice to ditch. Additionally, its age would be a source of liability when attempting to flash it or perform other operations on it that would require new vendor tools that may no longer support the MCU.

5.3.2 Microchip

The following are Microchip's own microcontrollers, one of which is much newer and has a far better chance of being supported in Microchip's ecosystem of vendor tools. Of all microcontroller producers/designers, Atmel and Microchip were the only two companies to consistently produce MCUs with integrated Ethernet controllers.

5.3.2.1 ATSAME70J19A-AN

The ATSAME70J19A-AN is a continuation of the shared line with the ATSAM4E8CA-AN. This time, the core is an ARM Cortex-M7 on a 32-bit bus. The MCU has 512 KB of flash memory the 256 KB of multi-port SRAM and runs at a max frequency of 300 MHz. The form factor is LQFP. Not only does this MCU blow the others out of the water, but it has only 64 pins, making it a great candidate for the final PCB. It costs \$11.48 per chip.

5.3.2.2 PIC32MX664F064L-I/PF

The PIC32MX664F064L-I/PF was released back in 2010, making it the oldest microcontroller on this list. It is based on the MIPS32 4K processor core. It runs at a max frequency of 80 MHz and has 64 KB of flash and 32 KB of SRAM. It has 100 pins and is in the TQFP form factor. This MCU may be the worst option of the bunch because of its lackluster showing in the memory category. At \$7.51 per chip, it's not the choice to make.

5.3.3 Infineon XMC4504F100F512ACXQMA1

The XMC4504F100F512ACXQMA1 is the one microcontroller that is not connected to Microchip which also fits the desired criteria. It was released in 2017 and is based on the ARM Cortex-M4 core. It runs at a max frequency of 120 MHz and has 512 KB of flash and 128 KB of SRAM. Its form factor is LQFP and has 100-pins. The cost is \$11.27 per chip. The Infineon MCU was a decent option, but the reality that it matches or underperforms the Microchip ATSAME70J19A-AN in every major category makes it a secondary option. The major problem is the 100-pin layout of the chip, which could be a problem when designing the PCB.

5.3.4 Comparison Chart

Table 8 shows the criteria that we used to compare the microcontrollers.

Table 8: Microcontroller Comparison

| Part Number | Atmel ATSAM4 E8CA-AN | Atmel AT32UC3A 1128-AUT | Microchip ATSAME70 J19A-AN | Microchip PIC32MX66 4F064L-I/PF | Infineon XMC4504F 100F512A CXQMA1 |
|--------------------------------|----------------------------|-------------------------------|----------------------------------|---------------------------------------|--|
| Release Year | 2016 | 2012 | 2021 | 2010 | 2017 |
| Unit Price | \$2.04 | \$8.39 | \$11.48 | \$7.51 | \$11.27 |
| Form Factor | LQFP | TQFP | LQFP | TQFP | LQFP |
| Pins | 100 | 100 | 64 | 100 | 100 |
| Flash Size (KB) | 512 | 128 | 512 | 64 | 512 |
| SRAM (KB) | 128 | 32 | 256 | 32 | 128 |
| Frequency (kHz) | 120 | 66 | 300 | 80 | 120 |
| SPI/I²C/USB | Yes | Yes | Yes (No SPI) | Yes | Yes |
| Ethernet Controller | Yes | Yes | Yes | Yes | Yes |
| LCD Controller | Yes | Yes | Yes | No | No |

5.3.5 Microcontroller Final Selection

Putting together the chart makes the decision as obvious as possible. With no power constraints on the project, we can select the most powerful microcontroller that fulfills our purposes. That would obviously be the Microchip ATSAME70J19A-AN. It may not be the cheapest, but for only a couple of dollars more than the competitors, it will far outperform in speed, total flash, SRAM, and being the better form factor with a smaller pinout. It is the best fit for our needs.

5.4 LEDs

In the grand scheme of the garage parking enhancements, the LEDs will play the role of being the immediate indicator for drivers of where to find open spots. The LEDs will purely be indicating where open parking spots are, nothing more. The microcontroller will receive data from the local server and then drive the LED to display whatever data was indicated by the server. That could be an arrow or a number indicating total open spots in that region. There are a number of types of displays that could get the job done, it is purely up to how much we want to customize the look and feel of what is displayed.

5.4.1 LED Options

The first option is a seven-segment display. These displays are very popular and straightforward to interface with. They can display numerical values as well as some letters, but they are pretty limited on that front. If one wants to display arrows or sentences, they are out of luck. Another problem is that most seven segment displays for sale are far too small to work for our use case. In order to have the display large enough, we would most likely have to manufacture our own display. Manufacturing a display would not be worth our effort since it takes focus away from where we are trying to truly innovate on our project.

The second option is to use a dot-matrix LED display. This would give the ability to have far more precise images displayed like arrows, full sentences, patterns, etc. The main trade-off is that because there is a higher density of LEDs, more pins are needed to drive the display. However, there is a workaround to limit the increase in pins from the microcontroller driving the display. If, in addition to multiplexing, shift registers are used, single pins can be stretched to control many more LEDs. This limits control of the LEDs and disallows the use of Pulse Width Modulation (PWM), but it offers far greater flexibility by freeing up the MCU's resources.

The third option is to use a Liquid Crystal Display (LCD). LCDs are popular for many applications. They provide a high-definition image and display vibrant colors. But an LCD can be thought of as a continuation of the philosophy behind a dot matrix LED display. The LCD has an even higher resolution and, generally, a smaller form factor to go along with it. This results in LCDs often being the most demanding display to drive of the three options listed. LCDs can be great in their specific applications, but when trying to display a large image powered by a microcontroller, a different display works better.

5.4.2 LED Selection

Since seven segment displays lack the ability to show detail and LCDs are too demanding, the dot matrix LED display is what made the most sense for our project. It can display arrows directing drivers where to go and even show letters and numbers in combination to potentially indicate specific open parking spots. There were multiple options for which matrix LED display to choose from, but SparkFun had the best offerings for what we needed in the 32 x 64 dot matrix LED display, as shown in Figure 13. Their displays could do better than directly addressing pixels through just multiplexing by the use of shift registers to help address the different colors of the LEDs (because they are three color

RGB). This way, only 16 pins are needed to drive the LED display, rather than the 96 that would normally be needed.

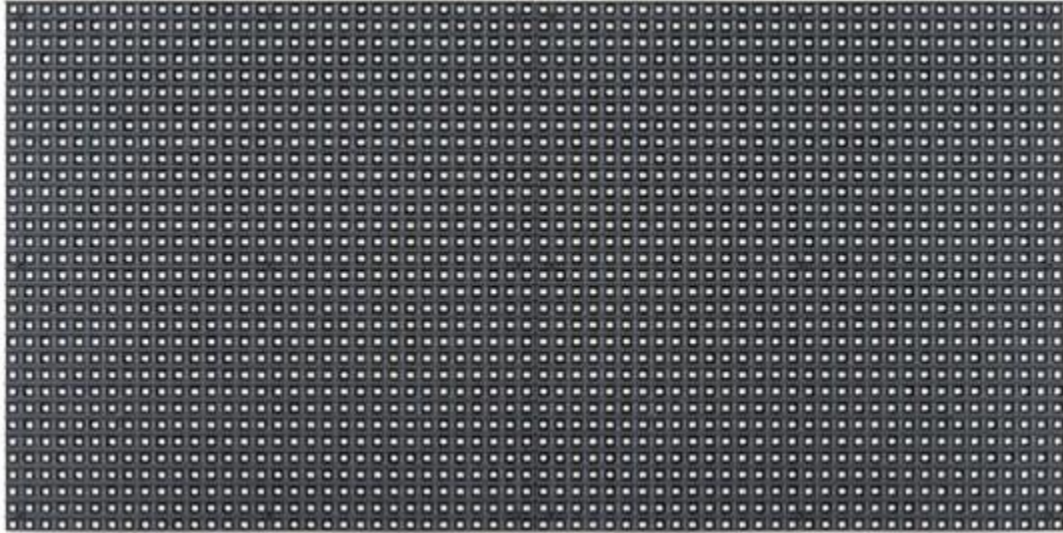


Figure 13: RGB LED Matrix Panel - 32x64

Another advantage of this display is that it can be daisy-chained with another to form a bigger LED. Or, through some clever programming, the daisy chain can be treated as separate displays so that different messages can be displayed concurrently. Either way, this display will provide a lot of flexibility for how we choose to communicate where drivers should look for open parking spots.

5.5 Local Server

The parking system will rely on a central processing computer (local server) for communication between the cameras, LED displays, and the web server supporting mobile and web applications. This local server will receive the number of available spaces from the cameras, update its local database, and send the open spaces data to the display's microcontroller and the cloud database server. The server will also operate as the gateway providing internet access to the parking system. One of the goals in choosing an appropriate server is to make sure the server is capable of the parking system of an entire parking garage, such as Parking Garage C at UCF. Another goal is to choose a computer to perform as a server with all the components built-in; this feature helps in our cost-effective goals of the project. The technical term for a computer that meets this description is called a Single Board Computer (SBC), a complete computer where the microprocessors, memory, and input/output functions are built-in on a single circuit board. It normally does not contain expansion slots for additional peripherals, and the amount of RAM is usually predetermined [30]. One feature that is not required for the normal function of the server but is needed during development and during a demonstration is a display port. Therefore, the server or computer board that will be used needs to have a display port.

In this research, three boards were selected to be analyzed and discussed that could potentially meet the parking system's needs. These three boards are discussed in the following three sections.

5.5.1 Odyssey X86J4125864

Edge computing devices are playing an increasingly important role in the field of IoT devices. ODYSSEY is a series of Single Board Computers for edge computing applications. This mini PC comes with an Intel processor and an ARM microcontroller on the same board. Its big built-in heat sink covers the whole bottom part of the board, and in conjunction with the CPU fan, both can efficiently dissipate the processor's heat and ensures that the system stays well within its operation temperatures despite the Florida Summer heat. The board supports Windows 10 or Linux. This board presents a coprocessor as an additional feature, the Microchip ATSAM21G18 microcontroller based on the ARM Cortex M0+; however, at this moment, there is no particular use for it, and therefore, it would remain unused if this board is picked. In addition, the board offers an incredible amount of computational power; at a 2.0 GHz base CPU frequency with bursts of 2.7GHz is more than capable of processing the data input from the video cameras, space availability calculations, data outputs to the LED displays, and data output to the web server in the cloud. The Intel processor comes with a built-in graphics processing unit, and therefore it supports a display. One downside of this board is the price; at a cost of \$238, it is somewhat pricey and will probably play a significant role in the decision stages of the project. The specifications for this model are summarized in Table 9.

Table 9: Specifications for Odyssey Mini PC

| Specifications | Description |
|-------------------------|---|
| Name and Model | Odyssey X86J4125864 by Seeed |
| CPU | 2.0-2.7GHz 64-bit quad-core Intel Celeron J4125 |
| Coprocessor | Microchip ATSAMD21G18 32-bit ARM Cortex M0+ onboard |
| Graphics | Intel UHD Graphics 600 (Frequency: 250 – 750MHz) |
| Memory RAM | LPDDR4 8GB |
| Wireless | Dual Band Wi-Fi 802.11 a/b/g/n/ac @ 2.4 and 5 GHz HT160 and Bluetooth BLE 5.0 |
| Networking | Intel I211AT PCIe Dual Gigabit LAN |
| Storage | 64GB eMMC |
| USB | 2.0 Type-A x2, USB 3.1 Type-A x1, USB 3.1 Type-c x1 |
| Video Interfaces | HDMI 2.0a, DP1.2a |
| Expansion Slots | M.2 (Key B, 2242/2280), SATA III, M.2 (Key M, 2242/2280), PCIe 2.0 x4, Micro SD card Socket; SIM card Socket. |
| Power | DC Jack 5.5/2.1mm or Type-c PD; DC Jack input: 12-19V; Type-C input: 15V DC |
| Dimensions | 110x100mm (4.3x4.3 inches) |
| OS | Windows 10 and Linux |
| Price | \$238 |

5.5.2 Raspberry Pi 4

Raspberry Pi is a series of single-board computers made by the Raspberry Pi Foundation, a UK charity. It is a tiny and affordable computer that can be used to learn to program and build hardware projects, automation, edge computing, and industrial applications. It can run Linux and provides general-purpose input and output (GPIOs) pins that allow controlling electronic components such as sensors. The specific model discussed in the section is the Raspberry Pi 4 Model B. A summary of specifications can be found in Table 10.

Table 10: Raspberry Pi 4 Specifications

| Specifications | Description |
|-------------------------|---|
| Name and Model | Raspberry Pi 4 Model B |
| CPU | Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz |
| Memory RAM | 1GB, 2GB, 4GB, or 6GB LPDDR4 SDRAM (Depending on the model) |
| Wireless | 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE |
| Networking | Gigabit Ethernet |
| Storage | Micro-SD card slot for loading operating system and data storage |
| USB | 2 USB 3.0 ports and 2 USB 2.0 ports |
| Video Interfaces | 2 micro-HDMI ports (supports 4K), 2-lane MIPI DSI display port, 2-lane MIPI CSI camera port. |
| Expansion Slots | Raspberry Pi standard 40-pin GPIO header. |
| Power | 5V DC via USB-C connector (minimum 3A) 5V DC via GPIO header (minimum 3A) Power over Ethernet (PoE) enabled (requires separate PoE HAT) |
| Dimensions | 3.37 x2.22 inches (56.5 x 85.6 mm) |
| Others | H.265 and H264 decode/encode. OpenGL ES 3.1, Vulkan 1.0 |
| Price | 1GB \$30, 2GB \$45, 4GB \$55, \$8GB \$75 |

5.5.3 UDOO X86 II

UDOO is a family of Open Source Arduino-powered Mini PC compatible with Windows, Android, and any Linux Distro. The UDOO X86 II series is a powerful x86 maker board

with an Arduino Leonardo-compatible board embedded on the same board. The specifications of this board are shown in Table 11.

Table 11: Specifications for UDOO X86 II

| Specifications | Description |
|-------------------------|--|
| Name and Model | UDOO X86 II Advance Plus |
| CPU | Intel Celeron N3160 @ 1.60-2.24GHz, Quad-core |
| Coprocessor | Atmega32U microcontroller. |
| Graphics | Intel HD Graphics 400 |
| Memory RAM | 4GB DDR3L Dual Channel |
| Wireless | M.2 Key E slot for optional Wireless Modules |
| Networking | Gigabit Ethernet connector |
| Storage | 32GB eMMC storage |
| USB | 3x USB 3.0 Type-A ports |
| Video Interfaces | 1xHDMI and 2x miniDP++ ports |
| Expansion Slots | SATA, M.2 Key B SSD slot, micro SD card slot. |
| Power | DC in Jack 12V 3A |
| Dimensions | 4.72 inches x 3.35 inches (120 mm x 85 mm) |
| Others | HW Video decode H.265/HEVC, H264, MPEG2, MVC, VC-1, WMV9, JPEG, VP8; HW Video encode: H.264, MVC, JPEG 1 x UART, 1 x I2C, 1 x SPI 23 digital I/O (PWM) 12 analog inputs |
| OS Support | Windows 10, Linux, Android |

| | |
|--------------|-------|
| Price | \$214 |
|--------------|-------|

5.5.4 Local Server Choice

All three boards are capable of being a suitable computer to work as the control unit. The most powerful one, the Odyssey board, comes with all the components needed to install Linux and start developing, except for the power supply. In contrast, the raspberry pi not only does not come with a power supply, but it does not provide onboard storage. Nonetheless, even after adding the price of an SD card to the cost of the raspberry pi, it costs less overall than the Odyssey or the ODDO boards. Since the parking project demonstration will have a couple of cameras and a couple of LED signs, then any of the three boards can do the job. However, the availability of the boards will play an important role when selecting the computer. At this moment, the raspberry pi 4 is out of stock with no option to backorder it. The ODDO board is also not available, and their website does not inform when it will be available. Although the Odyssey board is currently out of stock, their website is allowing backorders, with an estimated shipping date of May 7th of 2022 at the latest. Although the team decided that the raspberry pi is the right choice to maintain a cost-effective approach, it will ultimately be the availability of the boards the deciding factor.

5.6 Ethernet PoE Switch and Local Network Internet Access

5.6.1 PoE Switches

Power over Ethernet (PoE) is a technology that describes the transmission of electricity along with data in a single ethernet cable. It is used to deploy powered devices to areas where electricity is not readily available or to minimize the expense of installing additional electric wires and outlets. Therefore, a PoE switch is a regular switch with many ethernet ports that provide power and data to devices. A parking system that entitles the entire garage will require many cameras; therefore, a large PoE switch would be required. However, a small 5-port switch will be enough for the project's proof of concept. The chosen camera in the research section of this document requires PoE 802.3af; this version of PoE provides 12.95 watts and a maximum current of 350mA. It is easy to match the PoE requirements of the camera to a PoE switch because as long as the switch supports PoE 802.3af or above, it will work.

There are several small switches readily available with the desired PoE feature. For example, NETGEAR carries a 5-port PoE switch, model GS305P-100NAS, from which four ports provide PoE+ (PoE = 802.3af and PoE+ = 802.3at, the next PoE generation that almost doubles the wattage). PoE+ devices are backward compatible with PoE devices; therefore, this gigabit switch will work with the camera, and its cost is \$49.99. TP-Link carries a suitable switch for \$44.99; model TP-Link TL-SG1005P V2 includes four PoE ports. Another switch, model GPOE204 made by STEAMEMO, is a 5-port gigabit switch for \$33.99 that offers four PoE+ ports. It comes with additional features such as VLAN and 1 SPI uplink port; however, these features are not relevant to the parking system and will not be discussed. A summary of these switches can be found in Table 12.

Table 12: Summary of PoE Switches

| Brand | Model | No. of PoE ports | Price |
|----------|---------------|------------------|---------|
| Netgear | GS305P-100NAS | 4 PoE+ ports | \$49.99 |
| TP-Link | TL-SG1005P V2 | 4 PoE ports | \$44.99 |
| STEAMEMO | GPOE204 | 4 PoE+ ports | \$33.99 |

5.6.1.1 PoE Switch Choice

The switch the team decided on was the STEAMEMO, model GPOE204. It offers enough ports for the parking system, so it is an excellent option if the team adds more than one camera and LED display. Also, with a price tag of \$33.99, this model is the more cost-efficient option of them all.

5.6.2 Local Network Internet Access

The local network needs internet access for the parking system to communicate with the database in the cloud that will support the mobile and web applications. Several options exist to provide internet access to the local network, such as using a wireless router to connect to UCF's network, a cellular modem, or a personal cell phone's hotspot feature.

5.6.2.1 Cellular Modem

Cellular modems are devices that add 2G/3G/4G cellular connectivity to laptops, desktops, and tablets; they come as USB dongles, PCI or PCIe express cards, or standalone modems [22]. USB dongles connect to a USB port and are usually available for Windows and Linux. For example, Hologram.io offers a 2G/3G Nova Global Cellular Modem for \$83, \$5 for the SIM card, and a monthly cost of \$0.70 plus \$0.08 per MB of data used [23][24]. This is a good option because it works on Windows, Linux, and Raspberry Pi single-board computers.

The PCI, PCIe, or M.2 cellular modems connect to an available slot on the motherboard of choice, and after installing the drivers and a SIM card, the server can connect to the internet and share its internet access with the local network via ethernet. For example, Quectel, a global IoT solutions provider [25], offers the EM06-A LET Cat 6 cellular module for \$49.50, compatible with the single-board computer model Odyssey X86J4125864, the server board chosen in section 5.5. This module plugs into the M.2 slot on the board, and it needs an antenna and a SIM card to function. The SIM card can be the same provided by Hologram.io using the same monthly plan mentioned above; an antenna for this modem is available for about \$2 to \$4 each.

Standalone modems are also called hot spots and are sold by many cellphone companies; these devices connect to a cellular network and share their internet access via a Wi-Fi network to which several WiFi-enabled devices can connect. For example, Verizon offers these mobile hotspot modems for \$79.99, \$199.99, \$299.99, and \$399.99 depending on the selected speed (i.e., 2G, 3G, 4G, or 5G) and a one-time activation fee of \$35 [26].

5.6.2.2 Smartphone's Hotspot and Tethering.

Most smartphones can turn into Wi-Fi hotspots; they can share their data 4G or 5G internet access via a Wi-Fi signal. The local server would have to have a Wi-Fi adapter to connect to the Wi-Fi signal. However, the server would still need to be connected to the local network via ethernet cable. Although a smartphone hotspot is not a permanent solution to a parking system, it can still work for the parking system proof of concept. As an alternative, smartphones can be connected to a computer via a USB cable to be used as a cellular modem to connect to the internet; This is called tethering.

5.6.2.3 Wireless Router

A wireless router can be used to create a local network that serves as the internet gateway by connecting to UCF's existing network. The WAN port on the router can be connected to any UCF's provided ethernet port on-site. However, after talking to UCF's IT department, it was determined that UCF's network policy does not allow wireless routers to be connected to its network for security reasons. Therefore, connecting a wireless router to UCF's network is out of the question for our parking system proof of concept. This research found an alternative wireless router that allows personal mobile device tethering capabilities to be plugged into the router to connect to the internet. The router then shares its internet via its ethernet ports and a Wi-Fi network. This Wi-Fi router model GL.iNet GL-AR750S-Ext costs \$72.90, and it comes with two-gigabit ports and supports both 2.4GHz and 5GHz Wi-Fi signals.

Table 13 is a collection of solutions and the breakdown of their pricing and monthly subscription rate for visual purposes.

Table 13: Summary of Internet Solutions

| Description | Unit Price | Monthly cost |
|---|--------------------|------------------------|
| USB 2G/3G Nova Global Cellular Modem | \$83.00 | \$0.70 + \$0.08 per MB |
| EM06-A LET Cat 6 Cellular Module | \$49.50 | \$0.70 + \$0.08 per MB |
| Verizon Mobile Standalone Hotspots (2G/3G/4G) | \$79.99 - \$399.99 | \$60.00 |

| | | |
|--|---------|-----------------------------------|
| Smartphone's hotspot or Tethering (T-Mobile) | N/A | \$0 (included in phone's plan) |
| GL.iNet FL-AR750S-Ext (Using smartphone Tethering) | \$72.90 | \$0 |

5.6.2.4 Internet Access Choice

Many choices were found that can provide internet access to the parking system. The most expensive solutions were the Verizon hotspots, but one fantastic feature these units have is that they are battery powered, an excellent option for testing purposes on-site where electrical outlets may not be available or close by. However, the team decided it was not a cost-effective solution due to the monthly commitment of \$60 for a data plan. The following considered units were the cellular modules. Each unit's price was not that bad; however, the EM06-A module is the cheaper option, and since both monthly commitments are the same, the EM06-a module would be a clear choice among the two.

On the other hand, the tethering option of smartphones connected directly to a USB port on the local server is the most cost-effective option since any team member's smartphone can be used at no extra cost. Fortunately, both Windows and Ubuntu operating systems support this scenario and can share this internet access to the network via their ethernet port. However, to have a backup Wi-Fi network in case there is difficulty connecting either the camera or the LED displays to the local server via ethernet cable, the Wi-Fi router with the tethering feature was chosen instead. The team will not incur any monthly fees associated with cellular modem data plans with this tethering option. Since the proof of concept is a small scale of the entire parking system, the team feels this option would be the most appropriate.

5.7 Web Application Research

Some form of web-outlet would be our main way of communication with the consumers/users. One of the main aspects of any technological project has come down to the UI/UX aspect of the application. The complete and robust implementation of WebApp not only makes the users want to utilize the app more but also aids in increasing the digital footprints of the App. We intend on having a user-friendly, highly accessible, completely responsive WebApp to support our SmartParking system and cater to our users.

5.7.1 Web Application Types

With the increasing and ever-changing demand for various forms of technologies, the need and necessities of web applications have also changed with the tide of time. Depending on the business needs and consumer requirements, there have been implementations of nine different web application formats.

5.7.1.1 Static Web Applications

The most primitive and original form of web application is the static web application, and it implies exactly what it suggests. These forms of websites are static, meaning there is no communication module between the server and the user. Most of these static websites are effortlessly simple with the only implementation of basic HTML and CSS language. There could also be GIFs, videos, or animated banners included on these websites to attract users. With mobile applications, these web application types do not integrate well due to the need to send and receive huge amounts of data, which often leads to poor performance of the application. Mostly for the Web-Based Portfolio of an individual or a corporation where no back-end interaction is needed, these types of applications are utilized.

5.7.1.2 Dynamic Web Applications

One of the most popular forms of web applications, the Dynamic Web App allows communication in real-time. It allows the generation of data upon the user's request and backend server response. They allow direct connections and interactions on the client-side of the web app. They are significantly more complex, technology-wise, than static web apps and have various interactive components for full functionality. Databases are used to store the data that need to be shown on the web app. Administrations and organization boards are utilized for the admins to alter and improve the contents within the application, mainly for the frontend and backend parts. A lot of different languages are utilized to implement such web applications. An example of such an application would be Netflix. Depending on what the user wants and searches, the web app returns the precise content or some extremely similar items if the exact content can not be found. This process requires real-time computation and analysis. This application allows users the ability to read, write, refresh and even delete data.

5.7.1.3 Single Page Web Applications

These apps do not need any page reloading as it reflects exactly what their title says. These websites allow efficient communication between the clients and websites with minimum discrepancy. Responses from the back end of the website and requirements are more agile due to the low quantity of data. The logic implementations are usually done on the browser rather than taking place in the backend server. The clients can interact with the totality of the website characteristics within a single page. However, SEO guidelines are not supported for the SPA websites due to their Universal URL nature. A lot of popular social media platforms are currently utilizing SPA websites for their quick response times and logic implementation on browser qualities. Websites like Google, Twitter, Google Maps, and Gmail - all are forms of single-page web applications.

5.7.1.4 Multi-Page Web Applications

These applications are more complex and offer a lot more features. For these websites, there is more than one page, and each time a user would like to navigate to another page, the entire page would have to be reloaded. Whenever a customer browses different features of the website, the data from the back server fills up and reloads the page with new data. So as can be expected, the logic implementation of the website takes place on the backend side, and the requirements from the client and server are reversed. The user Interface gets

affected by this long process of creating pages on the backend and presenting them to the browser for the user. So AJAX is utilized to tackle the unexpected segment issues without reloading numerous times. Multi-page applications are widely used in the modern world, and often they also are supported by both mobile and web browsers if the front-end design of the application is made responsive. Some of the more popular languages used to implement Multi-Page Applications are HTML, CSS, JavaScript, JQuery, AJAX, etc. SEOs are better supported for these web applications since all the pages are optimized for keywords. Due to their scalability with page limitations, modern websites, portals, marketplaces, stores, and enterprise applications are heavily utilizing the MPA websites. But the only constraint for these web apps is that they are complex and hence difficult to maintain. Websites like Amazon, eBay, and Trello are some examples of MPA.

5.7.1.5 Animated Web Applications

Closely correlated with the modern Flash technology, Animated Web Applications help market or showcase content via numerous visual animation effects. Thus, SEO is not suitable for these web apps because keywords are not clear for the web pages and can not be read accurately. These web apps largely imply the usage of the creative and design aspect of web development; thus, the UI and UX engineers have the freedom of being extremely creative and utilize elephants of web design that are not supported by other web applications. This helps enrich the user experience because the users are exposed to unique designs and catchy effects along with the necessary information that the website is initially programmed to convey. CSS3, JavaScript, and SVG are some of the more popular tools for implementing such websites. Squadeasy and Miki Mottos are some of the best Animated Web Applications. However, very few companies rely on these websites to communicate with customers because of search engine optimization limitations.

5.7.1.6 E-Commerce Web Applications

E-Commerce is a big contributor to the modern-day economy. E-Commerce businesses help increase digital foot traffic to the consumers who are opposed to doing shopping in physical stores. Lately, e-commerce has gained popularity, and thus web applications that are utilized to port the digital stores onto the web have gotten more important and are being optimized to aid in sales. The web applications offer a lot of complex features to support all the functionalities of a physical store. These include electronic payment, online transactions, customer service capabilities, online browsing of store content, online shopping cart option, adding new products or removing old products, and many more. However, with all these complex and intricate features come the burden of increased complexities on the developer end and maintenance constraints. Thus usually, for these websites, dedicated developers are often hired to maintain the security, sustainability, and efficiency of the final app. Also, such websites need robust and user-friendly graphical interfaces to increase digital food traffic or scale-up. As a result, User Interface and User Experience engineers are needed to ensure all aspects of an efficient graphics interface are covered. Along with the interface for the users, developers must also include an admin interface for the website so that the online store employees can add new products. Shopify is one of the biggest E-Commerce web apps that is widely utilized.

5.7.1.7 Portal Web Applications

These applications offer a peculiar feature of having different categories of items available on the home page. Such features can be forums, chats, e-mail, search engines, browsers, blogs posts, latest content, register/sign up options, and many more. Since these applications have high feature attributes, a lot of enterprise companies often use these applications to customize and tailor the experiences of the specific targeted audience. Often portal web applications offer different views depending on regular user or admin privileges. Admin sometimes also can monitor the activities of the signed-in user to the app. It also allows the capabilities of restricting certain functionalities of the web app depending on the user or admin account. Many websites, such as Udemy and Coursera, are utilizing the portal web application format to uniquely design the content on their website to cater to the specific users.

5.7.1.8 Rich Internet Applications

These web applications are fundamentally implemented to be integrated better with the browsers and aid in resolving the restrictions implied. They utilize client-level plugins, which include Silverlight, Shockwave, and Flash. These applications support numerous desktop applications functionalities but with the extra addition of improved, more agile, and engaging communication systems. Due to these applications relying on the plugins and tools that improve efficiency and user engagement, they are better suited to improve visual user experience than the generic program applications. But complete reliance on such plugins and tools brings up constraints of unannounced vulnerabilities as well security gaps. For instance, any outdated or faulty plugins or tools will result in the websites partially being inactive or certain major components/functionalities being unavailable. Websites like Google Gears, Adobe Flash, and Microsoft Silverlight utilize Rich Internet Applications to improve the visual aspects for the catered users.

5.7.1.9 Progressive Web Applications

With similarities in visual aspects of mobile applications, these web applications are one of the most advanced forms of web development. With the increase of digital footprint from mobile devices, the importance of websites that are supported by mobile browsers is gaining more and more popularity. With Progressive Web Applications, users have the luxury of accessing all the necessary data from the web application with vast features and increased performance from any mobile browser. Such functionalities allow users to enjoy the enhanced mobile web experience as well as improve service even with slow network connections. However, the main objective for implementing such web applications was never to increase functionalities or new features within the architecture. Rather this particular web form was intended to optimize the agility and adaptability between web apps and mobile devices through poor network connections. Some of the major benefits of such applications include caching, home screen installation, efficient data transmission with HTTP, and many more. A lot of more popular websites for renowned companies are implemented with Progressive Web Application features. Companies like OLX, Starbucks, Forbes, Pinterest, and Spotify are just some of them.

5.7.1.10 Decided Web Application

The “smart” aspect of the Smart Parking Project refers to the utilization of advanced technologies to enhance the current parking experience and solve issues relating to the topic. Thus usage of a website to enhance the user experience for the targeted users is crucial to the project implementation. The project is planned to have video cameras set to record vehicle movements and presence within a certain amount of spaces and utilize machine learning algorithms to make aid in parking decisions for the intended users. Thus being able to visualize the different metrics such as open parking spots and available spots is absolutely crucial. Showcasing video camera recording in real-time would also aid users in navigating and making smart parking decisions. However, other than these features, for Smart Parking, no more complex logic implementations or real-time processing would be needed. Considering all aspects and the necessary complexities of the project, it can be stated that the utilization of a Dynamic Web Application to support server-side and user-side, real-time communications would fit in and support all the major requirements.

5.7.2 Web Development Stacks

Web development stacks mainly refer to the collection of different software set up together to implement and support the total functionality of websites. Different software is usually utilized to support different sides, ex. User side, server-side, database side, of the web applications. In this retrospective, the software is stacked to support the website as a whole. With the raging improvements in technology, the web development world has adapted many different web stacks for their specific needs and advantages. Selecting a correct stack to support the functionalities of the Smart Parking Web Application is a critical element of the project.

5.7.2.1 LAMP Stack

LAMP stack is considered the most mature web development stack. It is one of the first open-source technology stacks. LAMP stack is the compilation of Linux, Apache, MySQL, and PHP. Each of these technologies has its own purpose. MySQL is the data server for the stack with the utilization of SQL language. Apache is used as the HTTP Web Server to support the communications done throughout the websites. PHP is the backend side of the web stack. Linux is utilized as the operating system for this particular stack. This stack provides enormous performance capabilities and flexibilities reflecting the needs through customized modifications. It is the industry standard with optimized efficiency and peak performance. LAMP stacks also support FrontEnd or the user side of the web applications with low-level JavaScript, HTML, and CSS for the implementation of necessary graphics interfaces for the users. Since the operating system can be changed for any tech stack, it is easy to change the Linux-based operating system to a Microsoft Windows system which would create WAMP, or even a MAC OS enabled stack which results in a MAMP stack.

5.7.2.2 MEAN Stack

This stack is one of the newer technology stacks of the modern world. It consists of Express.JS, Angular.JS, Node.JS, and MongoDB. Express, Angular, and Node are all languages utilizing JavaScript. Thus MERN can be considered through and through JavaScript tag. This feature aids in the learning and the implementation organization. The

entire web stack utilizes JavaScript, so the structure is more optimized and ensures the simplification of the learning curve and usage, aiding in seamless integration within the components of the stack. Express.JS supports the server-side, back-end portion of the stack. Angular.JS implements the front end or user end of the web stack. Node.JS is a runtime environment, which could be considered the web application server. MongoDB is a NoSQL database to store information from the client and server-side. With the usage of a single language MEAN stack also provides the ability of code reusability across the platform to decrease redundancy. The technologies within the MEAN stack are all open source and free which enables an ample amount of resource sharing for the web developer community. JavaScript is gaining immense popularity throughout the different industries; thus, complete utilization of Javascript across both servers and client-side makes it stay with the trend as well as simple execution for the developers. MEAN stack offers scalability and flexibility in cloud hosting, and with the included web server, the web deployment is super simple as well. Different components within the stack communicate in JSON data transmission, which optimizes communication efficiency. So, the MEAN stack allows the implementation of agile and immensely efficient apps.

5.7.2.3 MERN Stack

MERN stack is extremely similar and shares a lot of the same technologies as the previously discussed MEAN stack. The new addition to the MERN stack is the implementation of the front-end user side of the web application is done with React.JS. React.JS has a lot of benefits of its own. With the utilization of the Virtual DOM embedded within, React.JS makes any changes made to the technology code - seamless. One of the advanced forms of modern JavaScript, JSX, is used within React.JS which provides harmonious component support and communication. React also allows simultaneous code usage between servers and browsers with a powerful built-in library. For a lot of off-the-shelf high-performance web applications, the MERN stack is extensively implemented. React.JS requires no templates. This allows the reduction of repetitive DOM and HTML elements. The isometric nature of React allows it to run on both the user side and the client-side of the web application. This aids in Search Engine Optimization purposes as pages can be created on the server. Thus, with the utilization of React.JS the code development is more efficient and faster. It is great at supporting applications with low-level complexities.

5.7.2.4 MEVN Stack

Very similar to the previous two discussed stacks, the MEVN stack is a combination of MongoDB, Node.JS, Express.JS. However, instead of the former Reach.JS or Angular.JS, this particular stack utilizes another JavaScript-based front-end framework called Vue.JS. The benefits of Vue.JS include a rich set of development tools and lightweight out-of-the-box functionalities, which also extends to third-party services. Vue.JS represents a framework, whereas React.JS is a library. Vue.JS also utilizes HTML instead of JSX which is used by React.JS. MEVN stack introduces the application of the newest technologies out of the other JavaScript stacks like MEAN and MERN. However, this also poses a constraint towards the MEVN stack due to the recent development of Vue.JS and not a lot of resources being developed compared to the other front-end frameworks.

5.7.2.5 Python - Django Stack

Different from the stacks utilized in the discussions above Python - Django stack uses one of the most popular and fast-growing languages Python. For the webserver Python - Django stack had the Apache for support. MySQL is used as the database for web applications. Also, due to the machine learning and data science capabilities of Python language, this stack has the potential to take web development to the next stage. Django framework is utilized for the server-side support of the web applications, which is entirely written in Python. JavaScript is utilized for the frontend aspect of the web stack. This stack also provides a rich collection of third-party packages to enrich the developing experience.

5.7.2.6 Ruby on Rails Stack

The dynamic programming language, Ruby, is used on the Ruby on rails web stack. The server-side backend aspect of the web stack utilizes Ruby, which supports default structures and database management. Sinatra, Hanami, and Padrino are some of the more popular frameworks for web development with the Ruby on Rails stack. Ruby language is open-source; thus, a large community of developers is experienced, and rich resources are available on the world wide web. A strong infrastructure along with test systems and database integration support Ruby on Rails which further enhances the developing capabilities utilizing the stack. Ruby on Rails also allows flexibility. One of the biggest fortune 500 companies, Shopify, utilizes Ruby on Rails which is also known for being able to handle almost 80000 requests per second. Websites like Github and AirBnB also implemented their website with the Ruby on Rails stack.

5.7.2.7 Decided Web Stack

Choosing the correct web stack is realistically the single most important step at the start of web development. Out of the many web stacks discussed and many more technologies that have not been put emphasis on, there are a lot of options with their own upsides and distinct downsides. With the ever-changing flow of technological advancements, web applications must correlate and support web stacks that change and keeps up with modern-day technologies. This would ensure new documents being produced, advanced resources being developed, and bigger communities supporting those technologies being formed. However, that does not mean that the older forms of technology are not efficient and should not be used. Stacks like LAMP stack and Ruby on Rails - despite being one of the earlier web development technologies, to this day are practiced and being updated. LAMP stack was developed in 1998, but websites like WordPress, Tumblr, and Wikipedia are still built on the LAMP stack. On the other hand, popular websites like Hulu, Twitter, Github, and Shopify are still utilizing and supporting Ruby on Rails technology. However, the recent technologies and the oncoming technologies like Python - Django stack and the MERN, MEAN, and MEVN stacks are being optimized religiously due to their popular usage and easy to implement capabilities. More tools are being generated to support the framework, and rich libraries are introduced for the ability to enhance the developing experience. With the promising future of accommodating the unlimited capabilities of machine learning and enhancing data analysis algorithms, Python-Django can revolutionize the web development era. However, with the same language, JavaScript, implemented on the Front-End side, Back-End, and server-side, stacks like MERN, MEAN, and MEVN are making

it simple for developers to learn and maintain and implement more robust and optimized web applications.

For the Smart Parking system, a dynamic and visually pleasing web application is needed to cater to the targeted audience. The web applications need to support real-time capabilities which would put emphasis on frequent yet fluent communication between the server-side and user end. The time frame and deadline of the project would also contribute to the web development planning since learning and implementing various technologies to support an efficient web application can pose a lot of challenges. Thus, to eradicate the time constraints for the project, choosing a stack that implements similar technologies that are easy to implement and can have redundancy across platforms between components could be one of the possible solutions. MEAN, MERN or MEVN utilization would make the prospect to be implemented in real life. All three of these web stacks implement some variant of JavaScript frameworks across the front-end, back-end, and server-side aspects of the web application. This allows code re-usage as well as simplifies the learning curve for the developing stage of web applications. However, the MEVN stack has its own constraints for being a moderately new technology. There are fewer resources for VUE.JS developed and the community for this open-source language is not yet matured as much as some of the other front-end frameworks available. MERN stack is a compilation technology that allows code reusability along with the implementation of a really popular React.JS front-end framework. React.JS uses libraries like Material - UI to improve user experience and web to mobile responsiveness. Using the UI libraries React.JS allows the development of rich user interfaces. Not only for web applications but also for future implementation of mobile app features React.JS also has an extension developed by Facebook which is a front-end framework for mobile devices called React native. The utilization of JSX implies the ability to write custom components catering to audiences. React also allows code component reusability, which allows redeployment of digital objects. For marketing purposes, React also enhances Search Engine Optimization for different keywords across the web application.

Thus considering the immense benefits of utilizing the MERN stack to deploy a Dynamic web application would be the ideal choice for the Smart Parking App. This will lessen the learning curve and lighten the workload through code reusability as well as with different library implementations to enrich the User Experience with unique graphical interface features.

5.8 Mobile Application Research

For our project, a mobile app that shows parking availability is desired. The app can potentially inform users of the number of spaces available per garage, level, and area in real-time. The two leading mobile OS platforms for which a mobile app can be developed are Apple's iOS and Google's Android.

5.8.1 Mobile Application Types

There are three major types of mobile apps to choose from Native apps, Web-Based Apps, and Hybrid apps.

5.8.1.1 Native Apps

These are apps developed for use on a particular operating system, which means the app only works for that specific device. The advantages of native apps are high customizable options, fast performance, and access to all the device features and functionalities. Among the disadvantages are the need to learn specific languages and IDE for their development, and the time it requires to build the app doubles as the same app needs to be written on each different platform. For example, creating native apps for Androids requires Java or Kotlin, and for iOS, languages such as Swift and Objective-C are used.

5.8.1.2 Web-Based Apps

Web-based apps are websites made using responsive web design in which the website reacts to the user's environment based on platform, screen size, and orientation. One advantage of this approach is that the app does not need to be installed onto the mobile device because the app is accessed through the already installed web browser. Another advantage is that the code is written once. Since the same web app can be accessed from either mobile operating system, writing apps for each platform is unnecessary. One of the disadvantages is that since web apps require a browser to run, web-based apps are slower than native apps because browsers typically do not work at the same speed as native apps. Another disadvantage is that all device functionality is not fully supported.

5.8.1.3 Hybrid Apps

Hybrid apps are a combination of the two previous types. These apps are cross-platform compatible, meaning they can be installed onto iOS and Android, but their functionality is similar to native apps. Therefore, these apps do not require a web browser. Among the advantages are that the codebase is written once (and thus takes less time to develop), and they can access the phone's features and functionalities. One of the disadvantages of hybrid apps is that since it uses web components, everything from buttons, navigations, and transitions looks and feels different. Another disadvantage is that hybrid apps do not run as fast as native apps because of the introduction of a middle layer that the app must go through to translate web functions and elements to the native device.

5.8.1.4 Mobile App Choice

After analyzing the pros and cons of each type of mobile application, the team decided to go with the Hybrid application for the following reasons. This project's mobile app does not require any of the phone's features and functionality beyond the display, touchscreen capabilities, and internet access; therefore, the app does not need to be native. Also, writing two native apps to support iOS and Android devices would take too much time. Although the web-based app allows writing the code once, it does not allow it to be installed. Therefore, the hybrid application provides the best of both worlds. It will enable writing a single codebase for both platforms and allow the app to be installed on either platform (cross-platform). Even though a web app may not have the same performance as a native app, the team believes a hybrid app will perform great for this project's objectives.

5.8.2 Cross-Platform (Android/iOS) App Development Framework Options

A cross-platform development framework is needed for this project, and there are plenty of options to choose from, such as React Native, Flutter, Xamarin, Ionic, Cordova

(formerly PhoneGap), JQuery Mobile, NativeScript, Swiftic, Corona SDK, Appcelerator Titanium, and Sencha. Due to the vast number of options, this research will focus on these four popular frameworks: Xamarin, React Native, Flutter, and Ionic.

5.8.2.1 Ionic

Ionic is an open-source software development kit (SDK) for hybrid mobile app development, and it was built on top of Apache Cordova and Angular JS. It was later rebuilt as a set of web components to allow developers to choose from different user interface components from Angular, React, or Vue JS. Besides being free and open-source, one of the main advantages is its extensive choice of user interface (UI) and quick prototyping; these ready-made elements speed up the construction of a graphic user interface (GUI) [4]. Another advantage is its documentation; with more than five million mobile apps built with Ionic, their exhaustive documentation is available on their website [5]. In addition, a strong online community of more than five million developers in constant activity on the forum makes it easier to find help. One of the disadvantages is the absence of hot reloading, which would allow changes to an app to be applied without reloading the whole app. Instead, the app refreshes the entire app to make changes active [4].

5.8.2.2 Flutter

Google created Flutter as an open and free framework to develop native Android and iOS applications with a single codebase. Unlike Ionic, which uses HTML, CSS, and Javascript, mobile apps in Flutter are written using Dart (Google's programming language) [6]. One of the advantages of Flutter is that it does hot reloading, which allows for changes to the app to be seen immediately [6]. It also has full customization and fast rendering, which permits graphics, video, and text animation without limits. Another advantage is that it has an app builder; this tool allows to write code using building blocks that can be mixed and matched to suit any needs [7]. One of the disadvantages is that Flutter is still a relatively new framework because it has not been in the market for as long as other frameworks; this translates into a lack of some advanced features and a vast resource base. Another disadvantage is learning a new programming language, Dart, to write mobile apps in Flutter. Since Dart is not a popular language, it could be a disadvantage since the online resources are not as extensive as other programming languages.

5.8.2.3 React Native

Meta Platforms, then Facebook, created React Native as an open and free framework to develop Android, Android TV, iOS, macOS, tvOS, Web, and Windows applications. React Native applications are written in React, which is coded in Javascript. One of the advantages of React Native is its countless ready-made solutions and libraries; this enhances the development process for quick prototyping [8]. Another advantage is that the developer community and the online support are large, which allows for getting help quickly. In addition, the use of Javascript opens up a vast online community that also offers support to React Native developers. One of the disadvantages is that it can be hard to debug, and developers need knowledge of the native language of each platform [8]. One more disadvantage is that 90% of the code can be applied to iOS and Android platforms;

therefore, depending on the native elements used, it may require separate optimizations for each platform.

5.8.2.4 Xamarin

Owned by Microsoft, Xamarin is a free cross-platform and open source app platform for building Android and iOS apps with .NET and C#. .NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications and Xamarin extends the .NET developer platform with more tools and libraries [10]. One of the advantages of Xamarin is its single tech stack. Due to its compatibility with the .NET framework, .NET is used to develop the app, but it is also used to develop its backend server; this advantage could potentially cut development time since the same .NET framework is used for both[9]. Another advantage is the close-to-native performance of the apps. Since Xamarin apps are compiled for native performance, the app can access all the features and functionality of the platform and device and platform-specific hardware acceleration. On the other hand, one of the disadvantages of Xamarin is that it is not suitable for heavy-graphic apps. If the app requires intense user interaction or relies a lot on appearance, then the app will require considerably more time to develop as this requires advanced knowledge of iOS and Android native technologies.

5.8.2.5 Cross-Platform App Development Framework Choice

After analyzing the pros and cons of each development framework, the team decided to go with React Native for the following reasons. The main reason is that React Native uses React JS at its core, and two of the team members have experience writing web applications in React JS; this dramatically reduces the learning curve. Although Flutter is more than capable of handling the project's mobile app requirements, learning Dart as a new programming language would considerably increase the time to develop the mobile app. Xamarin is a great choice, but since none of the team members have worked with C# or .NET, additional time would be spent learning the language. Ionic could have been chosen; however, to match the team's decision to use the MERN stack, React Native was a clear choice. In addition, since the web application will use the MERN stack, the same web server can be used to accommodate the APIs needed to serve the React Native mobile application and the Web Application.

5.9 Web Server Research

The parking garage system needs a place to store the information it is receiving from the various video cameras throughout the parking garage from which web app and mobile app clients can get their data. A database is the best solution to store this type of data, and a web server is the best place to host the web app and the mobile app files, which facilitates communication with the database.

Since the MERN stack was selected as the software development technology for this parking garage system, as discussed in the web development stack section, what is left to discuss is where to host the database and the webserver. There are two main options for hosting the backend and frontend of the app; it could be hosted in a local server or the cloud through a service provider. However, for the purpose of maintaining a cost-efficient

solution, choosing a cloud service provider would be the best option financially speaking. A local server capable of supporting hundreds of connections from students' phones could quickly escalate to thousands of dollars in upfront cost associated with buying all the parts needed to build a server, as opposed to a cloud service provider that offers hosting plans of less than a hundred dollars per month. In addition, in the case where the webserver is reaching its capacity, a cloud webserver can easily be upgraded with a few clicks.

5.9.1 PaaS vs. IaaS

Platform as a service (PaaS) and Infrastructure as a service (IaaS) are two options to choose from when selecting a service provider. The infrastructure as a service (IaaS) provides access to servers on the cloud; these physical or virtual servers are fully managed and maintained by the app developers or by IT staff. Therefore, knowledge of server operating systems, installation and maintenance, storage capacity and speeds, and CPU speeds maybe be needed when choosing the right plan. For example, among the features to choose from are storage devices such as SSD or HDD, amount of RAM such as 1GB or 8GB, Linux distribution such as Ubuntu or Fedora, and shared or dedicated CPUs. One of the benefits of IaaS is that businesses can purchase resources as needed instead of buying hardware upfront [11].

On the other hand, PaaS provides the necessary software components to deploy an app without managing the servers on where apps run [11]. The server provider takes care of the virtualization of servers, so resources are easily scaled up from the developer's point of view. Therefore, it is an excellent option for developers that do not have the server knowledge or do not want to spend time managing the servers.

For the parking garage app, either of these two options would work because, fortunately, one of the team members has IT experience and can manage an IaaS plan if chosen as a cost-effective solution. In addition, most of the team members have some Linux installation experience in virtual environments, as some of our courses required this knowledge to complete a few assignments.

5.9.2 Web Server and Database Hosting Providers

Many cloud service providers offer Web server hosting plans; some popular ones include Digital Ocean, Heroku, Microsoft Azure, Google Cloud, Amazon web services, and MongoDB Atlas. Each of these service providers is explained in the following sections, and its pros and cons are analyzed.

5.9.2.1 Digital Ocean

Digital Ocean is a simple and scalable cloud platform for all developer needs. It provides services such as infrastructure as a Service (IaaS), Cloud-Native (managed Kubernetes), and Platform as a Service (PaaS) for all computing, networking, storage, and database needs. In their IaaS plans, Digital Ocean provides one-click virtual machines called droplets. A droplet is a virtual server with a Linux-based distribution already installed and configured; this is a good option because it reduces the time for the OS installation. Their plans include a shared CPU server or a dedicated CPU server. The shared CPU is where one physical server contains many virtual servers, each hosting other people's backend,

and frontend applications; thus, one CPU is shared among many virtual servers. The dedicated CPU plan means that a physical server is dedicated to hosting only one backend/frontend application. The shared CPU plan is their more cost-effective option, and they range from \$5 per month to \$96 per month, while the dedicated CPU cost ranges from \$60 per month to over \$2000 per month. Digital Ocean offers database hosting that includes MongoDB (part of the MERN stack) starting at \$15 per month, but it also offers the option to connect to a MongoDB database hosted somewhere else.

In their PaaS plans, Digital Ocean offers a basic plan for \$5/month and a professional plan for \$12/month. The basic plan is recommended for prototyping only, and the professional plan for the production stage.

5.9.2.2 Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages such as Node.js, Ruby, PHP, and more, allowing companies to build, deliver, monitor, and scale apps. Apps in Heroku run inside smart containers, called dynos, in a fully managed runtime environment; Heroku handles these containers, so everything from configuration, orchestration, load balancing, failovers, logging, security, and more, is handled by Heroku. Therefore, app developers can entirely focus on developing the app frontend and backend. One advantage of Heroku is that it has GitHub integration, which means that a repository can be set up to auto-deploy with every push to a branch, effectively reducing the deployment time of every new code change. Heroku offers a free plan to try Heroku with no commitment; in this plan, the container sleeps after 30 minutes of inactivity. The Hobby plan for small non-commercial apps costs \$7 per month and includes an always-on container, a free secure socket layer (SSL), and automated certificate management. The standard plans range from \$25 to \$50 per month, including threshold alerting. The Performance plan for high traffic applications ranges from \$250 to \$500 a month, including deploying the app in different regions globally. Although Heroku offers database hosting, they do not offer MongoDB hosting; instead, they make it easy to connect the app to an existing MongoDB hosted somewhere else.

5.9.2.3 MongoDB Atlas

MongoDB Atlas, a database-as-a-service (DBaaS), is a cloud computing service. The database is stored in AWS, Microsoft Azure, or Google cloud platform, but Atlas provides the management tool in a centralized and convenient dashboard in a web app. Users do not have to handle the setup of hardware or software installation; the service provider handles everything related to managing the database. One of the advantages of MongoDB Atlas is that it allows accessing and manipulating the data programmatically; therefore, the data can be edited by the backend server hosted somewhere else. They have essentially main plans. Their share plan offers a free option, a \$9 per month option, and a \$25 per month one. Their dedicated plans are charged by the hour and vary depending on where the files will be hosted (i.e., AWS, Azure, or GCP), but they range from \$0.08 per hour to \$33.26 per hour.

5.9.2.4 Microsoft Azure App Services

Azure app services is a platform-as-a-service product to host web and mobile apps frontend and backend code. It works with .NET, .NET Core, Node.js, Java, Python, or PHP programming languages. It offers Azure Cosmos DB API for MongoDB, making it easy to leverage Mongo databases by pointing the application to the API for the MongoDB account's connection string. One advantage of using Azure is that it offers continuous integration and deployment where the app is automatically updated and redeployed whenever there is a change in the code repository, such as GitHub. Azure App Services offers different plans from free to isolated plans (i.e., dedicated servers). The app sits on a shared server, 1GB of storage in the free plan, but it does not support a custom domain. The next plan is the Basic; it includes a dedicated server created for development and test only. This plan supports custom domains and costs \$12.41 to \$48.91 per month. The production plan is the Standard service plan and goes from \$69.35 to \$277.40 per month.

5.9.2.5 Google Cloud Platform

Google offers two options for app deployments. One of them is Cloud Run, a managed computing platform to run containers invocable via requests or events, and it supports many programming languages such as Go, Python, Java, .Net, Node.js, and more. It is serverless, which abstracts away all infrastructure management so that developers can focus on the app itself. Cloud Run has a pay-as-you-go where it charges for CPU, about \$6.75 per month, memory, about \$6.75 per month, and request usage, about \$0.40 per million requests. It must be noted that the first 50 hours of CPU, the first 100 hours of memory usage, and the first 2 million requests are always free. In addition, new customers get \$300 in free credits to spend on Google Cloud during the first 90 days.

Another option offered by Google is the Google App Engine (GAE), a platform-as-a-service product that provides web app developers access to Google's scalable hosting in Google-managed data centers. It is similar to Cloud Run, but it is a bit easier to deploy an app in App Engine than in Cloud Run. Cloud run provides more options that can easily complicate an app deployment for the inexperienced developer. The App Engine pricing includes charges of \$0.05 to \$0.30 per hour per one container instance; there are charges of \$0.12 per gigabyte of outgoing network traffic.

5.9.2.6 Amazon Web Services (AWS)

AWS is a subsidiary of Amazon that provides on-demand cloud computing platforms. It offers PaaS and IaaS solutions suitable for the parking garage system. As part of the IaaS, Amazon Elastic Compute Cloud (Amazon EC2) offers virtual servers. Amazon offers AWS Elastic Beanstalk for the PaaS, an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on servers such as Apache, Nginx, Passenger, and IIS.

The Amazon EC2 offers plenty of different options, including the webserver but not the database. The on-demand plan price is computed by the hour or the second; for example, the most basic option starts at a \$0.052 hourly rate, which translates to \$37.44 a month if the server is used 24/7. The spot instance plan, which allows the use of spare computing capacity for up to 90% off the On-Demand price, was designed for flexible start and end

times applications. Unfortunately, it will not work for this project since the app must be available 24/7. The savings plans, which allow discounts based on a 1 to 3-year commitment, will not work for this project because the service will be used for a few months rather than a few years. The dedicated plans offer a physical server only for an app, which means no other apps are running on the same server (not shared). The pricing varies depending on the server's performance, but it can range from \$0.45 per hour to \$67.00 per hour.

The AWS Elastic Beanstalk has no additional charge for its usage. The cost is associated with the AWS resources needed to store and run the application. For example, the cost of Amazon EC2 instances or the cost of S3 buckets for storage used for the app would be the only cost for the hosting of the webserver. Therefore, AWS Elastic Beanstalk is a simple way to get web applications up and running on AWS.

AWS services offer one more option: the free tier designed to gain free, hands-on experience with the AWS platform, products, and services. It includes 750 hours per month of Amazon EC2 for 12 months, among many more perks. The 750 hours is equivalent to 31.25 days per month, so an EC2 instance can be left running 24/7 for a year before incurring any cost.

5.9.2.7 Web Server and Database Hosting Provider Summary

A summary of the cost associated with the analyzed service providers that are relevant to this project is provided in table 14 below.

Table 14: Web Server and Database hosting plan and Pricing

| Provider | Web server hosting (Monthly) | | Database Hosting (Monthly) | Total cost (Monthly) |
|--------------------------------|------------------------------|-----------------------------------|----------------------------|----------------------|
| | | | | |
| DigitalOcean (IaaS) | Basic: \$5 | Next: \$10 | n/a | \$5 to \$10 |
| DigitalOcean (PaaS) | Basic: \$5 | Pro: \$12 | n/a | \$20 to \$27 |
| DigitalOcean Database | n/a | n/a | Basic: \$15 | \$15 |
| Heroku (PaaS) | Basic plan: \$0 to \$7 | Standard: \$25 to \$50 | --- | \$0 - \$50 |
| MongoDB Atlas | n/a | n/a | Shared plan \$0 to \$25 | \$0 to \$25 |
| Amazon EC2 (IaaS) | Free tier: \$0 | On-Demand: \$37 to \$158 | n/a | \$37-\$158 |
| Google Cloud Run (PaaS) | Free tier: \$0 | After free tier: starting at \$15 | n/a | \$0 - \$15 |

| | | | | |
|--|----------------|---------------------------|-----|--------------|
| Microsoft Azure App Services (PaaS) | Free tier: \$0 | Standard \$69 to \$277 | n/a | \$0 to \$277 |
|--|----------------|---------------------------|-----|--------------|

5.9.2.8 Web Server Hosting Provider Selection

After a team discussion about all the analyzed providers, the team arrived at the following conclusion. Microsoft Azure App Services, Google Cloud Platform, and Amazon Web Services seem somewhat more complicated to deploy apps. Due to their excellent scalability and vast additional options would be great for app deployment that requires a more robust solution that can be spread out in multiple regions and guarantee 99.999% uptime reliability. Although all three options offer free tiers, their pricing seems more complicated. In addition, none of the team members have experience deploying or using any of the three services; this can increase the learning curve to deploy the web and mobile app. Digital Ocean seems to be a great option; their one-click deployment of virtual servers with the Linux-based OS already installed makes app deployment more simple. In addition, one of the team members has previous experience deploying this kind of virtual server using Digital Ocean droplets.

On the other hand, Heroku provides an excellent, easy-to-use app deployment that includes the automatic update and redeployment of the app with every push to the app's code repository. Also, the free tier or the \$7 per month cost makes it straightforward. Some team members also have experience using Heroku in previous university projects, which reduces the learning curve time. For these reasons, the team decided to use the Heroku platform to host the frontend and backend of the parking app and MongoDB Atlas as the database hosting provider.

6.0 Related Standards

Standards are documented to provide consistency and compatibility for products and efficient production processes as well as for safety for consumers. Such standards act as building ground or base guidelines for product development. But with these advantages, standards can also limit development options. These standards could be implemented by the government or any other companies that produce the initial products. Following certain standards often times are mandatory, and there are also soft standards that are optional. However, considering all the standards in the development process can ensure safety in the developmental process.

6.1 OSHA Standards

Occupational Safety and Health Administration is in charge of creating workplace standards for development safety. They document procedures for workplace safety from construction industries to electrical regulations. In the smart parking project, there would be advanced work in the mounting and installation process as well as electrical work for the PCB boards and the main control unit. Following the OSHA standards would ensure safety and decrease the risk of development accidents. Some of the instructions to follow are:

- Inspect the electrical equipment before use and locate equipment that is not in the most optimum condition.
- Make sure all the electrical equipment is connected with a ground wire before they are being used.
- Avoid wet areas and keep the electrical equipment away from any liquid spill.
- All the exposed metal parts need to be grounded before operation.

Exposure to current flows directly could have mild to severe reactions while in contact. This depends on the duration of contact and the electrical flow. Electric flows up to 5 milliamps can have a faint reaction in the human body. Up to 30 milliamps of electrical flow contact can cause painful shock in the human body. Coming directly in contact with up to 150 milliamps of electricity can cause severe pain and sometimes, depending on the contract duration, can also cause respiratory arrest. From 1 amp to 4.5amps current can cause the heart to cease and serious nerve damage. Electric flows of more than 10 amps can cause cardiac arrest and, in worst cases, probably death.

Exposure to direct current flow can cause severe consequences. Thus following the standards of OSHA is crucial for the smart park project. Working with the PCB or the control unit would make the team come in contact with metal - electricity-driven devices. So following OSHA standards would allow the creation of safety barriers and working habits during the development process of the system. While handling or working with high current flows, following these standards can prevent any possibilities of major accidents during the senior design 1 and 2-time durations. It is also advised by the authors of the standards that while working with equipment accessing high voltage of current flows, to work in groups. This would enable quick access to help in case of emergencies.

Make sure the ground connections for the electrical devices are steady all the way through the end of the connection. This would help ensure the ground connection to avoid short circuit incidents.

6.2 Data Communication Standards

IEEE standards for data transmission and communication can be used as a base guideline for the smart park project while establishing a communication network between the local server and the cameras and the web application. IEEE 802.11 standards establish communication standards for WLAN architecture and specifications.

In [21] the IEEE documentation can be found, and the depiction of the standard max data transmission rate is added in the figure below,

Table 15: Data rate based on IEEE 802.11

| Key Standards | Max Rate | Spectrum (U.S.) | Year |
|---------------|----------|-----------------|------|
| 802.11 | 2 Mbps | 2.4 GHz | 1997 |
| 802.11a | 54 Mbps | 5 GHz | 1999 |
| 802.11b | 11 Mbps | 2.4 GHz | 1999 |
| 802.11g | 54 Mbps | 2.4 GHz | 2003 |

This standard documentation 2 Mbps speed with 2.4GHz ISM bands. As a replacement for a wired network, this implementation could be the best alternative. When implemented along with TCP/IP this network system works extremely well in the mid-range with the most efficient and optimized transmission speed.

6.2.1 Ethernet Standards

In order for computer systems to communicate over a network, they follow a conceptual model known as the Open Systems Interconnection (OSI). The OSI model outlines seven different layers that exist in the connection between two computer systems. These layers are commonly known as physical, data link, network, transport, session, presentation, and application.

The ethernet connection standards addressed by IEEE's 802.3 documentation define how wired ethernet pertains to two of these layers being the physical layer and the data link layer's media access control (MAC) of wired Ethernet. The physical layer specifies the types of electrical signals, signaling speeds, media and connector types, and network topologies. This layer also implements the ethernet physical layer portion of data transmission speeds, including 1000BASE-T (1000 Mbps), 100BASE-T (100 Mbps), and 10BASE-T (10 Mbps). The data link layer specifies how communications occur over the media as well as the frame structure of messages transmitted and received. In other words, this layer dictates how the bits come off the wire, and the arrangement that they will have

so data can be extracted from the bitstream. In ethernet applications, this is called media access control.

The connection between ethernet components and a media access controller can be accomplished through the media-independent interface (MII). The MII is standardized by IEEE's 802.3u and allows for different types of ethernet PHYs to be connected to any MAC. There are many variations of the MII including the Reduced MII, Gigabit MII, Reduced Gigabit MII, and the Ten Gigabit MII; however, we will be using the reduced MII as this is the only interface supported by our MAC block on our MCU. The signals used to accomplish this interface are detailed in section 8.6. In the table added (table 16), the different standards for ethernet wires have been pointed out for convenience.

All Ethernet cables tend to use the same standardized RJ45 connectors; however, the wiring configuration inside the cable can differ from cable to cable. There are two common standards for ethernet wiring called T568A and T568B as shown in the table below. There are 4 pairs of wires in each standard, and they are commonly referred to as twisted pairs. These standards are similar as the only difference between them is the green and orange pin assignments are swapped. With this said, it does not matter which of these is used in our system as long as the wiring standard is the same on both ends of the cable.

Table 16: Ethernet Wiring Standards

| | T568A Wiring | | | T568B Wiring | | |
|-----|--------------|------|--------------|--------------|------|---------------|
| Pin | Pair | Wire | Color | Pair | Wire | Color |
| 1 | 3 | 1 | White/Green | 2 | 1 | White/ Orange |
| 2 | 3 | 2 | Green | 2 | 2 | Orange |
| 3 | 2 | 1 | White/Orange | 3 | 1 | White/ Green |
| 4 | 1 | 2 | Blue | 1 | 2 | Blue |
| 5 | 1 | 1 | White/Blue | 1 | 1 | White/ Blue |
| 6 | 2 | 2 | Orange | 3 | 2 | Green |
| 7 | 4 | 1 | White/Brown | 4 | 1 | White/ Brown |

| | | | | | | |
|---|---|---|-------|---|---|-------|
| 8 | 4 | 2 | Brown | 4 | 2 | Brown |
|---|---|---|-------|---|---|-------|

The standard that defines how a single cable is able to provide both data and electric power is known as IEEE 802.3af. Any port that is designed for Power over Ethernet applications should provide up to 15.4 W of DC power (minimum 44 V DC and 350 mA). There are three techniques that are used to accomplish this, and they are shown in the figure below. The first set of blocks in the figure is known as Mode A, and this mode transports power on the same wires that are transporting data. This is made possible by applying a common voltage to each pair. Since the twisted-pair ethernet wires use different signaling, applying this voltage does not interfere with data transmission. Using this method only allows for 10BASE-T and 100BASE-T speeds; however, this will not be an issue for our system.

The second set of blocks is known as Mode B, and this mode uses the extra two twisted pairs for the purpose of transporting power, while the first two pairs are only responsible for data. This is effective as two of the twisted pairs can be treated simply as V+ and V-. The final set of blocks is known as Mode 4PPoE, and this mode uses all 4 twisted pairs to transport both power and data. This mode is not commonly used as it only applies to high-power applications. A summary of the PoE modes is shown in Figure 14.

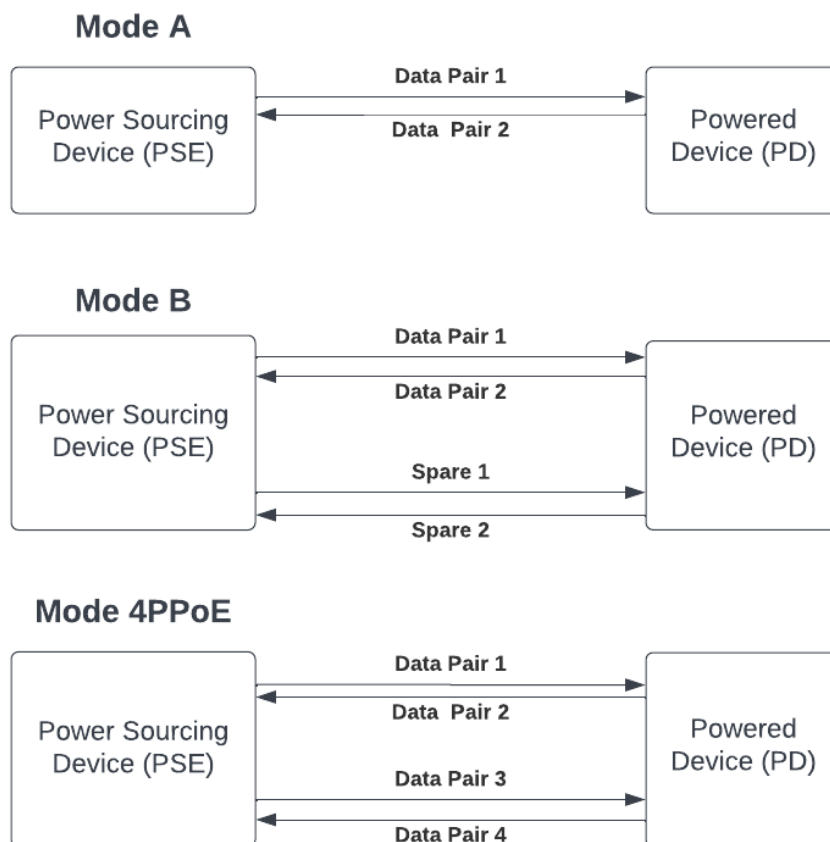


Figure 14: PoE Connection Standards

6.3 Programming Standards

Despite not affecting the actual functionality of the code or the functionality of the system in general, following programming standards can help developers stay organized. The standards include naming standards, bracing standards, formatting, and syntax standards. These standards help the codebase stay organized across platforms. This also improves the readability of the code base, which helps anyone outside the developer team to be able to glance at the code base and understand the business logic of the code. The object-oriented implementation of the code is also enhanced as these standards bring portability to the codebase. The maintainability is optimized; thus it helps the developers with the debugging process. However, an even bigger advantage is in the case of adding new features or advancements to the overall system; following a standard code system can make the integration process extremely smooth.

6.3.1 Programming Naming Standards

Naming standard is one of the biggest advantages of the programming standard process. These processes include camel case methodologies. There are three types and they are the initial uppercase, the lowercase, and the consistent case. These naming conditions simplify the naming process across the code base and define the purpose of numerous variables defined. For the understanding of certain parts of the code, consistent variable naming allows developers to keep track of multiple usages. Some examples of these naming standards are listed below,

Consistent Case - `CONSISTENT_CASE`
Upper Camel Case - `UpperCamelCase`
Lower Camel Case - `lowerCamelcase`

6.3.2 Programming Syntax Standards

Syntax standards include consistency in white spaces, comments, line length, etc. This allows for improved readability and reusability in the code. A line worth of space in between different modules within the code would increase the portability of the codebase. Between multiple business logic codes, vertical single line spaces can help developers distinguish different parts of the code easily.

Comments are one of the main indicators to communicate the purpose of the code to different developers or even to skim through the code base with speed. Commented each code block would increase general context understanding for the readers who are not part of the code developing team. However, following a specific standard while commenting is important. The standards could include the usage of a single “//” character at and the end. Also, starting each code block with a single line comment consisting of 50 characters max can allow code blocks to be structured. Avoiding excessive white spaces while commenting should also be included within the standards.

Having a set max length along the codebase allows the integration of smooth structure across. This standard allows no logic to exceed the width of normal devices. Developers will have efficiency while browsing through or debugging code with these implementations. This standard makes it possible by wrapping the line after a set amount of characters. A generalized example would be wrapping a line of code to another line after it exceeds a maximum of 100 character count.

6.3.3 Programming Indentation and Bracing Standards

Indentations and bracings are two of the most vital components of keeping the codebase neat and organized. Braces are used frequently in code statements for logic implementations. However, even though, in some cases, braces are mandatory, some “if” statements do not require braces. However, as standards, for any type of business logic or driver level logic, braces covering code blocks should be used. It helps developers keep track of the start and end of existing code blocks.

Indentation is another important concept for organization purposes. After an if-statement or any type of loops within them, the code needs to be indented by 1 tab worth of space. Every time there are embedded loops within, the spacing for indentation should be consistent and increase by multiples of 1 depending on the level of depth within the logic. Without the consistent standard implementation of indentation, the code blocks would become difficult to read.

6.4 Ingress Protection Code (IP Rating)

The Ingress Protection or IP rating code classifies the protection provided by an enclosure for electrical equipment against intrusion, dust, contact, and water [27]. IP ratings are defined in the International Electrotechnical Commission (IEC) number 60529 (also published by the European Union by CENELEC as EN60529) [27]. The two numbers that come after IP have two meanings (e.g., IP65). The first digit refers to the solid particle intrusion protection. This number is required and indicates the level of protection of persons against access to hazardous parts inside the enclosure and protection of the equipment inside the enclosure against ingress of solid foreign objects. The meaning of the different options as the first digit is summarized in Table #. The second digit is optional and indicates the level of protection against water ingress, and each level is outlined in Table 17 and Table 18. The IP rating for the parking system was chosen to be at least IP65 which provides dust-tight protection and protection against water jets.

Table 17: IP Code First Digit Meaning

| Level | Effective Against |
|-------|----------------------------|
| X | X means no data available |
| 0 | No protection |
| 1 | Large surface of the body |
| 2 | Fingers or similar objects |

| | |
|---|--------------------------------------|
| 3 | Tools, thick wires, etc. |
| 4 | Most wires, screws, large ants, etc. |
| 5 | Dust protected |
| 6 | Dust-tight |

Table 18: IP Code Second Digit Meaning.

| Level | Effective Against |
|-------|--|
| X | X means no data available |
| 0 | No protection |
| 1 | Vertically dripping water |
| 2 | Dripping water when the enclosure is tilted 15 degrees |
| 3 | Spraying water |
| 4 | Splashing of water from any direction |
| 5 | Water jets (12.5 liters/min at 4.4 psi) |
| 6 | Powerful water jets (100 liters/min at 15 psi) |
| 6K | Powerful water jets at higher pressure (75 liter/min at 150 psi) |
| 7 | Immersion (up to a meter) |
| 8 | Immersion (more than a meter) |
| 9k | Powerful high-temperature water jets |

6.5 Voltage and Power testing Standards

Safety is one of the major components to consider while working with electrical tools with higher voltage and power. Most corporate companies who work with PCB designs and microcontroller manipulation often have their own set of rules for a safe environment for working with electronics. In [31], the IEEE Std 510-1983 has focused on the guidance for working in a safe environment while working with electrical equipment. The paper then goes in-depth with the definitions and usages of rubber protective tools, safe work practices while dealing with electrical tools, field tests, special concerns working with current flow, safe electrical equipment, and many more.

As far as protective equipment goes, insulated rubber equipment can provide a lot of core safety while the machine is energized. During the process of switching equipment, one must use insulated equipment. Because the machines still may be energized at the time of the switch. Also, while the equipment is being turned off or de-energizing, it should be considered energized, and caution must be taken. National Consensus Standards NCS is mainly responsible for employing and publishing safety protocols for Rubber - Protective types of equipment. All the rubber insulated equipment must be National Consensus Standards approved. Such equipment could include rubber-insulating gloves, matting, blankets, hoods, line hoses, and sleeves.

During laboratory or research implementation work - multiple safety protocols are mentioned to be followed. Having permanent and temporary test areas, interlock or fail-safe signaling systems, grounding protocols, and power supply safety are just some of them. The main or permanent test area should be enclosed with some sort of barrier with doors or some type of blocker to restrict public access due to potential power hazards. At the same time, caution boards must be utilized to warn individuals working closely. As far as temporary testing areas go, they do not need permanent barriers. However, there should be some sort of temporary separators like portable fences. Safety tapes that are easily visible and at least waist high should enclose the parameter. Only once the equipment is turned off and fully de-energized can the tape be removed for others to enter the area.

Access to a loudspeaker will aid in communicating with large numbers of individuals at the same time. For fail-safe purposes, some sort of signal or interlock system can also be put in place. Any type of audible sound or gesture can help enhance the safety net. For the main and temporary test areas, a systematic procedure should be put in place. This will ensure that the equipment is not energized while someone else is in close proximity within. In critical cases, companies have been known to select observers to make sure the test areas are safe for other individuals to engage in.

With the implementation of workspace and workbench safety comes the concept of work equipment safety. Without the equipment being thoroughly checked for safety measures, any type of workspace area precaution would end up invalid. These equipment safety measures can include special and basic grounding practices, safety in power supplies, being cautious of any physical hazards, and many more. Before working on any type of test object, it should be a mandatory requirement to ground any type of high voltage circuit. This can save any type of personnel hazards. For extremely high voltage equipment, operators usually attach a ground to the high voltage terminal using some sort of insulating materials. All the exposed intermediate terminals with running electricity should be grounded. Also, grounding instruments should be checked for efficiency and safety, and this procedure should have more precedence over proper signal grounding.

For the direct voltage power supply, all the individuals working in close proximity need to be notified as there could be scenarios of random discharge of electricity in underground metallic objects and capacitive elements. The equipment in close proximity also should be grounded and short-circuited before initializing the direct voltage source. Also, after usage, the equipment should be allowed some time for the test voltage to decay to a low enough

value so that other individuals can gain access to the area in a safe manner. With these, caution should also be followed for airborne porcelain or any other material in case of equipment failure. Having any sort of open container should also be prohibited as it may cause physical injuries in case of spillage.

The Smart Parking system is planned to have the PCB set to be able to support the control unit and LED signs. Thus various electrical equipment will need to be utilized for the proper implementation of the system. To ensure a safe environment during the implementation and the test procedure, the safety precautions mentioned in IEEE's Std 510-1983 standard can indeed provide a multitude of guidelines. Following these strategies, a safe execution of the electrical component testing as well as application of the Smart Parking System can be achieved.

6.6 NEMA Ratings for Enclosure Standards

National Electrical Manufacturer Association (NEMA) signifies and enforces protocols on the environment where the electrical component resides (fixed enclosures) and their stability of them. The association standardizes the quality of the fixed enclosures used for the storage of electrical components and tests the ability to withstand certain environmental calamities. NEMA not only helps the end-users to achieve protection of a safety net to keep the electrical components enclosed, but they also ensure the economic aspects of the enclosing are subpar. They aid in improving communication safety and economics between the end user(purchaser) and the seller (manufacturer).

NEMA standardizes and supplies an ample collection of enclosures. IP Enclosures, IEx Enclosures, ATEX Enclosures, UL Listed Enclosures, C - UL Enclosures, and many more. The enclosures are all powder coated carbon steel, 304 stainless steel, 316 stainless steel, or the aluminum enclosures of NEMA are all supported and verified by the UL or IP ratings. There is a selection of NEMA standards that ensure all the necessary capabilities for the safety of the enclosed electrical components.

NEMA Type 1 standard ensures protection of the electrical components against solid falling dirt. The Type 2 standard enhances the safety of minor water splashing and dripping as well. Type 3 of the standard enhances the protection of the enclosures from windblown dirt, rain, and even snow. This standard specifies the safety features for the components even against the external formation of ice. The Type 3X standard allows standardizing the enclosing equipment to be sturdy enough to fight against corrosion and stay undamaged by any external formation of ice. Standard Type 4 enforces stronger safety nets by adding security to the enclosure products to keep the electrical components operable even against external hose-directed water at a strong force. Type 4X improves the Type 4 standard with the capability of working against corrosion. Type 5 standard allows protection against more fine-grained objects that may affect the components. Such objects include dust, lint, fibers, and other flyings. Type 6 standard states that the enclosures must additionally provide safety of the components from external submersion for a limited time at a limited depth. In Type 6P standard, this safety net gets improved with safety against prolonged external submersion of the enclosure.

NEMA Type 7 and the later standards focus also on the safety outside as well as inside the enclosures. Standard Type 7 states that the enclosures must contain any internal explosions without causing any external hazards. Type 8 of the standard is stated to protect the components by preventing any possibilities of combustion through the use of oil-immersed equipment. Type 9 standards prevent the possibility of combustion through airborne combustible dust. Type 12 of standards and the Type 12K of standards - both are specified to provide the safety features mentioned above however with the additional emphasis on being constructed without a knockout. Type 13, which is the latest standard of NEMA, is introduced to add an extra layer of protection for the electrical components. The standard provides safety against dirt, circulating dust, and any airborne particles like dust, lint, and fiber. As well as water dripping, splashing, and spraying. However, it adds a layer of security by adding security against oil and non-corrosive coolants as well.

NEMA standards provide safety measures and standards to shield electrical components and systems from any environmental (external) calamities as well as offer protection from external hazards from internal combustion and explosion. The Smart Park system has multiple crucial electrical components such as the POE cameras, PCB boards, and the Control Unit. These components are a vital part, in fact, the heart and soul of the system. Thus protective measures must be taken to ensure the safety of these components. Following the NEMA guidelines and standards can allow the developers to achieve safety and maintenance for the electrical components.

6.7 CPSC Standards

Consumer Product and Safety Commission is a government agency that ensures safety in product manufacturing. The commission is responsible for issuing safety standards for all sorts of merchandising products. From children's safety for walkers and cribs to safety standards for writing instruments and other materials - all are standardized by this corporation. These standards do not specifically enhance safety in the workspace or the workbench. However, these government-regulated standards act as primary guidelines to help consumers with the safe handling of all the necessary products.

Poison Prevention Packaging Act enacts upon a set of standards and regulations that tread closely with safety in packaging for the end users (buyers) from different materials. While product parts for any project are delivered at home, it is extremely important that the packaging materials are safely maneuverable. Some health constraints may come from the materials of the packaging substances. The standard paper also focuses on the safety of children from the packaging materials. With an emphasis on child-resistant packaging, this standard ensures that any of the packaging materials are manufactured, keeping consideration of physical hazards. Ensuring all the components of the electrical system being CPSC rated is an important step to ensure safety in product safety and packaging. This way, the developers can safely handle the delivered packages at home, and even consumers can get a sense of security while unboxing and utilizing the devices. Batteries are one of the more common components that the paper emphasizes upon.

CPSC also implies on electronic products being shipped packages that are approved by its 1700.15 section, which standardizes poison prevention packaging standards. Since in

batteries, low-viscosity hydrocarbons can be found, they must abide by the 1700.15 a and b packaging section regulation to protect the consumers from any type of physical hazard during or after the delivery process. The section also mentions the packaging going through Younger - Adult tests and Senior - Adult tests for scoping purposes. In the Younger-Adult test, the packaging is tested for child resistance with at least 80 percent effectiveness. Whereas in the Senior - Adult testing, the packaging goes through ease of opening methods with 90 percent senior adult panel test.

For Smart Park System, all the parts will be ordered online and thus, making sure that the products as well the parts being CSCPS PC rated is crucial. Since the product delivery will be done directly at home, the packaging safety measures need to be monitored closely for the safety of family members and individuals living in close proximity. The CPSC regulations can provide enough context in safety for product and packaging safety.

6.8 Soldering Standards

Joint Industry Standard (J-STD-001) is the current state-of-the-art industrial specification for the grouped assembly of electronics and electrical product classes. Institute of Printed Circuits regulates the usage of J-STD-001 standards for the soldering process of the manufacturing PCBs. There are a lot of training programs to instantiate the J-STD-001 soldering and manufacturing process. Released in 1992, the J-STD-001 A version was the pioneer of global soldering regulation, and now the latest standard document is at J-STD-001 H. This standard initializes and defines different materials and production processes of soldering for stability, reliability, and quality in solder joints and assembly. The standard initializes material requirements, soldering requirements, component placing guidelines, rework conditions, assembly inspection, connector details, and many more.

The general provisions of soldering of J-STD-001 are the first steps towards a proper execution of the soldering process. To prevent contamination of materials and surfaces, J-STD-001 emphasizes the cleanliness of the workbench and the tools. These inspections for cleanliness need to be done even before the conformal coating and stacking applications. It also specifies to abide by the manufacturer's rules for the environmental heating or cooling rates. The stacked and multi-layer chip capacitors are treated as thermal shock-sensitive. This allows protection against thermal excursions. The soldiers also must wet the tinned areas of the wire. This aids in the strands of the wire not being damaged. It also emphasizes the defect during the soldering process and differentiates the acceptable mistakes from the mistakes that must go through a rework process.

The proper welding demonstration is shown in Figure 15 [32]. Visually noticing the soldering processes can often aid in determining the severity of the rework needed for any soldering job. A snapshot of the article portrait can be found below,

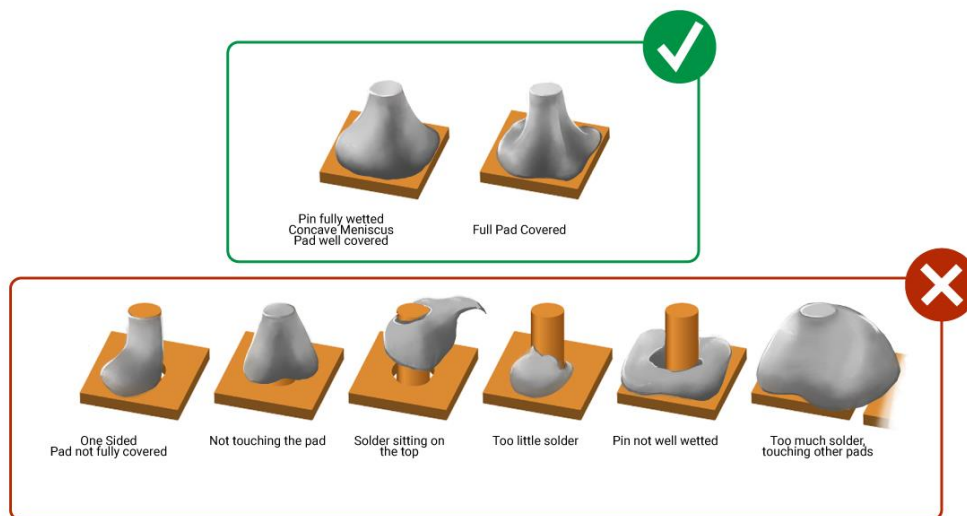


Figure 15: Acceptable vs. Rework Needed Soldering Process.

In the standards, there have been many mentions of process requirements that aid in space addendum applications. One of them is the prevention of corrosion. User-approved red plague plan is kept in consideration for silver-coated copper conductors. This aids in lessening the growth of cupric oxide and latent damage due to various environmental conditions. The main constituents of the soldering process are solder paste, cleaning media, soldering system, flux, etc. When these or any of these are changed, proper tests should be run for validation and the changes should be documented in a changelog. Sn60Pb40, Sn62Pb36Ag2, Sn63Pb37, or Sn96.3Ag3.7 are different types of soldering and are considered ideal per standard. Making sure to choose the correct alloy with good service life, performance and reliability - is important. Then the Flux considerations also emphasized these sets of standards and requirements. Decided in two, RO (rosin) and RE (resin) are two of the J-STD-001ES flux categories. These also have activity stats of L0 and L1. The compatibility of these different types of Fluxes and their activation level must be compatible with the process and thus need to be tested. This goes hand in hand with Solder Paste Testing to check the paste spreads. At the same time, solder balls caused by oxidation should also be checked.

While considering chemical strippers, tests should be run to check for degradation or damage. Chemical strippers are flux removers to clean up excess flux after the soldering process. For thermal protection, heat sinks also should be utilized. Soldering or reworking during the soldering process, the components are at risk of heat and thermal shocks. Heat sinks or thermal shunts give an extra layer of protection for the electrical components. The components also should be considered to have a higher thermal threshold to hold up against the thermal shocks and excessive heat during the soldering process. During PTH soldering connections, the solder must completely fill up the PTH. This allows the top and bottom barrels, better leads, and wetting of lands. The leads could be Flat Leads, Coined Leads, or round Leads. Figure 16 shows a comparison between what is an acceptable and not acceptable soldering




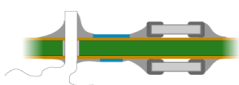
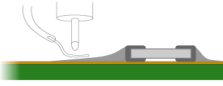
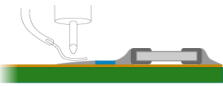
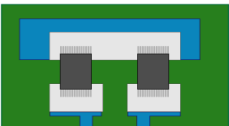
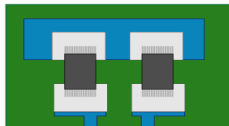
| Item | Poor Example | Recommended example /Separated by solder resist |
|----------------------------------|---|---|
| Multiple parts mount |  |  |
| Mount with leaded parts |  |  |
| Wire soldering after mounting |  |  |
| Over view |  |  |

Figure 16: Acceptable vs Not Acceptable Part Mount

In the figure above, the different mounting processes and standard mountain procedures can be witnessed. While mounting, there should be enough space between different components for cleaning purposes. An exposed base metal must not prevent a solder connection from forming. Inhibition of formations of the solder joints with OSP must not be permitted. Wire ends, and lead ends can not exceed the terminal by more than one lead worth of diameter. Leads should be formed in accordance with the system design even before assembly because reworking of leads is not recommended. Just bending angles and minor adjustments to the leads can be acceptable at best. Lead seals, welds, or connections inside components, must not be interfered with or damaged by lead forming procedures.

For cleaning, there are also mentions of ultrasonic cleaning, and emphasis also has been put on particulate assembly mattes being clean. Transmitting high-frequency sound waves through the liquid enables the Ultrasonic cleaning process. In the scenario where nothing but terminals or connectors are present on the bare boards - this is used. Caution must be abided, as ultrasonic cleaning should only be used if the manufacturer's documentation specifies that the cleaning would not affect or interfere with the electrical performance of the system assembly. Debris, solder splash, wire clips, and solder balls should be cleaned from the particulate matter assemblies.

The Smart Parking System incorporates a PCB design to support the LED and Control Unit Communications. To customize the connection on the PCB board and to enable ethernet communication, soldering must be done between different components. The universal standard of soldering - J-STD-001 - will allow standardizing the soldering process of smart parking systems. It offers guidance to follow the optimum environment built with the correct materials and placements.

7.0 Design Constraints

The conditions that limit the choices that are able to be made during the design process of a system are called constraints. Constraints force developers to consider edge cases of failure and make sure of a safe and reliable execution of the desired system. Constraints include the economic aspects of the project, environmental effects, safety standards, ethical considerations, and time limitations, just to name a few. While developing the design of our smart parking system, there were a variety of constraints that we needed to consider and adhere to. This section will discuss those constraints and address how they will impact the design of our system.

7.1 Economic Constraints

One of the biggest constraints we have needed to adhere to during the design process of our system is the total cost of all the components. Since our project was not sponsored by an external company and we are instead splitting the total cost of our system evenly between all group members, we are limited in some of the design decisions we are able to make. To counteract this, we will cut costs where we can by using components that we already have on our hands, such as ethernet cables and a PoE switch. Additionally, we have done an extensive research to ensure that the components we will be using are of high quality but at an affordable price. Lastly, before we purchase any components, we will be communicating with all suppliers to check if there is a student discount available.

It is also important to mention that this constraint plays into why we will not be designing a parking management system for an entire garage. Every single garage on UCF's campus has over 1000 spots that could be monitored, and we do not have the finances to design a system at that scale. Instead, we will be designing a system that is just a proof of concept and can be scaled to fit the needs of an entire garage. The final budget for our project can be seen in section 12.

7.2 Environmental Constraints

The parking garages at UCF are exposed to all the elements that Florida's weather presents and we need to be sure that the electrical components in our project are protected from these elements. For this reason, we have chosen to use at least an IP65-rated enclosure for all of the electrical components in our project. This will ensure that they are protected from any moisture and dust that could be present in the air. Additionally, with the high temperatures that Florida experiences during the summertime, we need to make sure we are picking components with an appropriate ambient temperature rating.

Another constraint that needs to be considered relating to the environment that our smart parking system will be functioning in has to do with the garage itself. Each level in the garage with the exception of the top level has a ceiling that could present quite a few issues. The first of these issues is the maximum height our camera is able to be placed. Ideally, we would like to place our camera at a high enough level such that the best view of all parking spaces to be monitored is achieved, and the ceiling could hinder us from doing that. The second issue is the lighting problems that the ceiling could create on lower levels. Since

there are many shaded areas throughout the lower levels of the garage, this could have a negative impact on the detection of parking spaces since the camera may not be able to clearly define the white line that marks out a parking space. The final issue that the ceilings create is the structures that could obstruct the view of parking spaces from the camera. With an obstructed view of the parking spaces, this could affect the accuracy of our smart parking system.

7.3 Social Constraints

One of the major constraints that could affect the design of our system is social constraints. With the fact that people often walk through the parking spots in the garage or use their car to take up multiple spots instead of just one, this could create some issues with the reliability and accuracy of our system. The students and faculty at UCF will likely not have the smart parking system in mind when they are navigating the garage, and we need to design our system in a way that responds accordingly. For example, we will need to place the LED sign in a location where the view of it has no chance of being obstructed by the way someone has decided to park their car. Additionally, we will need to design the computer vision system in a way that it does not see a person walking through a parking spot as being occupied and is able to conclude if a parking spot is still open or occupied if someone has not parked their car perfectly in between the lines. We will also need to think of the ideal camera placement should be placed such that they are not able to be tampered with.

7.4 Political Constraints

This is a constraint that will not have a major impact on the design of our system if any, since we are not designing our system for government use.

7.5 Ethical Constraints

With the goal of designing and building a successful smart parking system, ethical constraints are something that we must keep in mind. IEEE defines their code of ethics in [19] and emphasizes the importance of their technologies having a positive impact on the quality of life, accepting a personal obligation to their profession, their members, and the communities they serve, and committing themselves to the highest ethical and professional conduct. Therefore, this code of ethics serves as a good set of morals to follow throughout senior design one and senior design two. After reading IEEE's code of ethics, as a team, we understand that we must do the following:

- Uphold the highest standards of integrity, responsible behavior, and ethical conduct in our activities.
- Treat all group members fairly and with respect, not engage in harassment or discrimination, and avoid injuring others.
- Strive to ensure this code is upheld by all group members.

7.6 Health and Safety Constraints

For our project to be considered successful, it must be able to operate without compromising the health or safety of students and faculty navigating garages. We will be using tripods to lift our camera to its required height during our testing and presentation, however, in a real implementation of our system, the cameras and LED signs would be mounted on a ceiling or wall. The vibrations that go through the garage while people are driving through them could loosen the screws used to mount the cameras and LED signs. This creates a risk of one of these components possibly falling onto a car or person, which could potentially cause damage or injury. For this reason, if this system were to be implemented on a larger scale, we suggest that the mounts for all of the cameras and LED signs be checked routinely.

With the fact that we will be using a camera embedded with computer vision and data will be transferred through a LAN within the garage, privacy is another constraint relating to safety that must be considered. Using computer vision gives the programmer a lot of capabilities that could potentially be considered an invasion of privacy against people navigating the garage; therefore, our group pledges that we will not use these cameras for any purpose other than keeping track of open parking spots in the garage. Additionally, since the router in our design will be connected to the internet over WiFi to send data to the cloud for use by the mobile app and web app, we must be sure to follow proper protocols to protect the LAN. Infiltration of the LAN by an external source runs the risk of the cameras being used for unethical purposes which could result in an invasion of privacy.

7.7 Manufacturability Constraints

One of the major constraints that could affect the design of our system in a variety of ways is manufacturability constraints. When we are picking components to be used in our system, we always need to be aware of if that part is in stock or not. We also need to be wary of if a component to be used in our design is only sold in bulk quantities or if we are able to purchase just one. Additionally, we need to check how long it will take us to receive any component if we decide to purchase it.

This constraint should have major consideration throughout our design process as it could have a negative impact on other constraints, including our time constraints and economic constraints. If we do not verify that a company is trustworthy before deciding to purchase a component from them, we run the risk of not receiving that component by the promised time or not receiving it at all. This would cause both a waste of time and money which would be detrimental to the final design of our project. To counteract this, we will start ordering components towards the end of senior design one and over the summer such that they are already on hand at the start of senior design two. After components are ordered, we will create a table listing each of them with the expected delivery time such that we know what to expect from the suppliers.

7.8 Sustainability Constraints

We would like to design a reliable system that is able to function for 5+ years before needing any maintenance requirements; therefore, we will need to design our system in a way such that this is possible. To do this, we need to consider the environment that our components will be functioning in and be aware of the impacts it could have on our system.

One of the ways we will accomplish a reliable system is to work with PoE connections throughout our design rather than wireless connections. This will rid our system of batteries which will need to be replaced on a routine basis; therefore, not satisfying our desire of a system that can function for 5+ years with minimal maintenance requirements. In a clean environment with no exposure to elements, ethernet cables should last for at least 20 years. Since our system will be functioning in an outdoor environment, we expect they should last for about 5-10 years before needing to be replaced.

In addition to working with PoE connections, we also need to pay attention to the operating temperature of the components we choose. Florida can experience some very hot weather, upwards of 90 degrees Fahrenheit, and we need to be sure that our components are rated to handle an ambient temperature in this range and will not malfunction. We will also be enclosing all electronic components in at least IP65-rated enclosures which will protect them from both moisture and dust.

7.9 Time Constraints

One of the major constraints we need to adhere to while designing our system is the amount of time we have. We only have two semesters to design and build a functioning project, and a final product for our smart parking system needs to be delivered by the end of senior design two. To ensure we are staying on track and not falling behind, we have created two tables consisting of strict deadlines that need to be met which can be seen in section 13. The first table is for senior design one, and the first is for senior design two; and while the table for senior design two does not have any dates yet, we have still listed out the tasks that need to be completed.

With only two semesters to complete all research, design, testing, prototyping, debugging and deliverance of a final product, we are prepared to make any major design changes to ensure we meet these deadlines. For example, we would like the computer vision system to be completely autonomous and require no human input; however, if we run into too many issues during the testing process, we understand that we will need to change the design to something that does require human input, but we are sure will work properly with no complications. Additionally, the time constraints we have is another factor that plays into why we will not be designing a parking management system for an entire UCF garage and will instead be designing a system that is able to be scaled up if need be.

7.10 Testing and Presentation Constraints

One of the major constraints that will come into play during senior design two is how we will test and present our system. Since we have decided to use ethernet connections

between the components within our system rather than wireless connections, we will always need a power source to be available to us at whatever garage we are working with. To deal with this, we will be using long extension cords to give us flexibility with where we could test our system. Additionally, when we present our project, we are thinking about potentially building a mock garage to show our proof of concept. This would ease the process of showing how our system works since we will not have to set it up at an exterior garage and instead remain inside the engineering building.

An additional issue that must be considered that relates to this constraint is the fact that connecting a server to UCF's WiFi is prohibited. This would prevent us from using an access point, a WiFi router, creating a local area network, or sharing the ethernet access via ethernet port as long as we are using UCF's WiFi. We are currently in contact with UCF's IT department to see if there is another option that could be suitable for our project; however, if they do not provide any helpful information, we will have to find another way to go about this. One solution we have come up with, but have not yet tested is using the HotSpot on one of the group members' cell phones to provide us with internet access.

8.0 System Design

In previous sections, we did research on the different technologies implemented in our project, standards we need to follow while developing our design, and constraints that need to be considered. Now we will be giving a detailed overview of how the major components of our smart parking system will function. This section is separated into 5 parts being computer vision system design, LED display system design, mobile application design, web app design, and local server design. Additionally, the final section in 8.0 addresses the components we will use on our PCB.

8.1 Computer Vision System Design

This section will give a detailed discussion of both the hardware design and software design to be implemented in the computer vision system.

8.1.1 Computer Vision System Overview

The computer vision system will be required to take a live video feed of parking spaces, apply computer vision techniques to compute the number of open spaces, and then transmit this data over ethernet to the local server and the microcontroller driving the LED sign. This system will utilize the OAK-1 PoE in order to satisfy both video capture and computing requirements. This camera employs the Movidius Myriad X VPU which will allow us to use advanced computer vision techniques while also limiting the amount of data that the camera is required to transmit, as explained in section 5.2.

The OAK-1 PoE is designed to be used for PoE applications; however, it can also be powered by USB-C as long as we remove the housing holding the circuit board. This will make for effective testing and prototyping, as we will be able to work with the camera before we need to buy any of the other components in our overall system.

8.1.2 Software Tools

The table below shows the software tools that will be used to develop the computer vision system. Each of these tools is further described in sections 5.1 and 5.2.

Table 19: Software Development Tools

| Software | Description |
|--------------------|---|
| DepthAI SDK | A python package containing convenience classes and functions that will help in completing the most common tasks while using the DepthAI API. |
| DepthAI API | The API we will use to connect to, configure, and communicate with the OAK-1 PoE. Supports both Python and C++ APIs of which we will use the Python API. |
| Python | A high-level, general-purpose programming language. This is the language we will be using to complete computer vision tasks. |
| OpenCV | An open-source computer vision library containing a variety of programming functions aimed at real-time computer vision. This will be used to complete image manipulation and object detection tasks. |

8.1.3 Software Design

The computer vision system will use a combination of the software tools described above to satisfy the requirements described in section 8.1.1. Upon bootup, the system should immediately start doing the tasks to meet these requirements, which will be made possible with the PipelineManager class from the DepthAI SDK. The PipelineManager will give us the ability to create a pipeline that defines the workflow processes of the computer vision system and then load it onto the camera.

To define these processes, we will use the DepthAI API installed on the local server to populate the pipeline with the nodes responsible for accomplishing each task. The pipeline for the computer vision system will consist of 4 main processes being:

- Obtain a live video feed of the parking spots.
- Apply image manipulation techniques (edge detector, hough transform, and regions of interest).
- Apply vehicle detection techniques (Cascade Classifiers, YOLO, or Tiny-YOLO).
- Send this data to the local server and LED display system via ethernet.

The DepthAI API offers some predefined nodes that we will be able to use to accomplish some of these processes. However, it is likely that we will also need to use the script node. In the script node, we will write customized Python code that applies functions from OpenCV to satisfy the needs of our system.

For our system to work properly, we will need to begin with a set of empty parking spaces. The camera will have an aerial view of the parking spaces such that as much of the white lines that define a parking space are shown as possible. Once the OAK-1 PoE is plugged into the PoE switch and the camera is booted up, it will take a picture of the parking spots to be observed and use the image manipulation techniques described in section 5.1.2 to define the regions of interest, i.e., each individual parking spot. These regions of interest will then be used as a mask over the live video, and a rectangle will be drawn for each parking spot. From here, we will use the chosen object detection technique described in section 5.1.2.4 to see if any cars are occupying the regions of interest. If a car enters into a region of interest, that region of interest will go from an unoccupied state to an occupied state and vice versa.

While the computer vision system will have an immediate reaction to a car entering or exiting a region of interest, it will only transmit this data via ethernet to the local server at a certain time interval depending on the time of day and how busy the parking garage is. This will be accomplished via the XLinkOut node on the DepthAI API, which allows data to be sent from the OAK device to the host. In addition to the updates that will be sent, the computer vision system will also transmit a live video feed with an overlay of the computer vision techniques that have been applied.

We would like the final design for our computer vision system to require no human input and work autonomously upon bootup. With this said, applying such advanced computer vision techniques will require extensive testing as there is a lot of room for error. The

methods we will follow for testing as well as the actions we will take relating to the design of the computer vision system depending on our results, are further described in section 10.

8.1.4 Software Flowchart

Figure 17 presents a flowchart diagram showing the workflow processes of the computer vision software. It can be seen that the primary responsibilities of the computer vision system are to detect how many parking spaces are available, transmit this information to the local server, and send a live video feed with an overlay of the computer vision techniques.

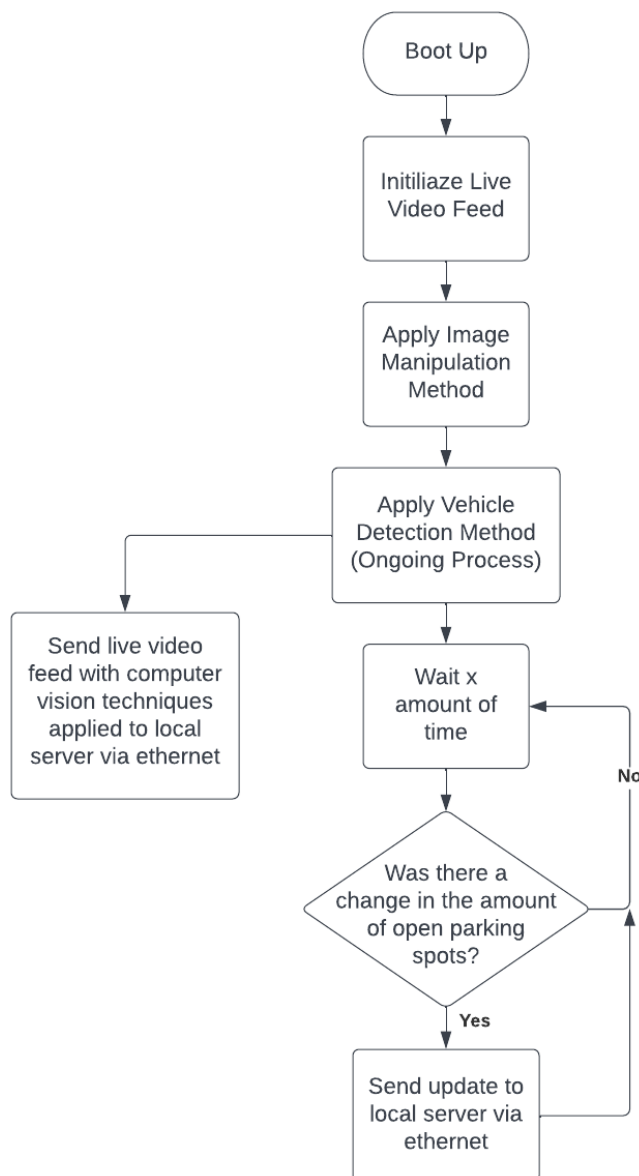


Figure 17: Flowchart of the Computer Vision System Workflow

8.1.5 Hardware Design

Referring to Figure 18 below, a basic block diagram of the hardware within the OAK-1 PoE is shown. This figure is only meant to provide a general idea of the primary connections between the components on the circuit board within the camera and does not show every single connection between the components.

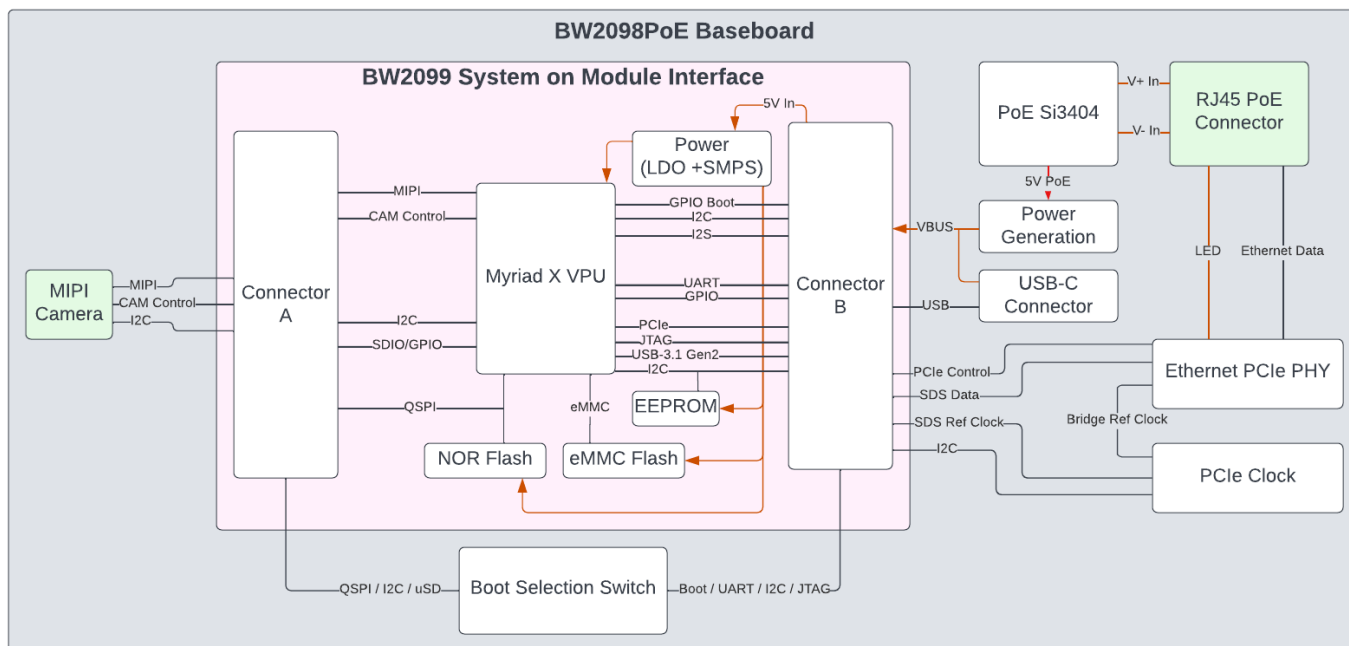


Figure 18: Basic Block Diagram of the OAK-1 PoE Hardware

This camera employs two circuit boards known as the BW2098PoE which is the primary baseboard, and the BW2099 SoM, also known as the OAK-SoM-Pro. The baseboard includes the RJ45 PoE Connector, which receives an external 1000BASE-T ethernet cable that handles both data and power transmission. This connector is wired to the PoE Si3404, which converts the high voltage ethernet connection into a regulated, low-voltage output supply that is suitable for the rest of the circuit board. These boards are interfaced through two 100-pin DF40C-100DP-0.4V(51) connectors which carry all input and output signals as well as the 5V input. The power system onboard the BW2099 employs a switched-mode power supply (SMPS) which regulates the 5V input in order to provide all the necessary digital and analog power for the electronics on the OAK-SoM-Pro.

The primary electronics on the OAK-SoM-Pro include the Movidius Myriad X VPU, a 16GB eMMC 5.1 flash device, a 128MB QSPI NOR flash, and a 32kB EEPROM. USB 3.1 Gen2, QSPI, UART, I2C, 1-lane PCIe, and SDIO, are included on the BW2098POE baseboard and are routed to the OAK-SoM-Pro through the connectors. Additionally, the OAK-SoM-Pro exposes two 2-lane MIPI CSI-2 D-PHY channels and two 4-lane MIPI CSI-2 D-PHY channels, allowing for multiple camera inputs. This system also employs an I2S interface which gives us the ability to connect microphones and an external audio device to the camera; however, this is a functionality we will probably not use for our application.

The baseboard utilizes a boot selection switch which allows the Movidius Myriad X VPU to be booted in a number of ways, including USB-C, EEPROM, NOR flash, eMMC, SPI, and ethernet. This is part of what makes this camera so effective since there is a lot of flexibility in the ways it can be booted; however, these will need to be verified through testing.

8.2 LED Display System Design

Major items to cover when describing the LED display system are the images to be displayed, as well as the software methods on the microcontroller to display the necessary images. This solution will include having two displays daisy-chained to each other back to back with the displays facing outward from each other. This allows for the same messages to be viewed from either side but also results in the MCU displaying an image as though it is on a 32 x 128 display rather than a single 32 x 64 display.

8.2.1 Display Images

The display has to indicate how many open spots there are down an adjacent “corridor” and in some cases, it may display data on two different corridors if the display is situated so that it is both adjacent to one corridor and adjacent to an end corridor of the garage. If there is only one corridor being displayed for, then all that needs to be displayed is an arrow pointing in the direction of indication as well as a total free space count. The open spot value should be updated every time the microcontroller receives new data.

For simplification, images can be shown in 32 x 32 blocks. This would result in one display showing two 32 x 32 images. With the two daisy-chained panels, there would be four separate 32 x 32 image blocks concurrently displaying. Most often, the second panel will be mirroring what is displayed on the other side, except for the numbers being properly flipped. Figure 19 shows an example of a 32 x 32-pixel image that gives a rough depiction of what the display will look like.

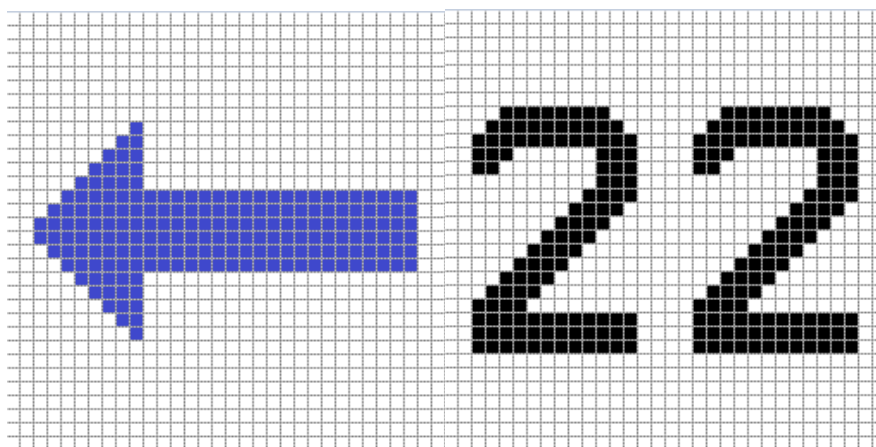


Figure 19: LED Example Images (Double Digits)

In reality, the LEDs will have a small bit of margin between them, unlike these pixel images which have zero margins. However, the actual images will have the same resolution, which is represented through the gridlines. In the final implementation, the colors picked will have a high degree of contrast and visibility for the drivers. If the previously mentioned case of the display being situated around a corner corridor is in effect, then the display would be periodically switching from one sign to another. A time of 5 seconds between display images will be used.

For example, if the adjacent corridor has 22 open spots (as shown in Figure 19 above), and the next corner corridor has 5 open spots, then 5 will be displayed, as shown in Figure 20 below.

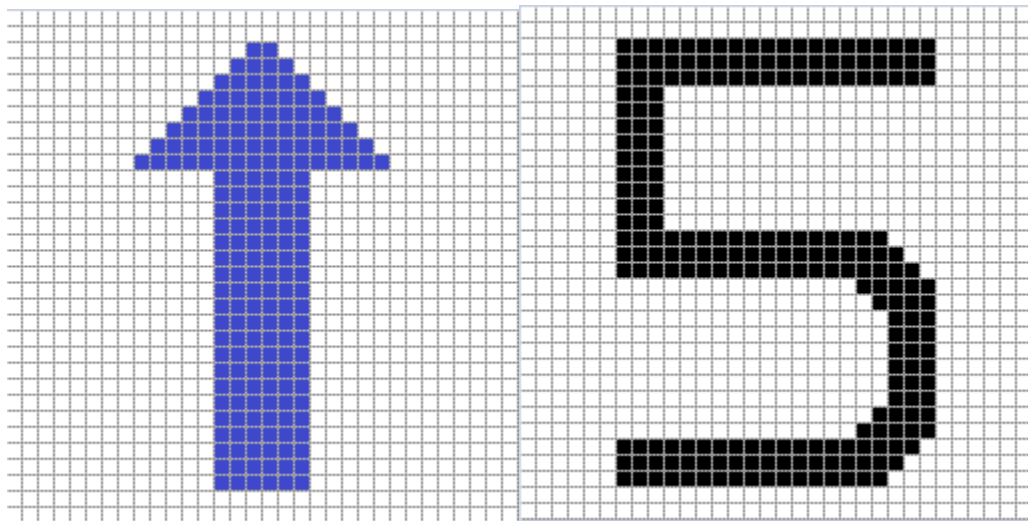


Figure 20: LED Example Images (Single Digit)

When a driver sees the sign, they would first see that there are 22 open spots to their left. After a few seconds, the sign would change to reflect that there are 5 open spots at the next corner corridor if they continue ahead. Since the signs are double-sided, these corner cases will have to be treated uniquely since the side of the sign facing the corner corridor cannot show data on that corner corridor since cars coming from that direction have already been there. Instead, only the adjacent corridor data will be displayed. This results in one side of the display cycling between images while the other side constantly displays one single image.

8.2.2 Hardware

To alleviate requirements from the PCB, the LED display shall be powered by a separate power supply. The LED matrix panel requires a 5V power supply at ~2A. This results in two separate power supplies needed for one cluster of panels.

The panel has many shift registers already integrated into the design, greatly reducing the pin-count required from the microcontroller. Therefore, the display uses a single 16-pin IDC connector, as shown in Figure 21, for the MCU to interface with it

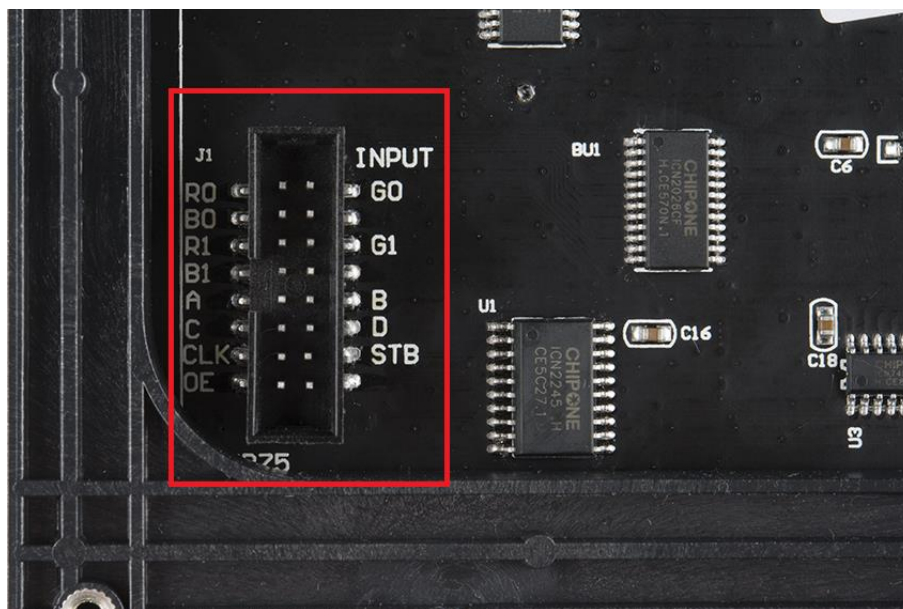


Figure 21: LED Display IDC Connection

Each of the R, G, and B pins correspond to red, green, and blue values for setting the LED colors. There are six of those pins in total, shown in the pinout figure. Pins A, B, C, and D serve as the demultiplexing inputs for the LEDs. CLK, STB, and OE correspond to the clock, latch, and output enable pins for the LED driver, respectively. All other unlabeled pins go to ground. The pinout will be further described under the PCB design section.

8.2.3 Software

Once the PCB is finished, the microcontroller will need to be programmed to both interpret data sent from the local server as well as handle driving the LED display. Thankfully, Microchip provides a Software Development Kit (SDK) for programming LCDs called the MPLAB® Harmony v3, which can help in the development process via its Graphics Suite. With minimal porting, the provided functions from the Graphics Suite should allow the description of shapes and patterns and a high level, making for a much easier process of displaying images on the LED display rather than painstakingly addressing LEDs individually. Even if problems arise with the SDK, porting existing LCD driver functions over should still be a straightforward task and not require creating a new driver from scratch.

An overall target in the software design is for the code to be easy to work with at a high level. In this case, it would mean having functions that handle tasks like displaying numbers on the LED display just by passing in the value received from the local server, or functions that handle cycling through multiple display images if there is extra data to display, etc. Maintaining this high standard of code readability and usability will make for fewer mistakes in the images displayed. Even with a graphics library, the images displayed are abstracted to fundamental shapes, so functions that handle these shapes to display defined images like numbers will make for higher-quality code with fewer errors throughout.

The goal with the LED display driver is for it to be a simple translation layer that has a suite of functions that handle receiving numbers with keywords or other kinds of parameters to potentially indicate the directions the arrows should point in combination with numbers to display. Each of these functions should be accepted only up to two-digit decimal numbers, which allows for keeping the functions small and efficient. Regular int data types of 8-bit precision will more than suffice, with their value ranging from 0 to 255. A pseudocode function call would look like: `spotsInDirection(section_A_spots, LEFT)`

This would be a parent function that would have subsequent calls to other sub-functions that perform lower and lower level tasks for displaying images on the LED display. This would start with a call to a function handling arrow displays, either UP, DOWN, LEFT, or RIGHT. Then there would be a call to a function handling the display of numbers, only accepting inputs ranging from 0 to 99. This function would break down by calling a sub-function for single-digit numbers and a sub-function for double-digit numbers since either of those cases require the numbers to be located in different areas of the 32 x 32 region they will be displayed in. The main function of the program would be greatly cleaned up and easy to follow through this design and implementation methodology.

There is another level of complexity to factor in when considering the different configurations that the LED display could be in. As mentioned earlier, sometimes a display can be situated in such a way that two parking alleys can be adjacent to each other when one of them is the end of the garage, and the other is the next alley in. In this case, at the intersection where a driver can enter the last alley of the garage or the second to last, the LED display should show information for both alleys. An example of this is shown in Figure 22.

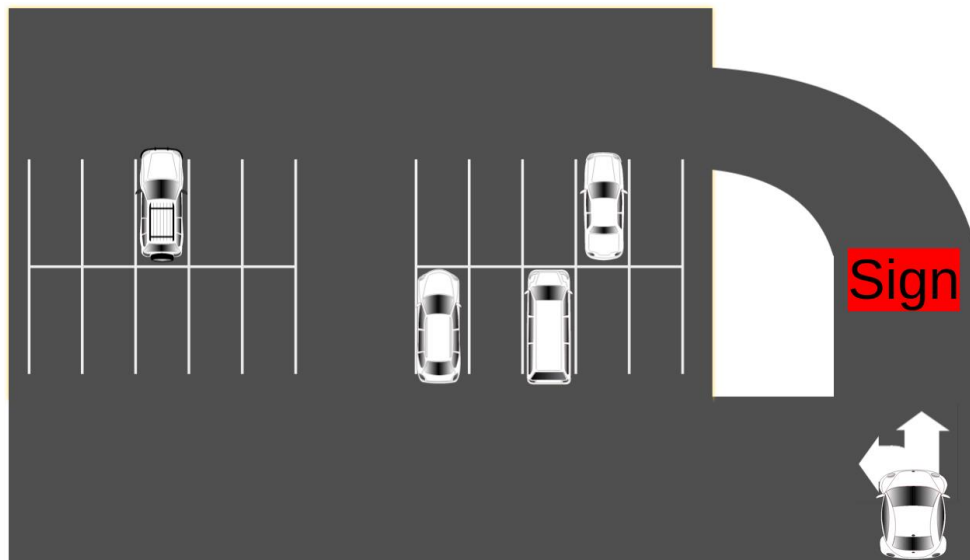


Figure 22: Corner Alley Example,

The general structure of the LED display program, is shown in Figure 23. Notice the key decision point of whether the sign is set up to be displayed for two alleys in the case it is at a corner, or if it should display for just one alley, the normal case.

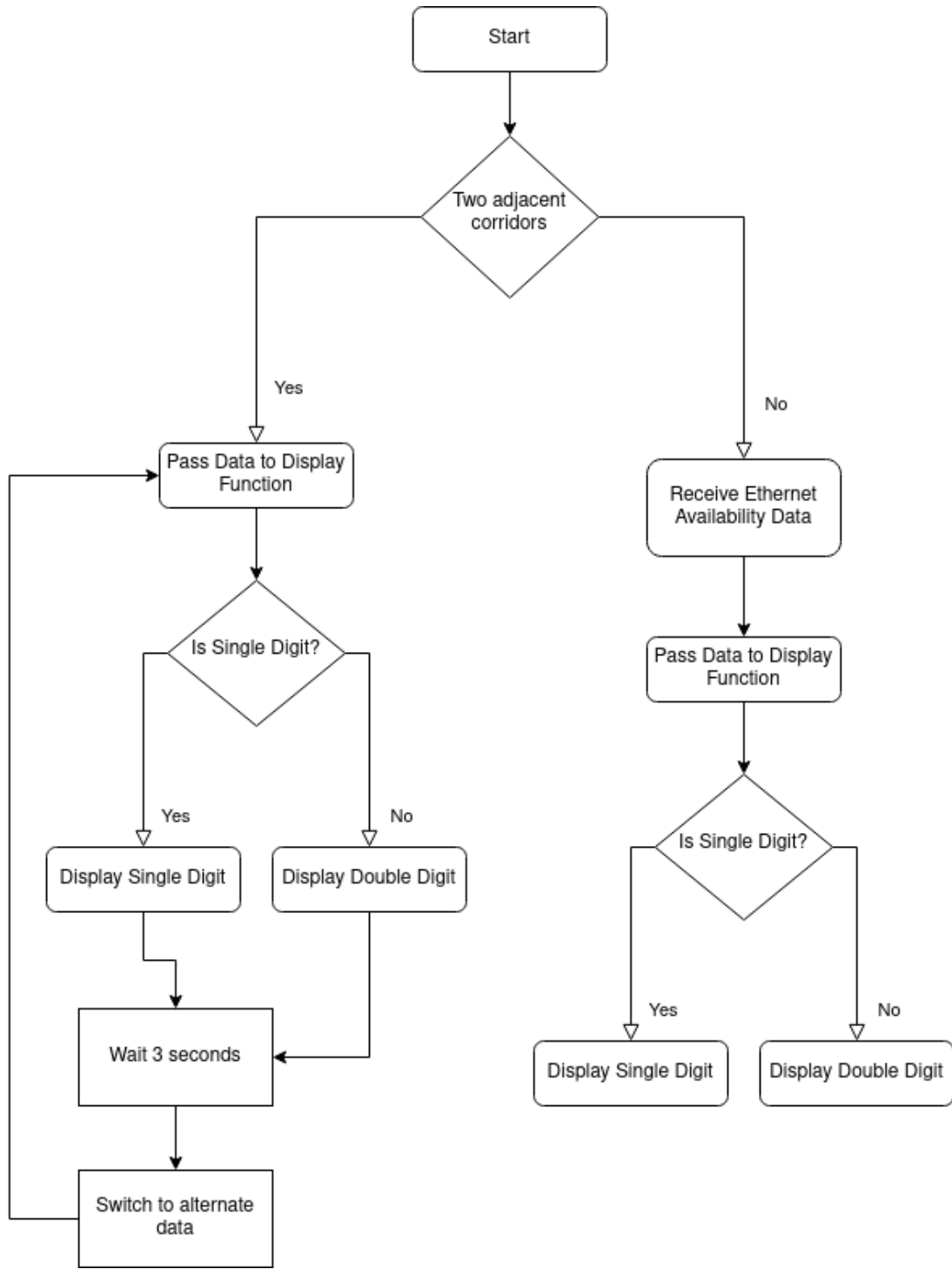


Figure 23: LED Display Program Flow

8.3 Mobile App Design

The parking garage system will offer a mobile app to provide parking availability to users. The app’s primary function is to provide the number of available spaces and the percentage

occupancy of all garages. When selecting one of the garages, it will provide the number of available spaces per level, information about parking services, mobile app developers, app version, and the last updated time. The app will not offer any administration capabilities as they will be offered in the web app version of the parking system. Also, the user will not be required to sign in. As soon as the mobile app is launched, it will fetch updated data from the database and display it within a second or two.

8.3.1 Mobile App Block Diagram

In the block diagram shown in Figure 24, it can be seen that the mobile app design is simple. As soon as the app is launched, the app fetches the data from the cloud database. Then, the app immediately displays a summary of all parking garages on the screen, their available number of spaces, and a percentage of the number of occupied spaces. Users will have the option to tap on any of the parking garages on the screen and obtain additional information, such as the number of free spaces per level of that specific garage. The app will update its data continuously every few seconds at peak times and every few minutes during slow times.

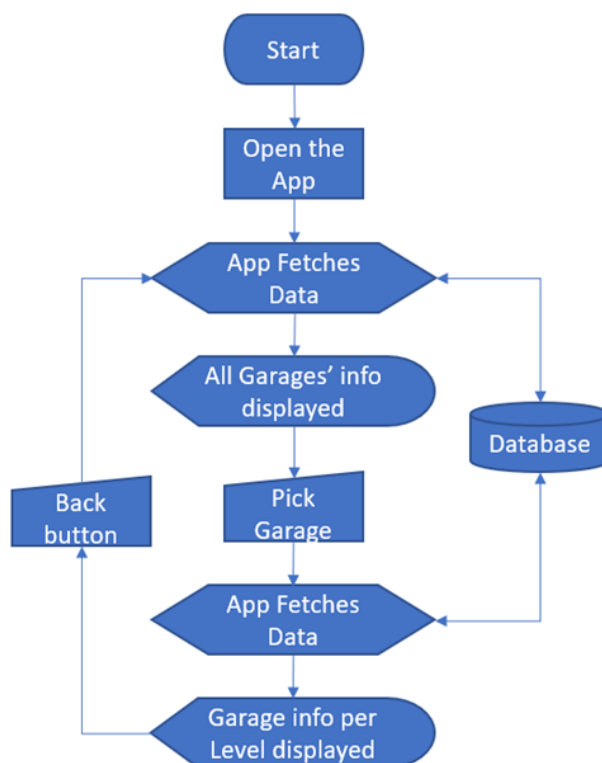


Figure 24: Mobile App Block Diagram

8.3.2 Mobile App User Interface Design

The user interface will comprise three different screens. The first screen will provide the information from all the garages, the second screen will offer the availability of spaces per level, and the third screen will display information about the app and its developers. The prototype design is shown in Figure 25.



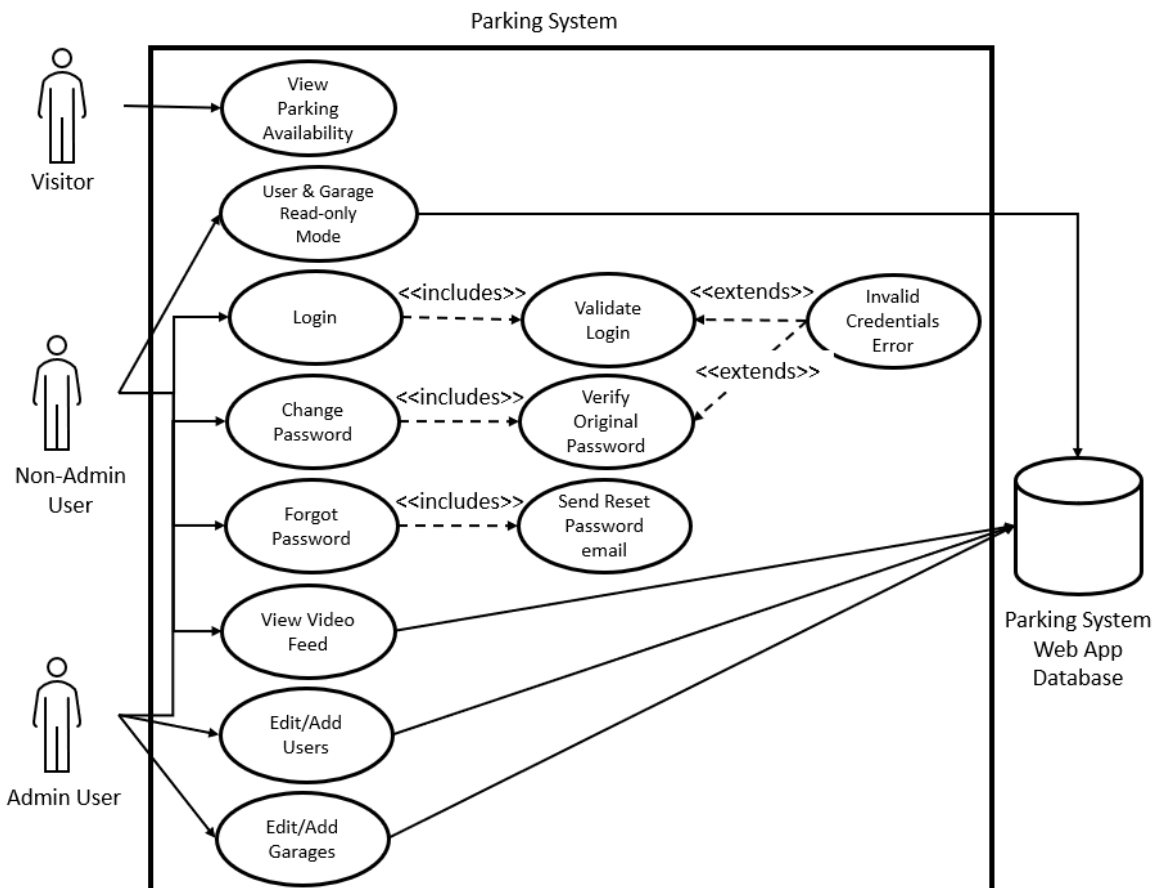
Figure 25: Mobile App GUI Prototype
(Left: main screen, Center: 2nd screen, Right: About screen)

8.4 Web App Design

The web application will offer the same information provided by the mobile app; however, it will also offer parking administration capabilities. Administrators will be able to create, read, delete, and edit new users, parking garages, levels, and sections. In addition, the web app's landing page will show the parking garage system general information such as the number of available spaces per garage, ratio of occupancy, and details of the parking garage levels. This information will be displayed by default without the need to log in. The web app will also offer a login page where admin and non-admin users can access a dashboard where managing the parking system is done. The dashboard will also offer the option to see a video feed of the selected camera in which the computer vision work in progress can be seen. Non-admin users will have read-only access to the web app, while admin users will have complete access.

8.4.1 Web App Use Case Diagram

The use case diagram shown in Figure 26 depicts the user's possible interaction with the web app system.



8.4.2 Database Entity Relationship Diagram (ERD)

The web app, the mobile app, and the local server (clients) will have access to the MongoDB database (DB). In order to serve the clients, this document-oriented DB will contain several collections and documents. There will be a parking system collection that will contain documents for garages, levels, sections, cameras, and logs. These documents will keep data related to the parking system, and the log document will be used to record any incoming and outgoing parking spaces requests for troubleshooting and maintenance purposes. A user collection will keep information related to the admin and non-admin user accounts to administer the parking system. In addition, a user log document will be kept to maintain a record of who did what within the web app. A summary of the collection/documents is shown in Figure 27.

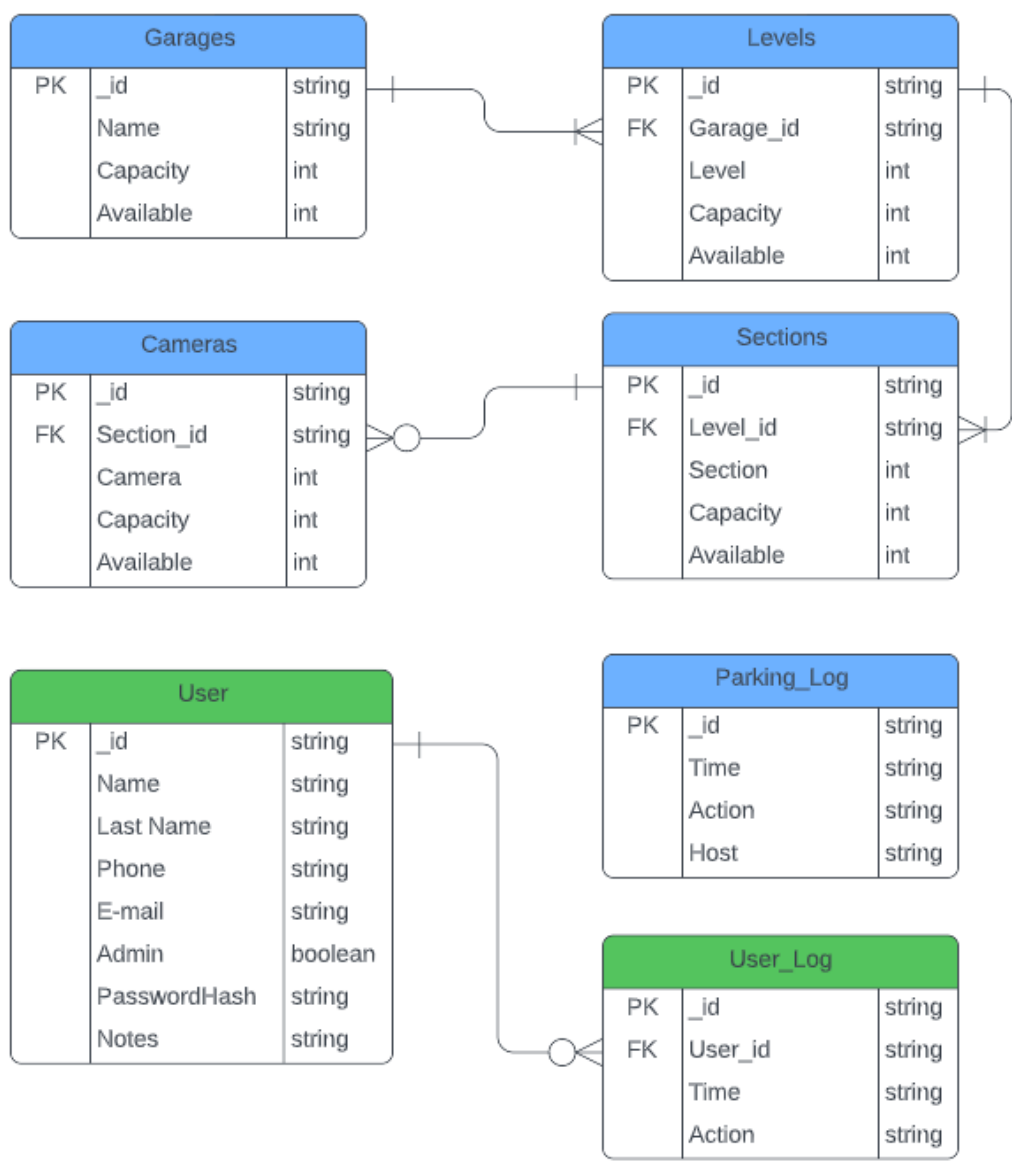


Figure 27: Database Entity Relationship Diagram (ERD)

8.4.3 Web App User Interface Design

The web app will be designed using ReactJS, a part of the MERN stack previously discussed. Here are some preliminary designs for the essential pages of the app, as shown in Figures 28 through 31.

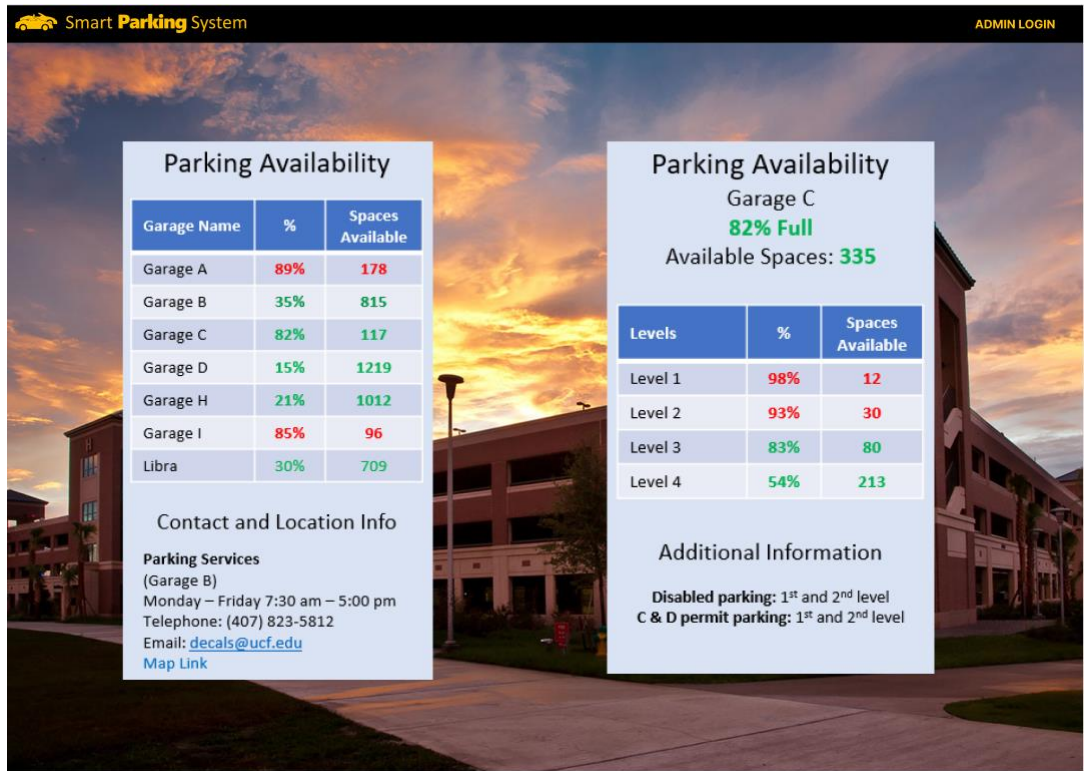


Figure 28: Front Page Design

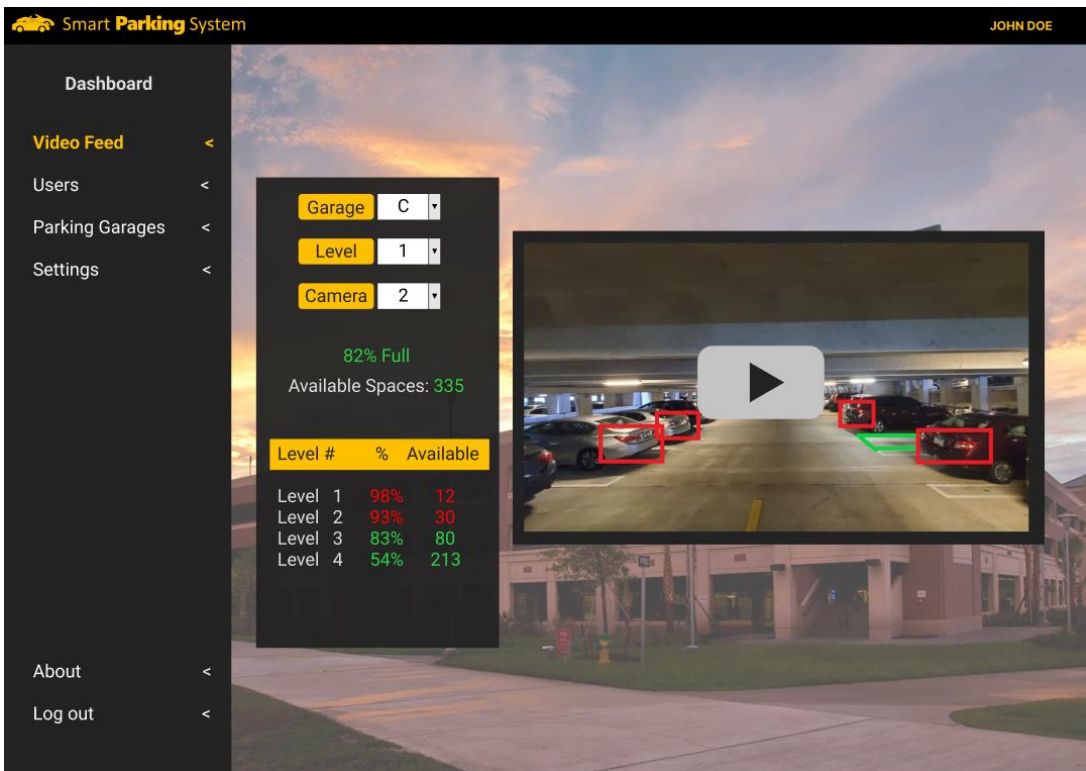


Figure 29: Video Fee Page Design

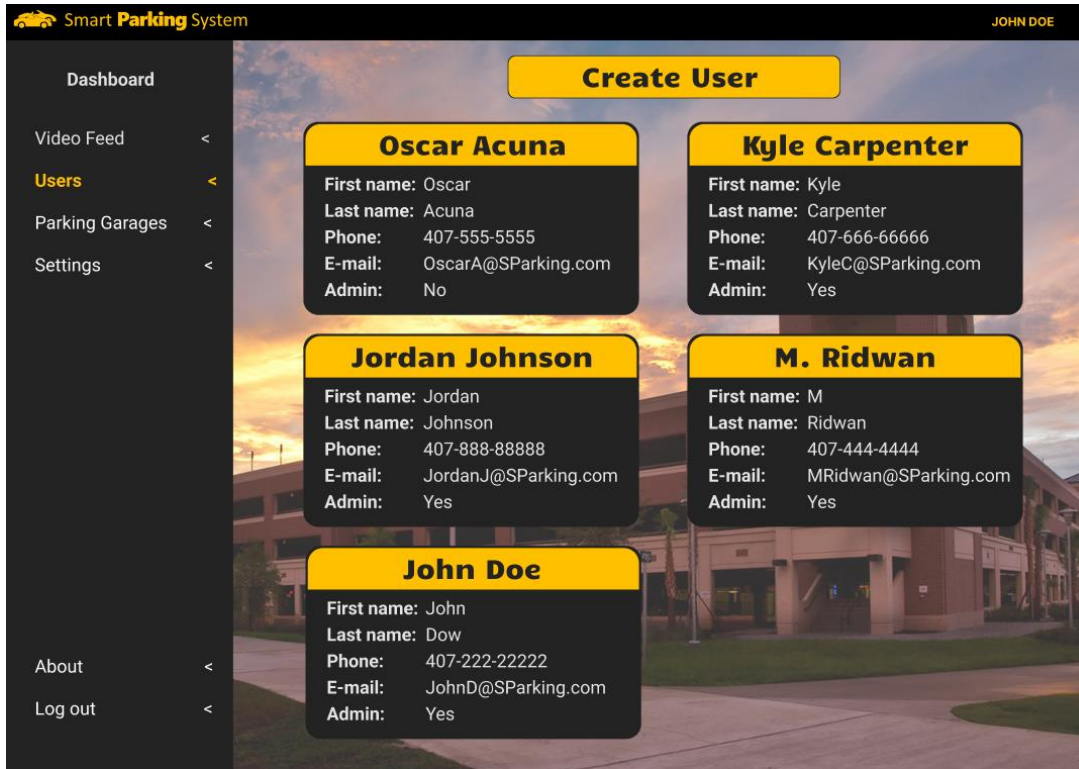


Figure 30: User Administration Page Design

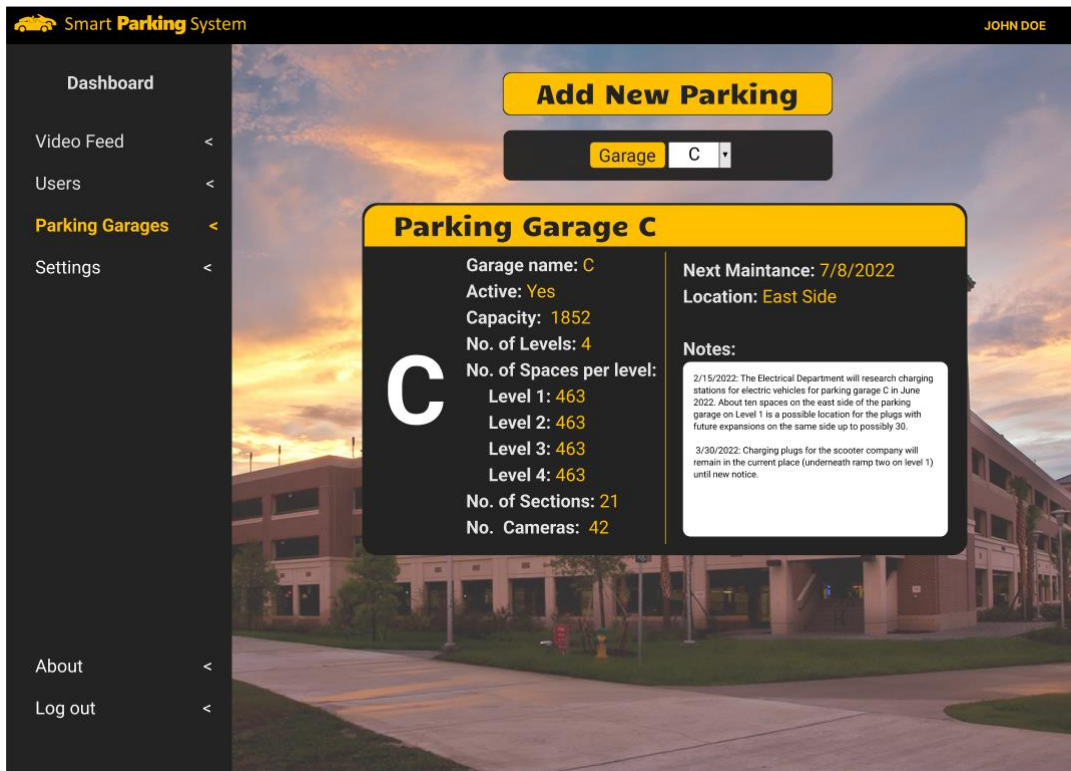


Figure 31: Parking Administration Page Design

8.5 Control Unit Design

The control unit will serve as the central node between the cameras, LED displays, and the cloud web server, and it will contain all the necessary equipment to support the local network, internet connectivity, Wi-Fi signal, and the software running in the server. The control unit is divided into two main components, the hardware (i.e., the equipment) and the software running on the server. Both parts are discussed in the following two sections, including a preliminary design of each major part.

8.5.1 Control Unit's Hardware

The control unit comprises several elements, as shown in Figure 32. For the server board, a single board computer will be used as the control server of the parking system. The selected server board comes with the necessary parts: the CPU, Memory RAM, storage on where to install the operating system, and a power supply that runs on 120V/60Hz. Although this server board provides a display port, no display will be needed for the server to function on its daily tasks; the server can be accessed remotely using remote control software such as TeamViewer. As long as the server has internet access, an administrator of the parking system can access it remotely for updates and maintenance. Nevertheless, a display will be needed for the development stage and for demonstrating the proof of concept since a video feed from the camera with the AI analysis of the parking spaces overlaid onto the video feed is desired.

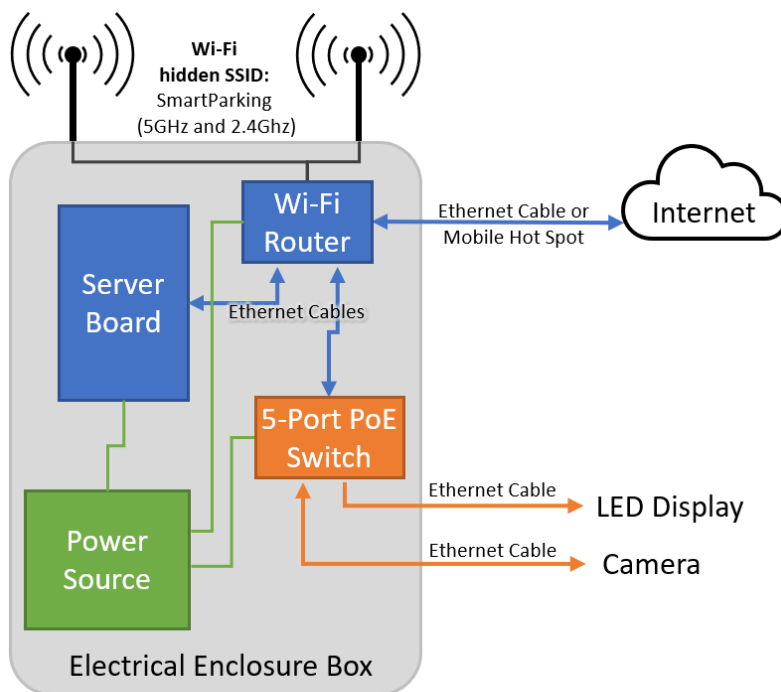


Figure 32: Parking System Control Unit

A 5-port PoE+ switch will provide connectivity between the server board, the Wi-Fi router, cameras, and LED displays, effectively creating a local network. The PoE switch will provide the Power over Ethernet (PoE) necessary to power the cameras and the LED

displays PCB boards. Fortunately, for the proof of concept, this switch does not require any configuration as it is an unmanaged switch; the electricity on PoE ports is not turned on unless a PoE device is connected, in which case, the switch will automatically negotiate the power allocated for the plugged-in device. A bigger switch would be needed for a complete parking system, probably with 24- or 48-ports, with most of them providing PoE.

For the Wi-Fi network, a Wi-Fi router will be used. At the time of this writing, there was an ongoing conversation with UCF's IT department about the proper way of providing internet access to the control unit via UCF's network. Therefore, if we are not allowed to connect to the internet via UCF's network, the team decided to use a Wi-Fi router capable of using a personal phone to connect to the internet as a backup plan. The Wi-Fi network will be used as a backup to the ethernet network. For example, in the case where a camera's power or LED microcontroller's PoE fail, they will still be able to connect using the Wi-Fi network.

In addition to these three main components, two 3-foot patch ethernet cables will be used to connect the server board and the Wi-Fi router to the 5-port switch, and two 25-foot ethernet cables will be used to connect to the camera and the LED display. These Ethernet cables can be either Cat5e or Cat6 as both support PoE and speeds of 1Gbps.

These components will be installed inside a wall-mounted IP-66 electrical enclosure box with 16in x 12in x 6in (height x wide x depth) dimensions. In a real scenario, the site will provide the power source as an electrical outlet where the equipment can be plugged in; however, an extension cord with a power strip will suffice to demonstrate the proof of concept.

8.5.2 Control Unit's Software

The software that will run on the server consists of three main parts, the operating system, the local database, and the custom program that runs the parking system. The custom program facilitates the communication between the cameras, the display's MCU, and the web server in the cloud. For the operating system, Ubuntu Server version 22.04 LTS will be used; the LTS designation stands for long-term support, which means that throughout the lifetime of this release, there is a commitment to update, patch, and maintain. Without long-term support, the operating system can become a security risk since vulnerabilities develop over time [29]. Besides being open source and free of charge, this Linux distribution is compatible with all the necessary software needed for the development of the parking system, such as OpenCV, Java, Python, DepthAI, MySQL, and most importantly, the server board.

Python will be needed to be installed since DepthAI, the program that controls the camera is written in Python. In addition, the custom program will be written in Java. It will contain a graphical user interface where cameras and displays can be added, edited, and deleted. Also, the program will provide the means to link cameras to LED displays. This feature is necessary since many LED signs can display different information based on specific cameras. For example, two cameras can monitor the spaces in a section of the parking

garage, and the information provided by them can be displayed in one or two LED signs; thus, these cameras and LED signs will be said to be linked. A preliminary design of the graphical user interface of this program is shown in Figure 33.

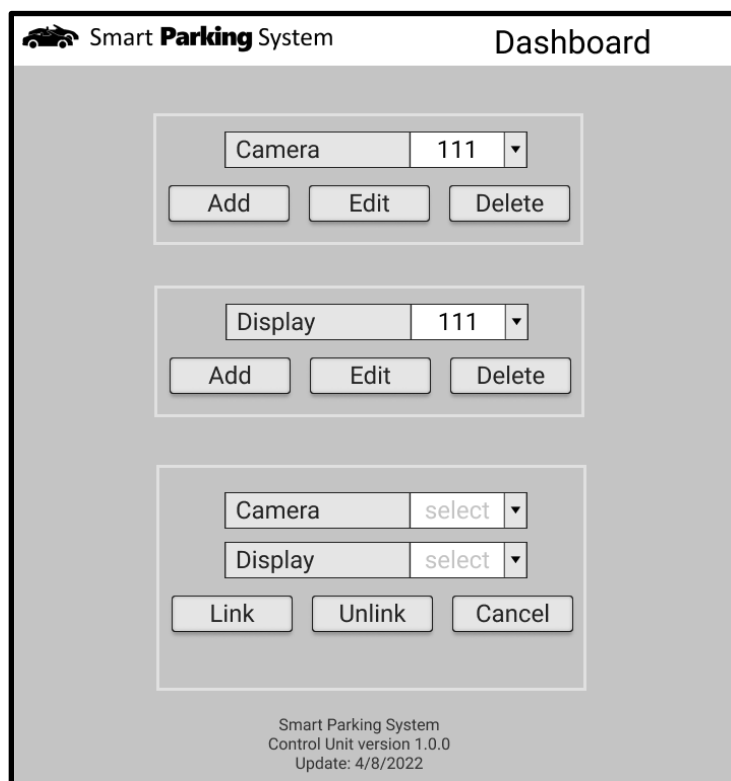


Figure 33: Local Server Java Program GUI Design

The data received from the cameras will be stored in a text file called MM-DD-YYYY-1-02-03.log; this file will be created by the DepthAI (the camera control software) with a name that includes the date of the log, the parking garage level, section, and camera number. The data inside this file will be stored in the following format: a timestamp + 12 bytes that will include the level where the camera is located, the section number, the camera number, and the number of available spaces. (e.g., MM-DD-YYYY-hh-mm-ss 1-01-01-08). The meaning of each digit is summarized in Figure 34.

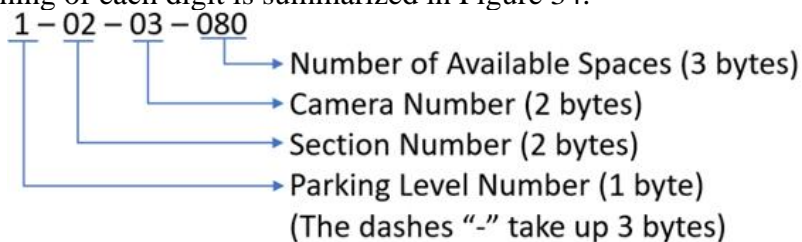


Figure 34: Camera Text File Data Format

The local server will read this data from each camera's logs file stored in the server; it will then update a MySQL local database and send the number of available parking spaces to the LED display linked to the camera that sent the data. This data will be sent via ethernet to the LED's microcontroller to be displayed. Additionally, the server will connect to the cloud database and send this data as well to be displayed in the mobile and web applications.

The database will be hosted locally using MySQL; this database is an open-sourced relational database management system (i.e., tables, rows, and columns). The MySQL program is compatible with the chosen operating system, and it works well with Java and Python. The database can be managed using a command line terminal or through software called MySQL Workbench. This software is a graphical user interface version of the command line terminal, which facilitates the management of the database. Fortunately, one of the team members has some experience integrating Java and MySQL, which reduces the learning curve when coding the program. The database design is rather simple, as shown in Figure 35; basically, it will contain three main tables: a camera table that will maintain information on every camera in the garage; a display table containing the information on every LED display in the garage; a camera/display link table to determine what display to send data from which camera. The camera and display tables will also contain the IP address of each device for the server to determine what address to send what data. The camera id and the display id will be the primary key for their respective tables, and they will be the foreign keys in the camera/display link table.

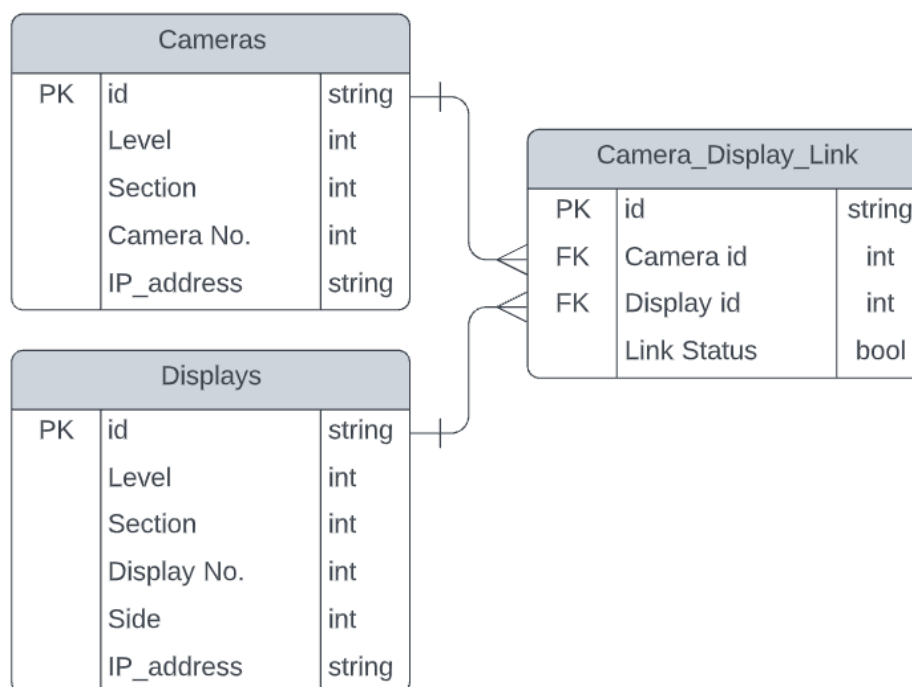


Figure 35: Local Server MySQL Database Design

8.6 PCB Components

This section will introduce the components that will exist on the PCB board along with the microcontroller. The components that will be discussed include an Ethernet PHY Chip, an RJ45 Ethernet Port, Insulation-Displacement Contact (IDC) Connectors, a Step-Down Voltage Converter Circuit, an oscillator, and a push-button. A detailed schematic including all of these components is shown in Section 9.0.

8.6.1 Ethernet Components

To interface an ethernet cable with a microcontroller, there are three major components that are needed, as shown in Figure 36 below. The components are the RJ45 Ethernet Port, Magnetics, the Ethernet PHY, and an external clock. Each of these components is further explained in the following sections.

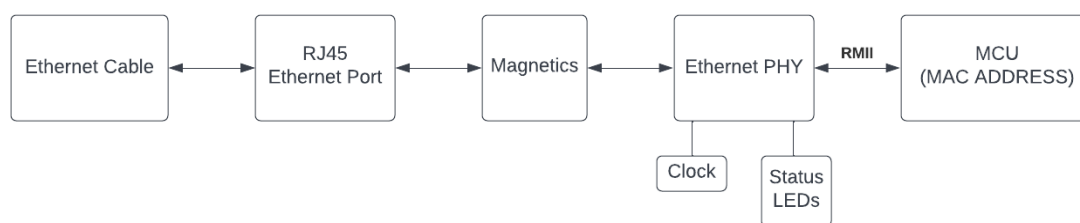


Figure 36: Simple Schematic of Interface between Ethernet Cable and MCU

8.6.1.1 RJ45 Ethernet Port

When our smart parking system is in operation, we will make use of a Power over Ethernet Splitter/Extractor close to the PCB to separate the power and data coming from the ethernet cable. The data from this splitter will be transferred onto a non-PoE cable and connected to the RJ-45 Ethernet port on the PCB. The ethernet port we are using is one of Würth Elektronik's non-PoE RJ45 LAN Through Hole Reflow Ethernet Ports. The connections between the ethernet port and the PHY chip allow for there to be two separate channels where data can flow with ease. One channel is for transmitting data, and the other is for receiving data.

This port is integrated with magnetics which simplifies the design that needs to be done on the PCB and makes for easy connections with the PHY chip. Magnetics are able to be implemented external to the port via an ethernet magnetics module, so using a port where the magnetics are integrated into it will be convenient. This is a vital part of the connection between the ethernet port and the PHY chip as magnetics mitigate electrostatic discharges, ground loops, and noise in addition to providing galvanic isolation.

This ethernet port has two status LEDs which will be connected to the Ethernet PHY to communicate information such as whether there is activity on the PHY chip or not, data speed, and if there is a link between an ethernet cable and the RJ45 port. The LEDs are designed in a way where they are parallel and facing opposite directions such that we can choose between two colors for the LEDs, as shown in Figure 37.

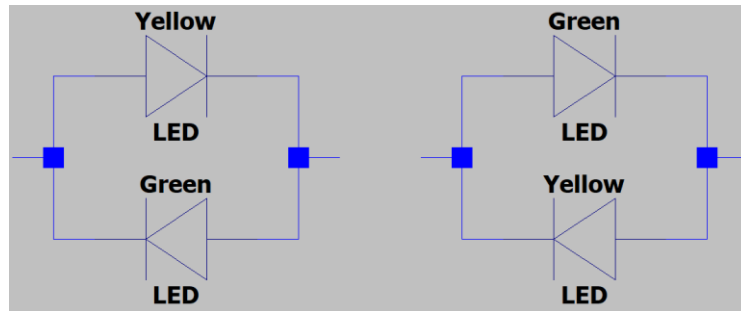


Figure 37: Simple Schematic of LEDs on RJ45 Ethernet Port

8.6.1.2 Ethernet PHY

The Ethernet PHY is a physical layer transceiver device for sending and receiving Ethernet frames based on the OSI network model. The OSI model describes seven layers that computer systems use to communicate over a network, and the Ethernet PHY is used to help cover two of them, the physical layer and the data link layer, as defined by the IEEE 802.3 standard. The PHY chip we have decided to use is Texas Instruments' TLK111. This is a single port Ethernet PHY used for both 10-BASET and 100-BASET signaling. It fits nicely with our design as it has the same power supply requirements as the MCU and provides all the pins we need for our system to work correctly.

The connection between the PHY chip and the media access controller (MAC) of the MCU is called the media-independent interface (MII). There are many variants to the MII; however, we have decided to use the reduced MII (RMII) as this reduces the number of pins required to connect the PHY chip with the MCU. Additionally, since we are using the 64-pin package version of our chosen MCU, it is the only interface that is supported. A table containing the pins that will be used for both the ethernet port interface and RMII is shown below.

Table 20: Ethernet PHY Port Interface/MCU MAC Interface Pins

| Pin | Description |
|---------------------------|---|
| Ethernet Port Pins | |
| TD-, TD+ | Differential Transmit Output: Differential outputs that will be automatically configured to 10Base-T signaling. |
| RD-, RD+ | Differential Receive Input: Differential inputs that will be automatically configured to 10Base-T signaling. |
| MCU MAC Pins | |
| MDC | Management Data Clock: Clock signal for the MDIO interface. |
| MDIO | Management Data I/O: Bidirectional command/data signal synchronized to the MDC. Either the MCU or PHY may drive this signal. |
| TX0, TX1 | Transmit Data Bit 0 and Bit 1 |
| RX0, RX1 | Receive Data Bit 0 and Bit 1 |
| TX_EN | Transmit Enable: Indicates the presence of valid data inputs on TX0 & TX1 |
| RX_DV | Receive Data Valid: Indicates that data is present on RX0 and RX1. |
| RX_ER | Receive Error: Indicates that an error symbol has been detected within a received packet. |
| CRS_DV | Carrier Sense/ Receive Data Valid: Combines the Carrier Sense and Receive Data Valid indications. |

8.6.1.3 External Clock Source

The RMI requires a 50 MHz external clock source to work effectively. This clock signal is fed into the Ethernet PHY and then sent to the MCU as a reference. A table containing the pins used for the clock interface is shown below.

Table 21: Ethernet PHY Clock Interface Pins

| Pin | Description |
|-----|---|
| XI | Crystal/ Oscillator Input RMI Reference Clock: Primary clock reference input. For RMI, this must be connected to a 50MHz \pm 50ppm-tolerance CMOS-level oscillator source. |

| | |
|--------|--|
| XO | Crystal Output: Reference clock output used for crystal only. Since we are using an oscillator input, this pin was left floating. |
| CLKOUT | Clock Output: With the RMII this pin will provide a 50 MHz clock signal. This will allow the MCU to use the reference clock from |

8.6.2 MCU LED Interface

Once the MCU has received the data sent over the ethernet connection originating from the central server through the MAC, it will then need to be reflected on the LED sign. Since the directions for pin connections are written for applications with an Arduino Uno or Arduino Mega, we had to look at the datasheets for the chips on both of these MCUs and see how the pins corresponded with the MCU we are using. After doing this, we found that we would need three types of signals for the MCU to communicate with the LED sign. The signals we are using and a description for each of them are shown in the table below.

Table 22: MCU LED Interface Pins

| Signal Name | Description |
|-----------------------------|--|
| PA0-PA31 PC0-PC31 ... | Parallel I/O Controller (Channels A - E): Manages fully programmable GPIO pins. Six of these signals will serve as digital pins to communicate RGB color values. |
| AFEx_AD0 AFEX_AD11 | - Analog Front-End Controller: Analog Front-End cell integrating a 12-bit ADC, Programmable Gain Amplifier, a DAC, and two 6-to-1 analogs muxes. Four of these signals will be used as demux input pins. |
| PWMCx_PWMH0 PWMCx_PWMH3 | - PWM Controller (Waveform Output High): Generates output pulses on 4 channels independently according to parameters defined by the channel. Three of these signals will serve as the Led Drivers' Clock, Latch, and Output Enable. |

8.6.3 Step-Down Voltage Converter Circuit

When our smart parking system is in operation, we will use a Power over Ethernet Splitter/Extractor close to the PCB to separate the power and data coming from the ethernet cord. This power will be transferred to a 12 V 5.5 x 2.5 mm DC plug, connected to the PCB via a DC jack. From here, we will use Texas Instruments' TPS563201, a synchronous step-down voltage regulator, to step down the voltage from 12 V to 3.3 V. Stepping the voltage down to this level will allow us to provide power to both the Ethernet PHY and MCU without damaging any components on the board. Referring to Figure 38, a simplified schematic for this voltage regulator is shown.

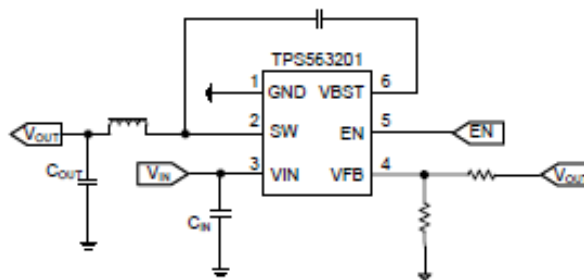


Figure 38: Simplified Schematic for the TPS563201 [$V_{out} = 0.768 \cdot (1 + (R1/R2))$]

In order to choose the electronic components we needed to step down the voltage to 3.3 V, the formula above was used. R2 is the vertical resistor connected to ground and R1 is the horizontal resistor connected to Vout. After plugging in 3.3 V for Vout and solving for R2/R1, we found that R2/R1 was equal to 3.3. Since this is a ratio, we could choose any set of resistors that fit this ratio and decided to work with 33 k Ω and 10 k Ω resistors. In addition to these resistors, the datasheet suggests using an inductor with a value of 2.2 μ H, and Cout can be any value between 20 and 68 μ F.

8.6.4 IDC Connectors

The PCB will include 3 IDC connectors to communicate with components that are external to the board. Two of these connectors will be used for the Ethernet PHY and MCU such that the host computer can communicate directly with each of these chips. The third connector will be used such that the MCU can communicate the LED sign. The IDC Connector that will be used to communicate with the LED sign will be Samtec's STMM-108-01-G-D. This is a 16-pin connector that will allow all digital, analog, and clock connections to be made between the LED Sign and MCU through a ribbon cable. The pins on this connector are set in 2 rows of 8 and have a 2.0 mm pitch.

The two IDC Connectors that will allow us to communicate with the MCU and Ethernet PHY chip for debugging purposes will be Samtec's FTSH-105-01-F-D. This is a 10-pin, 1.27 mm pitch, micro connector which will give a host computer the ability to communicate with the two chips on the board. Each of these chips supports JTAG, which is a common way of interfacing with chips, and it is what we will use as shown by the schematic in section 9.0. To use these connectors effectively, we will need to use a debugger as this will allow us to change the signals on the 10-pin connectors into a signal that is able to be plugged into a USB port on a host computer.

8.6.5 Push Button

The PCB will include one push button, which will allow us to easily reset both the MCU and Ethernet PHY Chip if need be. The reset pin is default high and in order to activate it, it needs to be driven low. This will be accomplished by tying the positive terminal of a button to the reset pin and the negative terminal to ground. When the button is pressed, the reset pin will be driven low and reset the MCU. The Ethernet PHY will also reset when this happens as we will use the MCU to alter an internal register on the PHY that is latched with its reset function.

9.0 Prototyping

The subsystem in our design that will use a PCB is the LED sign system. The components described in section 8.6 were used to develop an initial schematic which is shown in the following section. The large chip that is split in half between the two pictures is the Ethernet PHY chip. Additionally, the pictures below do not yet show the connections for the external clock, LEDs, and some of the values of the electronics are not final. After all testing and prototyping are complete in senior design two, this schematic will then be built onto a PCB.

9.1 PCB Schematic Capture

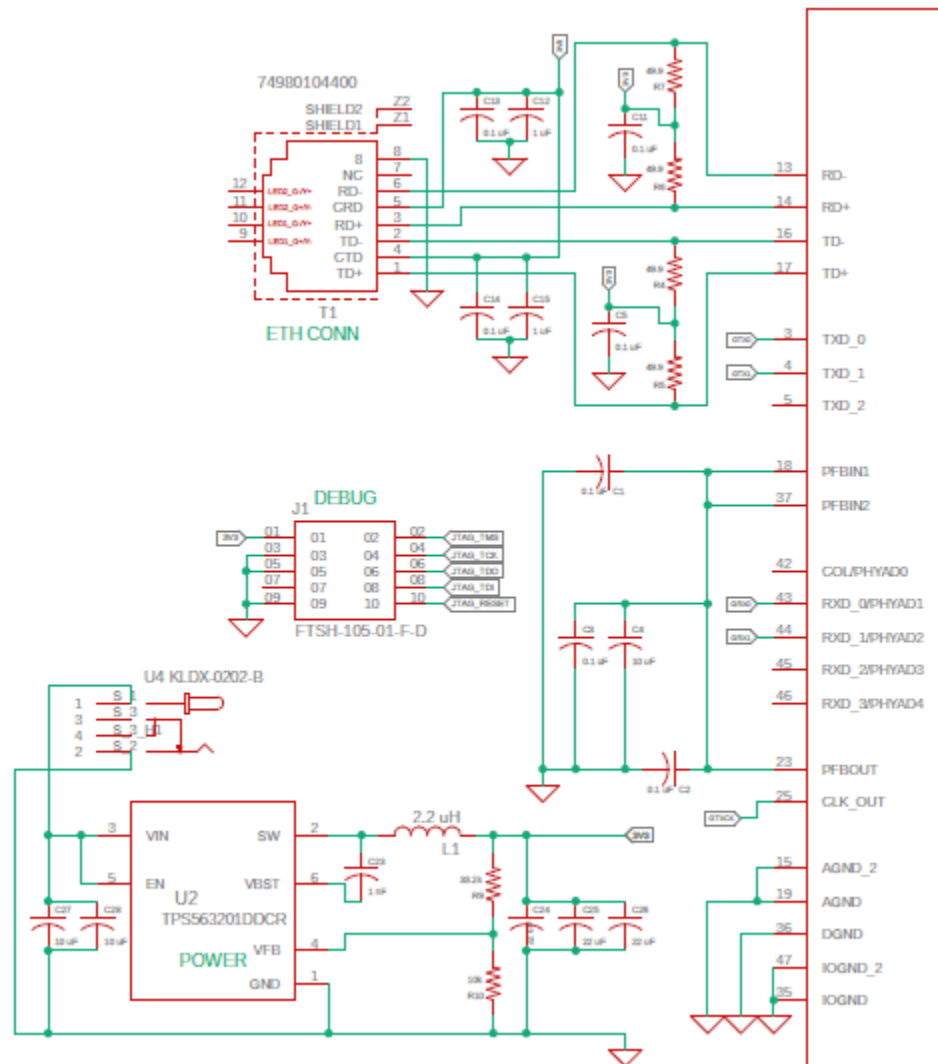


Figure 39.1: PCB Schematics (Left Side)

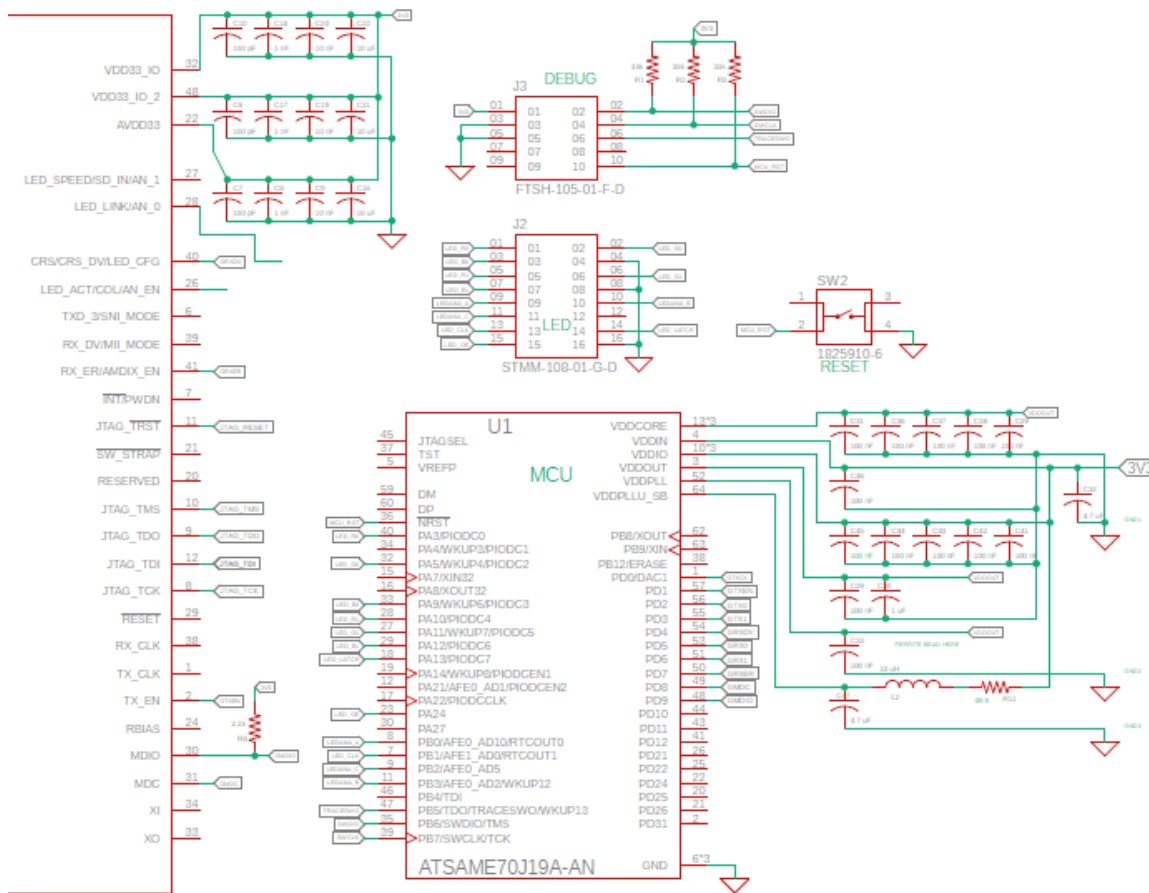


Figure 39.2: PCB Schematics (Right Side)

9.2 Bill of Materials

Table 23 below displays a BOM for the PCB shown in the previous section. The price per unit for each component is represented in the price column, and the value at the bottom of the table shows the total cost of all components with quantity included. All components that are not resistors, capacitors, or inductors, will be ordered early so that we can begin our prototyping and testing process. We will work with through-hole components at first, and then when we have our PCB manufactured, we will order SMD components for the board.

It is important to mention that this is only a preliminary BOM, and the prices are not final. Additionally, this BOM is missing some resistors and an external clock as we have some connections on the PCB that we have not yet finalized. With this said, all components in the current BOM are in high quantity and fit the requirements of our design.

Table 23: PCB Bill of Materials

| Item | Qty | Value | Device | Price | Package | Mfr. | Part Number |
|-----------|-----|------------------|---------------------|-------------|-------------|-----------|--------------------|
| C | 20 | 0.1 uF/ 100nF | Capacitor | \$0.13 | C0805 | Mouser | C0805C104K5RAC7411 |
| C | 4 | 1 nF | Capacitor | \$0.13 | C0805 | Mouser | C0805C102K5RACTU |
| C | 3 | 1 uF | Capacitor | \$0.11 | C0805 | Mouser | C0805C105Z4VACTU |
| C | 2 | 4.7 uF | Capacitor | \$0.29 | C0805 | Mouser | C0805C475K4RACTU |
| C | 6 | 10 uF | Capacitor | \$0.20 | C0805 | Mouser | C0805C106K4PACTU |
| C | 3 | 22 uF | Capacitor | \$0.22 | C0805 | Mouser | C0805C226M9PAC7800 |
| C | 3 | 10 nF | Capacitor | \$0.14 | C0805 | Mouser | C0805C103K1RACAUTO |
| C | 3 | 100 pF | Capacitor | \$0.12 | C0805 | Mouser | C0805C101J1GACTU |
| L1 | 1 | 2.2 uH | Inductor | \$0.81 | | Digikey | SRN8040-2R2Y |
| L2 | 1 | 10 uH | Inductor | \$0.27 | 805 | Mouser | LQM21DH100M70L |
| R | 1 | 2.2k | Resistor | \$0.18 | R0805 | Mouser | ERJ-P06F2201V |
| R | 1 | 10k | Resistor | \$0.16 | R0805 | Mouser | SFR10EZPF1002 |
| R | 1 | 33.2k | Resistor | \$0.19 | R0805 | Mouser | KTR10EZPF3322 |
| R | 3 | 33k | Resistor | \$0.19 | R0805 | Mouser | KTR10EZPF3302 |
| R | 5 | 49.9 | Resistor | \$0.17 | R0805 | Mouser | ERJ-6RED49R9V |
| T1 | 1 | | WE-RJ45 | \$6.02 | | Digikey | 74980104400 |
| SW 2 | 1 | | Switch | \$0.12 | | Digikey | SW_1825910 |
| U1 | 1 | | ATSAME70J 19A-AN | \$12.6 3 | LQFP- 64 | Digikey | |
| J1, J3 | 2 | | FTSH-105- 01-F-D | \$2.08 | | Digikey | |
| U4 | 1 | | KLDX-0202- B | \$0.68 | | Digikey | |
| J2 | 1 | | STMM-108- 01-G-D | \$5.56 | | Samtec | |
| U3 | 1 | | TLK111PT | \$7.62 | LQFP- 48 | Rochester | |
| U2 | 1 | | TPS563201D DCT | \$0.80 | SOT-23 | TI | |

| | | | | | | | |
|--|--|--|---------------|----------------|--|--|--|
| | | | Total: | \$47.29 | | | |
|--|--|--|---------------|----------------|--|--|--|

10.0 Testing

To ensure that our final product for senior design 2 functions as expected, each of the major components within our overall system must go through a testing process to ensure they are working properly. This section is divided into two sections: hardware testing and software testing, and we detail the process each component will go through before being implemented in our final product. Additionally, we will address the results we expect from the testing processes, and the steps we will follow in the case testing does not go as expected.

10.1 Hardware Testing

The subsystems within our overall design that will need hardware testing include the computer vision system, the microcontroller, the PCB, and the PoE switch. The testing procedure for these components is explained in the following sections.

10.1.1 Computer Vision System Hardware Testing

With the camera being the primary sensing element in our system, it is vital that the hardware components within the camera are working properly and functioning as expected. The testing procedure for the computer vision system hardware will verify that the easily testable components are working. This includes powering up, Ethernet connection, data transmission over Ethernet, and verification of the camera sensor working. By nature of the OAK-1 PoE being an All-in-One solution, we are limited in exactly what we can test. Therefore the hardware testing for the computer vision system will not be extensive.

One basic component that needs to be tested straight away is the power supply. If the camera doesn't power on, then that needs to be addressed immediately before any other tests can be done. The power supply setup is unique in that it relies on the power delivered over the Ethernet cable since it is a PoE system. Because the power is delivered over Ethernet, if the camera does not receive power upon connection, then there is most likely something wrong with the switch that all the Ethernet connections go to. The switch will be supplying power over Ethernet unless it does not have this capability, in which case we will use injectors.

If the PoE system is proven to power successfully on the OAK-1 PoE camera, then the next course of action is testing data transmission over Ethernet. For the final implementation, a preconfigured image will already be flashed onto the internal eMMC flash, and it will automatically be sent over data as programmed. But for our tests, we want to see basic functionality before we invest time into implementing the whole computer vision code. One way to see if the device is connected to the Local Area Network (LAN) properly is to check the listed connections with the local server that drives the system. If the local server detects the OAK-1 PoE camera, then we are set for testing custom data transmissions.

The final aspect of the camera we can test is the data transmission. Since the camera's primary purpose is to detect objects using the built-in OpenCV library, we can use a program that either makes use of the library or performs some other form of sending data

over the local network. This test would be simple and largely dependent on the local server listening to the data. If the local server receives the data correctly, then the OAK-1 PoE camera can be determined as having its hardware features working.

The final aspect of the camera we can test is the video feed. The purpose of the camera is not to relay footage over to the local server but rather to detect any objects, specifically vehicles, and send relevant data about those objects. But we will obviously not be able to receive any relevant information if the camera sensor is not functioning properly or the mainboard of the camera has an improper connection with the camera sensor. Therefore, we can test this by relaying the data over the Ethernet connection to the local server. Using the DepthAI Python API, we can run a demo provided by Luxonis that performs basic image detection but, most importantly, displays the camera feed as well, as shown in Figure 40.

With these tests all passing, the OAK-1 PoE camera is ready to go for implementation.

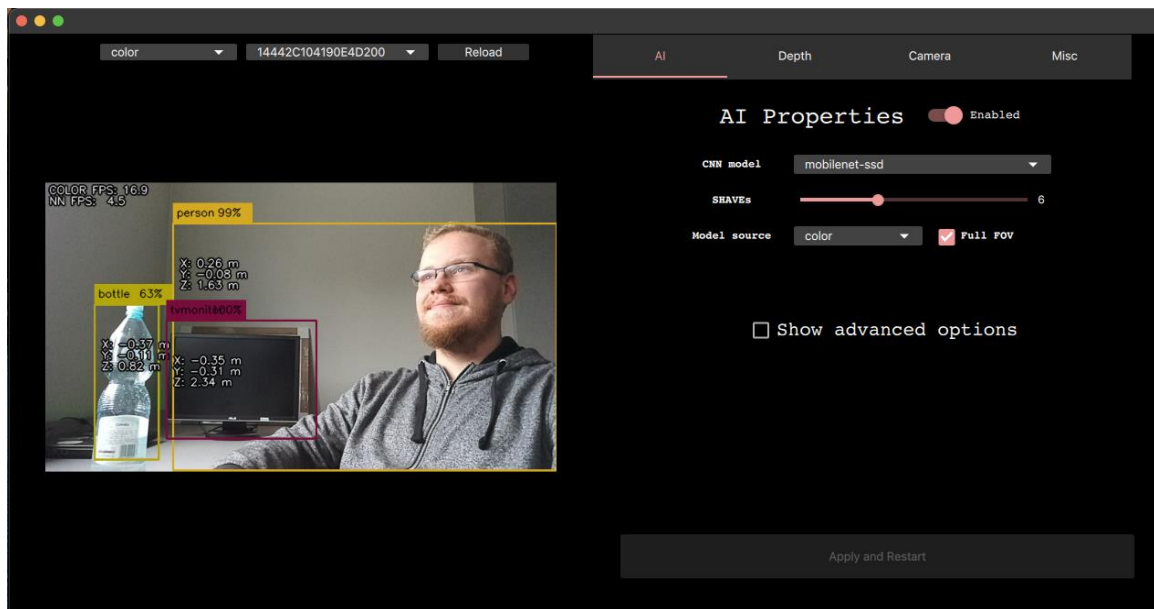


Figure 40: Luxonis documentation example result of `depthai_demo.py` running

10.1.2 Microcontroller Hardware Testing

The microcontroller will need to be tested for a number of its capabilities, including communication over MAC from the integrated Ethernet controller to the Ethernet PHY chip and PWM capabilities for driving the LED display. In order to easily prototype our design, we will make use of a breakout board that fits the specifications of our MCU. This breakout board is designed for the Quad flat package (QFP) and a 64-pin layout. Its dimensions are 10 x 10 mm with a 0.5 mm pitch between pins. This fits perfectly with the dimensions of the Microchip ATSAME70J19A-AN, which uses LQFP (low-profile QFP) and fits the same standard 10 x 10 mm area with 0.5 mm pin pitch. The breakout board outputs all of the pin connections to contact pads that allow for either through-hole connections or surface-mount type connections. This allows for headers to be soldered onto the board, which would allow for jumper cable attachments. Additionally, since the spacing

of the output pins lines up with a standard breadboard with headers, the breakout board could connect directly to a breadboard making for simple testing and prototyping.

The only way to confirm whether any component is working as desired is to first program the microcontroller and run the test code. When testing MAC, PWM, or any other communication interface, a C program will need to be written to drive the pins on the microcontroller to output through the corresponding pins for the aforementioned purposes.

For observing the outputs, a multimeter will be needed to observe the voltage levels. This way, we can ensure that the pins are within the rated parameters described in the datasheet. When reading data sent over the pin, we will need an oscilloscope to observe the waveforms and a logic analyzer for reading the digital signals. The oscilloscope will allow us to inspect the integrity of the signals, while the logic analyzer will read the waveforms as digital signals, giving us more clarity in debugging and analyzing the interfaces.

In our initial testing, all we are looking for is basic functionality from all the needed peripherals on the microcontroller and to verify they behave as expected. As long as this is the case, we can continue testing the results of code flashed onto the microcontroller without worrying whether there are any defections in the MCU itself or whether our understanding of how the function on the MCU works is wrong. This speeds up the debugging process greatly because there are fewer variables in play to contend with when debugging possible points of failure in our design.

In order to run code on the MCU, we need a proper interface to flash the onboard memory with our code to execute. Like a lot of modern microcontrollers, the Microchip ATSAME70J19A-AN can be programmed through JTAG. Specific to Arm-based microcontrollers, this can be debugged through a 2-wire variant of JTAG called Serial Wire Debug (SWD). A common problem is finding a proper hardware debugger that is both compatible and also cost-efficient. There are many vendor-specific hardware debuggers available, but most of them come with a hefty cost. Thankfully, due to the open-source community, there is another option to use any hardware debugger. Through the use of the open-source software OpenOCD, we can use an existing hardware debugger like the popular ST-Link (which is designed for STMicroelectronics devices) for our own Microchip MCU.

With a compatible hardware debugger, we can write any code to our microcontroller for testing. There is another feature that the debugger can do besides being a mode of flashing the MCU; it can also debug the code, as the name implies. This will be further highlighted when describing the software side of testing.

10.1.3 PCB Hardware Testing

The PCB will have a variety of hardware components that need to be tested separately from the MCU, including the Ethernet Port and Cable, Ethernet PHY, and the Step-Down Voltage Converter Circuit. The following sections will detail the procedures we will follow for each of these components before mounting them onto the final PCB.

10.1.3.1 Ethernet Port and Cable Testing

The testing procedure for the ethernet port will verify that ethernet data is able to be delivered to the MCU. Before we have our PCB manufactured, we will utilize an ethernet connector breakout board which will allow the 8 pins of the RJ45 ethernet port to be easily interfaced with a breadboard. Since the ethernet port we are using has integrated magnetics, the circuit between this breakout board and the breakout board used for the PHY chip will be fairly simple. From here, we will prototype and troubleshoot any connections between pins before making the decision to place the RJ45 ethernet port onto the PCB.

Since the ethernet port will help in linking the Ethernet PHY with the rest of the ethernet network, it is important that we make the correct connections such that there are two channels for transmitting and receiving data with no interruptions. To do this, we will follow the schematics available in the datasheet for the ethernet port very closely. Additionally, we will need to use an ethernet cable tester between the ethernet cable and the ethernet port to tell us if each end of the cable we are using is following the same wiring standards described in section 6.2.1. If the ends of the wire are not following the same standards, we must either change the wiring on the breadboard or change the wiring inside the cable.

10.1.3.2 Ethernet PHY Testing

The testing procedure for the Ethernet PHY will verify the data speed we are able to achieve with our system, 10BASE-T or 100BASE-T, in addition to making sure all connections between the ethernet port and MCU are correct. Since the PHY we are using has a JTAG interface, we are able to communicate directly to the hardware chip through a host computer. We will use this JTAG interface to complete a boundary scan on the PHY chip and ensure that every external wire connected to this chip is on the correct pin.

In addition to the major connections for data transfer between the ethernet port, MCU, and PHY chip, we will also need to test the status LED drivers and the clock. Testing the status LED drivers will show us each of the modes that these pins are able to function in and the type of data that is transmitted with each mode. To test the external clock that is connected to the PHY chip, we will simply need to use the JTAG interface to verify that the PHY chip is receiving a 50 MHz signal and transmitting a 50 MHz signal to the MCU.

10.1.3.3 Step-Down Voltage Converter Circuit Testing

The testing procedure for the step-down voltage converter circuit will verify that the voltage signal coming from the DC jack is able to be stepped down to a level that is suitable to power the MCU and Ethernet PHY without causing any damage. The voltage regulator in this circuit uses a 6-pin 1.6-mm x 2.9-mm SOT-23 package, so we will need to use a breakout board that fits this in order to get access to the pins for prototyping. All of the power circuits in our design will use a variety of electronics, so it is important to mention the type we will be using. We plan to use surface-mount electronics when we order our PCB; however, during the prototyping phase, we will be using through-hole electronics to model all power circuits.

The step-down voltage converter circuit will use a DC jack at the input, so the first thing we will need to do is use a multimeter to measure the voltage across the input and ensure that we are seeing 12 V. From here before we make any connections between the MCU, Ethernet PHY, and the output of the step-down voltage converter circuit, we will need to verify we are seeing the expected output voltage of 3.3 V. This will confirm that the voltage output is at a level required for the safe operation of the MCU and Ethernet PHY and no damage will be caused. In the case that we decide that we would like our design to be a bit sleeker, instead of using a DC jack, we would replace the ethernet port on the board with one that is designed for PoE applications. This would have some drawbacks though, as this introduces challenges with finding a voltage regulator or buck converter that is able to handle a PoE voltage input range.

10.2 Software Testing

The subsystems within our overall design that will need software testing include the computer vision system, the central server, the microcontroller, the web app, and the mobile app. The testing procedure for these components is explained in the following sections.

10.2.1 Computer Vision System Software Testing

The computer vision software testing procedure will verify that the software is able to define what a parking space is, compute whether that parking space is occupied or not, and receive/transmit data with the central server over ethernet. We will not need the camera on hand to start testing the first two requirements, as we will only need some sample pictures and videos of UCF parking garages. We will write a Python program that takes these pictures and videos as input and see how the canny edge detector and hough transform performs on them. If the program is able to correctly define what a parking spot is and mark out the regions of interest, we can then move on to the second requirement.

For the second requirement, we need our program to be able to differentiate between a human or a car occupying a region of interest and only respond if a car enters or exits the parking spot. To do this, we will make a video where we use the group members to walk through parking spots in addition to using our cars to occupy the parking spots. If the region of interest only changes to occupied when a car is inside of it, then we will conclude that this requirement has been met.

To test the final requirement, transmitting/receiving data with the central server, we will need to have the camera on hand. Once we have access to the OAK-1 PoE we will write a simple program to send numerical values and a live video feed over the ethernet connection. We will use the XLinkIn and XLinkOut nodes available on the DepthAI Python API to accomplish this. If the values are able to be sent back and forth between the server and the camera is able to deliver a live video feed to the server, we will conclude that this requirement has been met.

With us using advanced computer vision techniques in our design, we need to have a backup plan for the software design in the case that parking space detection does not

function as expected and presents us with too many issues. Some of the methods described below would require our system to need human input before it is able to function instead of working autonomously; however, we know they will be easier to implement and not present as many issues.

The first method we are looking at implementing is using bright colored squares on each of the parking spots that we want to be monitored. The computer vision software system will then count these squares and return the number of open parking spots based on the number of squares it is able to see. This method would get rid of all object detection features and be much more simple than the method described in previous sections.

The second method we are looking at implementing would involve using OpenCV's mouse as a paintbrush function. The moment we set up the camera, we would simply use a mouse connected to the server to draw on the screen the parking spaces we would want to be monitored and then define these as our regions of interest. From here, we would then use the chosen object detection method to compute whether a car is occupying a region of interest or not. Using this method would be simpler than our original method but also more complex than the method described in the paragraph above since we would still be using object detection functions.

The final method we are looking at implementing would involve not keeping track of each parking spot but instead keeping track of the number of cars that enter and exit a row of parking spaces. This method does have one drawback being that it would require us to buy a second camera so that we are able to monitor the entry/exit points on each side of the row; however, it would also allow us to keep track of many more parking spaces. Using this method would negate the constraints caused by the ceiling, as explained in section 7.2, which makes this a very attractive option.

10.2.2 Local Server Software Testing

Several tests are required to ensure the correct functioning of the local server. The team needs to ensure that all the components are functioning as designed, such as having an internet connection, connectivity with the cloud database, connectivity with the camera, and the displays' microcontroller. One way to check that the local server has internet access is by pinging a website outside the local network like yahoo.com and waiting for a response (e.g., ping yahoo.com). If there is a response from this website, then it means the server has internet access and that it can see the network outside the local network. This test can be performed in the Linux command terminal in Ubuntu.

After internet access has been established, a check of the connectivity to the cloud database must be performed. Since the program will be done in Java, then MongoDB drivers for Java must be downloaded and imported into a small Java program. Then, a new MongoDB client variable is declared and initialized with the internet address of the MongoDB database site, then JAVA is instructed to get the database information from the provided address and provided credentials. If everything is configured correctly, the program will display the database information; otherwise, an error will be shown instead.

Checking the connectivity of the local server with the cameras is straightforward. Once the cameras are connected to the PoE, switched and turned on, and python and DepthAI are installed in the local server, a simple python script provided by DepthAI documentation can search the local network for connected cameras. The script will display the camera's IP address and other camera information. If the script does not find any cameras, the camera may not be on, or it may not have obtained an IP address.

The connectivity with the display's microcontroller is also straightforward. A ping command ran from the Linux command terminal using the microcontroller's IP address which has previously been assigned, can help determine if the server can talk to the microcontroller. Connectivity has been established if there is a response from the microcontroller.

It is important to note that the camera and the display's microcontroller are assigned IP addresses before these connectivity tests. The wireless router will contain a DHCP server which will handle and give IP addresses to these two devices. Two IP addresses will be reserved beforehand for these two devices using each ethernet port's MAC addresses, so when they initially connect to the network and request an IP address, the DHCP will see their respective MAC addresses and assign the reserved IP address. This is essential because the local database will maintain the IP address of the devices connected to it in its records, and these addresses should not change.

One last test can be done on the wireless router's Wi-Fi network. We can scan for available Wi-Fi networks using a smartphone built-in Wi-Fi app to determine if the Wi-Fi parking system shows up on the list. If it does, we can ensure that the phone can connect to that Wi-Fi using a predetermined password.

10.2.3 Microcontroller Software Testing

Testing C code written for a microcontroller can be more cumbersome than that of PC targeted C code. When writing code for a PC, it can be quickly debugged since the target platform is the same platform the code is generally written on. When writing C code targeted at a microcontroller, the code has to be compiled and then flashed over to the MCU. This adds extra steps to the process, which discourages programmers from implementing minute changes in code before recompiling and testing again. Additionally, there are few options for observing the output of code when running on a microcontroller, unlike a PC targeted executable which can be observed in the terminal, with functions like print statements used for aiding in debugging.

There is a solution that helps to alleviate the problem of debugging for microcontrollers, and that's the Serial Wire Debug capabilities of the Arm Cortex-M7 used for our target microcontroller. Through SWD, we can step through code and manipulate the flow of the program to better observe the behavior of the code on the MCU. Though we cannot see direct output to a terminal through print statements, we can observe the direct effects of the code on the physical outputs either through a logic analyzer or oscilloscope, as mentioned in the hardware testing section. This almost completely eliminates the problem posed by

the limitation of needing to flash the target microcontroller when code can immediately be run on a step-by-step basis when it is connected via a hardware debugger.

10.2.4 Web App Testing

For web applications some of the core testing to make sure the smart parking system is catered to the end-users sufficiently are - security aspects of user info, the performance of the website while broadcasting video footage, graphics user interface usability also the core functionality of reading the parking data and showcasing them onto the web application.

Functionality testing is indeed the main testing requirement for this project. The web application needs to be able to access the data from the server database and showcase the data on different components added to the website. So, after the computer vision testing, when the data of parking locations and the number of open parking spots have been determined - the web app API needs to be able to gather the data from the local server database. This data further needs to be displayed on the front end of the web app with proper effects and positioning for the end-users to be able to access the data easily. Thus, the receiving and updating of the data as well as showcasing of the updated data, needs to be tested deliberately for proper implementation. This testing can be done through numerous interactions of parking data gathered over time. Frequently changing parking situations and observing the data displayed on the web application will ensure the accuracy of the accuracy.

Along with the functionality, efficient implementation of the graphical user interface is crucial. In modern days one of the main drivers of usability and increasing user traffic for a system is to have a user-friendly interface for the consumers to interact with. Smart Parking is planned to have a user-friendly web app interface for the customer to interact with the system with ease. This includes visual representations, complementing color themes, and user-friendly components to enhance the user experience. Since the smart park app is catered to UCF students, a black and gold theme is intended to be applied. The web application would allow multiple pages to be hosted; that way, the data on the web application are not all clustered together. The vacant positions in the parking spots are designed to be in green font color and red for the taken spots. A web page is designed to showcase the live video feed of the parking spots. To test the user experience, multiple students would be asked to try out and browse through the web application. That was, the usability of the web application can be tested, and with the feedback gathered, there would be the potential scope for improvements.

For a steady and reliable connection between the web application and the local server set (which gathers the data from the cameras), interface testing needs to be implemented. Smart Parking web application will communicate with the server database in real-time to ensure a direct data feed for the end-users. Thus, live, and agile communication between the application and server needs to be thoroughly checked over a long period of iterations to confirm edge cases. Also, the database can crash, or momentary network interrupts would cause the web application to stop acting accurately. Thus, to prevent mistakes in showing the intended data, a notification system or an error message system needs to be

set and tested to communicate with the users if needed. Interrupts while browsing through the web applications for the users also would be checked.

Web applications for smart parking also should be tested for compatibility. In the present day, there are a lot of technologies that support a lot of different types of web applications. However, depending on operating systems, different devices, and web browsers, the architecture of the web applications must be programmed to support all different types of technologies. Thus, for the smart parking system, the web application should be launched and tested on different devices and different web browsers to ensure complete compatibility across technologies. Responsiveness is also one of the components that need to be considered. Some end users may use mobile phones to access the web application externally. In that case, the web application must be able to cater to the users' needs. Thus, responsiveness across multiple devices needs to be implemented and thoroughly tested as well. The web application should retain its readability and accessibility across platforms. An iPad screen view and an iPhone screen view would be different. However, during implementation, the front-end side of the webserver must be designed to change as the platforms of the devices are changing. Google lighthouse can be used for compatibility and accessibility metrics.

With functional testing, the concept of performance testing must be considered as well. Smart Park app will dynamically access the local database and have a feature to showcase real-time video footage of the parking slots for the end-users on a web page. Any real-time application can put stress on the performance metrics. It would require high computational power to be able to constantly communicate with the server and display the video footage to the users. At the same time, load testing is one of the more important concepts as well. UCF being one of the largest schools in Florida, the app potentially would need to support a lot of students accessing it on a daily basis. This would put stress on the performance of the web application due to the huge amount of digital traffic. For the scope of the project, even though the web application would not need compatibility with thousands of students, multiple individuals need to be able to access the web application at the same time. Thus stress load testing must be done on the web application for a number of iterations over the senior design 2 periods. This includes testing the connection between the web application and the users. This would ensure web applications the capability to support a large number of data requests and, at the same time, handle a large amount of data from the database.

Potential vulnerabilities for the web application would need to be thoroughly tested as well. The user login would need to be authenticated for the smart parking application. Features like password strength and password matching while creating and accessing user profiles must be tested on multiple occasions to ensure secure authentication. This needs to be implemented by adding multiple restrictions on passwords while creating a user profile and validation API while trying to access the profile. The restriction could allow standard security measures like having numbers, special characters, and lowercase and uppercase letters. The web application needs to be tested to make sure only correct combinations of usernames and passwords are utilized to access the web application. Right user name and wrong password, wrong user name and right password, wrong user name and wrong password - all three of these scenarios should be checked and tested thoroughly for the

security aspects of the web applications. At the same time, the validation capabilities need to be checked and tested as well. Whether the users can connect and access the website with the correct credentials must be tested on different occasions.

10.2.5 Mobile App Testing

The Smart Parking system is designed to enable the feature of a mobile app. A lot of the new technology corporations are focusing on developing mobile applications along with web application implementations. This caters to different users, is cross-platform, and plays a vital role in increasing digital foot traffic and improving user experience through multiple outlets. However, for the current project scope, keeping workload and time constraints in mind, a mobile application along with a web application could be hindering productivity. Thus the mobile application part of the smart park system is planned to be designed with barebone concepts.

The functional testing would need to be done across the mobile application. Just like web applications, the core functionality is the main aspect of the success metric of mobile applications. Being able to properly handle the functional properties like showing empty parking lot status and the login and logout functionalities. The mobile application would also need to access the local database to showcase the data gathered. The system is planned to be designed to gather data from the computer vision algorithms from the cameras and make an educated decision on detecting vacant parking spots. Later this detection data would be stored in a local database, and the API would need to communicate with the database to ensure data transmission. Later from the frontend stack, the website would access the API through different requests to transmit the necessary data through web pages. This functionality needs to be accurate and must be tested over a number of iterations throughout the period of senior design 2. Since the mobile application is not planned to have as many features as a web application, the testing processes would be less rigorous. However, functional testing has to be done to provide the end-users with the most accurate data.

The web application graphical interface is planned to have an easily maneuverable content interface so that the user experience can be improved to increase digital foot traffic for the system. With the integration of the mobile app and since in today's world, usage of mobile devices is increasing with time, the importance of a robust graphic user interface is immense. As depicted in the Figma snapshot in an earlier section, the mobile app is designed to follow the UCF color theme of black and white. On top of that, the data is set to display on a light blue box for better visibility, and different font colors are set to be used for vacant spots or parking that are not available. To test the user experience of the mobile application's graphics interface, several students would be asked to use and browse through the application and provide feedback on their experience. Improvements can be made to alter the interface based on the test results.

Like the web application, the mobile application would also require flawless data transmission from the local server to the application. So the communication network and efficiency in communication need to be tested. However, the testing procedure would be similar to the web application interface testing. The process would include observing

numerous interactions of communication between the database and mobile application and measuring interrupts while using the mobile application. Notification of the error message system would also need to be implemented for any network outage or server crash situations.

The memory storage for mobile devices is not the same as a desktop application. Thus memory tests need to be done to make sure that the memory usage for the mobile app is optimized. This would include figuring out the core components of the mobile application functionality and only utilizing the components absolutely necessary for the main function of the system. This would ensure memory optimization and metric performance enhancement.

Performance testing is one of the main tests that need to be done for the web application. However, the web application is set to be able to provide a real-time video feed to its web page for the users. However, the mobile application is set to only showcase the number of parking available within the parking slot. Thus the application would not require as in-depth computation power. However, performance testing still needs to be done to ensure correct data is being shown to the users at all times. In addition to that, mobile applications have a new aspect of performance testing. Mobile devices operate on DC power, and depending on the computational operation done on the device; the battery gets drained faster or slower. Thus the mobile application would need to be optimized to preserve battery power. For this aspect of testing, the mobile app is intended to be activated at multiple intervals of time throughout the day and observe the battery drainage metric over time. Also, the loading and start-up time for the mobile application should be tested while the phone is in regular usage as well as in more intensive usage. This is also done to test the response time while a user tries to put in a request to fetch data from the database.

The mobile application needs to be compatible with multiple operating systems and devices. There are a lot of mobile device technologies introduced daily and the mobile application needs to be designed to be supported on any device screen size and device operating systems like android or iOS. Thus the compatibility test should include using the application on numerous devices with different screen sizes and observing the changes. Based on the observation, improvements can be made to integrate the application with the different devices.

The login and sign-up option would be provided in the mobile application; thus, security testing is required. Like the web application, password requirements of lowercase and uppercase letters, number, and special characters need to be implemented for the mobile application sign-up process. Also, the validation process while logged in needs to be tested so that users can log in with the correct credentials. The same three scenarios need to be tested for security purposes - login with the wrong user name and right password, right username and wrong password, wrong password, and wrong username. The mobile application system should reject logging in for all these scenarios.

11.0 Mounting and Installation Procedure

Parking systems require proper positioning and a clear view to be as efficient as possible as well as to serve their purpose to the fullest of their functionalities. Some of the implementations have been discussed in the existing technologies section earlier in this document. Some of the more popular mounting systems include overhead - onto ceiling sensor mounts, on-wall camera mounts, camera and network sensors mounted on lamp posts, and one of the more innovative applications of drilling sensors within the streets or on the pavement for smart detection purposes. These implementations need to be researched and implemented in the most efficient way to gather data or capture video recording with optimum lighting conditions as well as to provide the safest location for the sensors to communicate the data gathered.

Such installations often require extensive measures, such as getting permission from authorities to be able to drill into the pavements or attach additional devices to walls or electric posts around the city, which would also require extra funding set aside just for the mounting procedure. Such costs are usually counted within the total spending budget for the research and development of large corporations. However, for the current Smart Parking Project, such extensive measures for installation and mounting would hinder the development project of the parking system. Thus much caution and creative thinking were needed for the installation plan module for the project.

The Smart Parking system at the University of Central Florida is planned to utilize computer vision algorithms through video cameras to make educated detection of vacant car parking spots from selected parking slots. With the increasing student count within the university as well as students returning to campus after a long remote session of the school curriculum, during certain times of the day - the parking conditions in the parking garages become unbearable - due to the lack of indication systems for the students. Thus Smart Parking system is planned to be designed to provide students with live video feeds of parking situations as well as indications of vacant parking spots utilizing LED signs. Thus the positioning and mounting of the camera equipment are really important.

The camera equipment needs a clear view from a high/elevated position to be able to record the selected parking spots. Also, the luminosity of the location needs to be optimized for smart detecting purposes. Keeping all these requirement categories into consideration, the UCF parking garage C top floor would be utilized to implement, test, and demonstrate the smart parking system. The top floor of the parking garage would provide ample sunlight as well as space to mount camera equipment in an elevated position to be able to capture the recording of the selected parking area.

The smart parking is being designed to use two cameras set in opposite directions to look over approximately ten parking spots. Thus two tripods mounted cameras would be able to capture the spots between them. However, after further testing with the cameras as well as after testing with the tripod positioning, the final numbers of parking spots that are intended to be monitored can be decided. The motivation behind the positioning of the cameras is to be able to capture multiple car positioning from side angles. A depiction of the camera view is added in Figure 41.



Figure 41: Point of View from Camera

If the computer vision algorithm fails to detect the cars from this angle in the garage, there is another angle that is in discussion which would need to be tested. The point of view from the camera from the second angle can be seen in Figure 42.

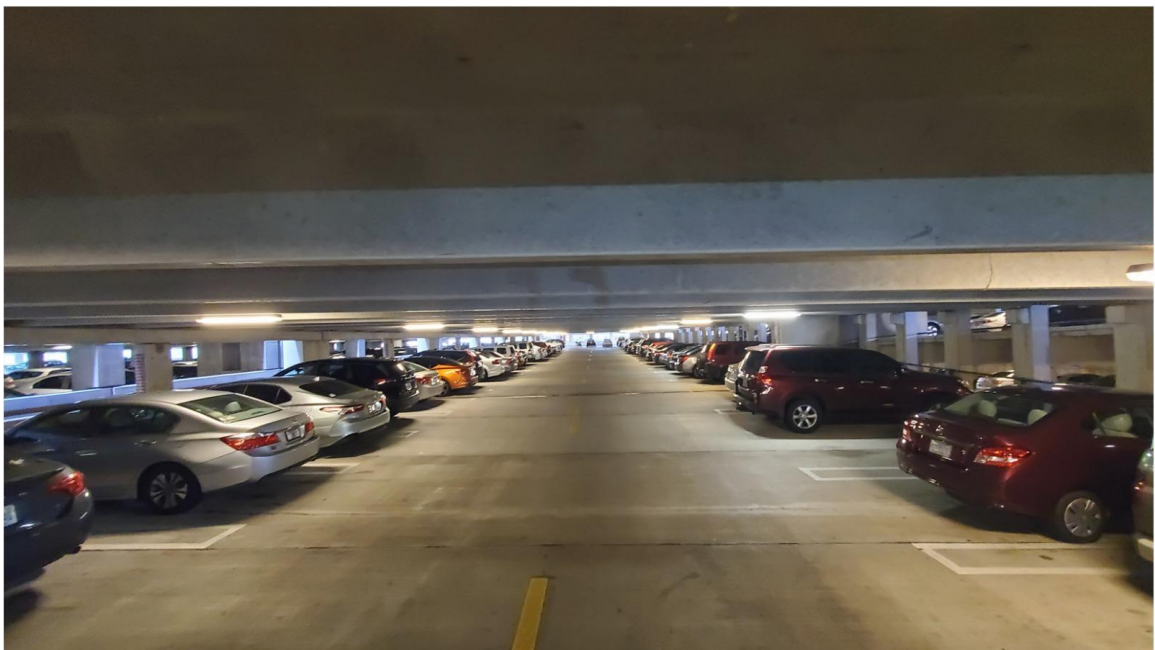


Figure 42: Secondary Point of View from Camera

The smart parking system will utilize the usage of a switch, wifi router, and local server for the main control unit. The control unit is the main component of the smart parking system, which is practically in charge of communicating between the cameras and transmitting data to the web and mobile applications. The database would directly gather its data from within the local server, and the server would also be running algorithms in real-time to detect the parking situation in the allocated parking slots. Due to the importance of the control unit, a VEVOR electric enclosure box, which is waterproof and dustproof to protect the control unit in severe weather conditions. Inside the protection box, plywood can be screwed in, and on the plywood, the switch, wifi router, and the local servers can be set up for steadiness. Ethernet cables need to be connected from the switch to the cameras for data transmission to the local server. The Wifi Router would provide an internet connection for the control unit and enable the communication system.

The box containing the control unit can be mounted on the wall with commercial equipment. However, that would require extra funding with a lot more production time with a lot of paperwork for authorization. However, the scope of the smart parking project is not designed to support such installation within the timeframe and budget. Considering the project circumstances, such extensive mounting is not possible; thus, to support the control unit box, it will be set to the side temporarily for testing for demonstration. Until further advice from the UCF authority, the temporary setup for the control unit box is being considered for the smart park project.

LED signs would be utilized for visualization as well. A lot of parking garages have been implementing the usage of LED signs displaying the number of available parking spots, the garage levels with open parking locations as well as directions indicated with arrows through LED signs. Orlando's Disney Springs is one of the corporations that have utilized this method to lessen the parking difficulties for visitors. Thus to enhance the user experience with the Smart Parking system, LED signs to give directions and indicate the number of spots that would be used. To mount the LED signs, there are a lot of wall mounts available. However, the mounting equipment could exceed the budget for the scope of the intended project. Thus keeping that in mind for demonstration purposes, double-sided heavy-duty adhesive tape would be used to mount the LED signs on the wall. This way the effectiveness of the LED signs can be tested as well as future improvements as far as the mounting and installation can be discussed as well.

12.0 Project Budgeting and Financing

The estimated budget we set for ourselves at the beginning of the semester was about \$1,500. A breakdown of the current budget with the components we plan to use is presented in Table 24 below. All financing will be provided out-of-pocket by the team members.

Table 24: Budget Breakdown

| Item | Description | Qty | Availability | Unit Price | Price |
|--|--|--------|--------------------------------|------------|-----------------|
| Camera | Oak-1-PoE (with 10% discount) | 1 | Available | \$224.10 | \$224.10 |
| LED Displays | RGB LED Matrix Panel - 32x64 | 2 | Available | \$49.95 | \$91.90 |
| Custom PCB | PCB + Components estimate | 1 | TBD | \$100 | \$100 |
| Power Supply (PCB & Displays) | Power Supply 12V/5V (2A) | 3 | Available | \$10.95 | \$32.85 |
| Server Board | ODYSSEY - X86J4125864 | 1 | On backorder (6/3/2022) | \$238.00 | \$238.00 |
| Wifi Router | GL.iNet GL-AR750S-Ext Gigabit | 1 | Available | \$65.61 | \$65.61 |
| 5-Port PoE Switch | STEAMEMO - GPOE204 | 1 | Available | \$33.99 | \$33.99 |
| Heroku | Free Basic plan | 1 | Available | \$0.00 | \$0.00 |
| MongoDB | Free Basic plan | 1 | Available | \$0.00 | \$0.00 |
| Ethernet Cables | 25 feet Cat5e ethernet cable 3-foot patch ethernet cables | 2 3 | Already owned Already owned | \$0.00 | \$0.00 |
| Miscellaneous | Parts | 1 | | \$100.00 | \$100.00 |
| | | | Total | | \$886.45 |

13.0 Project Milestones for Each Semester

13.1 Semester 1 (Senior Design 1)

The major milestones we pursued throughout senior design 1 are shown in Table ... below.

Table 25: Senior Design 1 Milestones

| Week # | Dates (Sunday - Saturday) | Milestones |
|----------------|--------------------------------------|--|
| 1 | 1/9/2022 - 1/15/2022 | Form Project Group |
| 2 | 1/16/2022 - 1/22/2022 | Begin thinking of project ideas to pursue |
| | | Attend SD Bootcamp on Thursday (1/20) |
| 3 | 1/23/2022 - 1/29/2022 | Submit Bootcamp Assignment on Friday (1/28) |
| | | Begin working on DCV1 |
| 4 | 1/30/22 - 2/05/2022 | Finalize DCV1 and submit on Friday (2/4) |
| 5 | 2/06/22 - 2/12/2022 | Attend a meeting with Dr. Richie on Wednesday at 8 AM (2/9) |
| | | Begin working on DCV2 |
| 6 | 2/13/2022 - 2/19/2022 | Finalize DCV2 and submit on Friday (2/18) |
| 7, 8, 9, 10 | 2/20/2022 - 3/19/22 | Begin working on 60 page draft SD1 Documentation |
| 11 | 3/20/2022 - 3/26/2022 | Finalize 60 page draft SD1 Documentation and submit on Friday (3/25) |
| 12 | 3/27/2022 - 4/2/2022 | Receive feedback on 60 page draft SD1 Document |
| | | Begin working on next 40 pages of SD1 Document |
| 13 | 4/3/2022 - 4/9/2022 | Finalize 100 page draft SD1 Documentation and submit on Friday (4/8) |
| 14, 15 | 4/10/2022 - 4/23/2022 | Begin working on the final 20 pages of the SD1 Document |
| 16 | 4/24/2022 - 4/30/2022 | Finalize and submit Final Documentation on |

| | | |
|--|--|----------------|
| | | Tuesday (4/26) |
|--|--|----------------|

13.2 Semester 2 (Senior Design 2)

The major milestones for senior design 2 have been broken down into two phases: the prototype phase and the final product phase. Although the dates and week numbers are left to be determined, the actual milestones have been described, as shown in Table 26 below.

Table 26: Senior Design 2 Milestone

| Week # | Dates (Sunday - Saturday) | Milestone |
|----------------------------|------------------------------|--|
| Prototype Phase | | |
| TBD | TBD | Begin acquiring materials for construction of first prototype |
| TBD | TBD | Construct first prototype and begin testing on hardware and software |
| | | If need be, make revisions to the design, requirement specifications, and functions of our project |
| TBD | TBD | Finalize any software development needed for the final product |
| | | Construct updated prototype and begin final design testing on hardware and software |
| Final Product Phase | | |
| TBD | TBD | Work with the updated prototype to create the final working product |
| TBD | TBD | Work on conference paper |
| | | Make updates to SD1 documentation to finalize SD2 documentation |
| TBD | TBD | Prepare for final presentation (Final Demo, Critical Design Review, Final Presentation PowerPoint) |

| | | |
|--|--|------------------------------|
| | | Present and conclude project |
|--|--|------------------------------|

14.0 Project Management

With such a large project and only a limited amount of time to complete, it is important to discuss the way that we managed our team and project throughout the semester. Project management played a large part in the way our team functioned because it ensured that we were all working well together and were following agreed-upon ways of conducting work and communication. This section will describe the tools we used and the methods we followed throughout our design process that helped contribute to our success.

Proper communication is a vital part of any team's success, and some tools that our team used to help us accomplish this were Discord and Zoom. Discord is a popular VoIP, instant messaging, and distribution platform. This application allows users to create unique text channels and designate the type of information that should be discussed in it in addition to allowing users to enter and exit a voice channel to make for quick voice communication. Discord served as our primary means of communication when it came to sharing documents, organizing team meetings, asking questions, and talking about anything relating to senior design. Doing this allowed us to keep all our conversations in one place and keep a record of anything that was talked about.

The second tool we used for communication, Zoom, was used for any virtual meetings that our team had. Zoom is a web conferencing platform used for both audio and video conferencing. The reason we chose to use Zoom for voice meetings rather than Discord is because Zoom gives you the ability to record meetings. Since there were instances where not every single team member could attend a meeting, Zoom allowed us to all remain on the same page.

Our team met twice a week during the design process on Mondays and Thursdays. On Mondays, we would have a virtual meeting through zoom, and on Thursdays, we would have a physical meeting on campus. In these meetings, we would brainstorm ideas for our project, discuss any upcoming deadlines, line out the expectations expected to be met by the next meeting, and discuss in detail how we design our project.

With four people working on one paper, it is important to make sure the paper reads as if it was written by one person. To accomplish this, a tool that our team chose to use was google docs. While this platform does not have as many functionalities as Microsoft Word, it is free and designed to allow many people to be working on one document. Using Google Docs allowed us to see each member's work as it was being written in real-time since we were all working on a single document instead of several unique copies. Additionally, this platform allowed formatting issues to be alleviated since we could make changes to what another member wrote.

The way we managed our team for senior design one proved to work well, so we will be following a similar structure for senior design two. With this said, our meetings will transition from ones where we are brainstorming ideas to ones where we are prototyping them, producing real deliverables, and having discussions on what we need to do in order to deliver a final product at the end of senior design two.

15.0 Conclusion

The smart parking system that we have presented in this paper would bring all of our team members' skills to test in addition to allowing us to see how all of our skills would complement each other. The implementation of sensors would aid us in learning about embedded system programming along with chip and power design. Implementation of a video camera would broaden our horizon of skills in machine learning and artificial intelligence development. Our website would pose as an effective outlet to show creativity in web development through UI-UX design.

The ongoing implementation of Smart Parking shows that there is a huge possibility of this sector growing from a business perspective, and from a user endpoint of view, it can be observed that the need for a robust smart parking system will only increase with the flow of time. Acquiring data from the current system providers and utilizing their vision as a fracture of this project's motivation, could potentially pave the way for a really promising system that can revolutionize the technologies implemented for smart parking and aid in lessening the daily hassles of urban city parking issues.

PCB design implications along with the communication between the camera and control unit via ethernet bus, make the Smart Parking Project competitive against the current technology. The project brings in possibilities of technological improvements in the firmware side as well as the web development side. Implementation of segment LED signs can also aid in a fail-safe process if a situation of website outage occurs. Throughout the project, various researches and articles have been considered to pick the perfect technology that supports the system. A lot of different technologies were considered and looked into throughout the research process, and in consideration of the developers' skill set as well as to stay relevant with the ever-expanding technologies - the product mechanism choices were made. Various standards and requirements are kept in consideration to meet the safety protocols of the system and the individuals during the implementation and the test process.

With the ambitious range of this project, we hope to grow as engineers as well as provide value to not only UCF but a solution that could potentially be able to be utilized globally. Successful completion of every aspect of this project will open up this significant problem for tons of new solution upgrades in the near future. In the simplest form, the goal of the project is to assist everyone in making a part of their significant lives easier. As engineers, we hope to accomplish this goal and serve the community as well as learn new technologies and become a better version of ourselves in the future.

References

- [1] <https://docs.luxonis.com/en/latest/>
- [2] <https://ams.com/en/belago1.1>
- [3] <https://opencv.org/>
- [4] “The Good and the Bad of Ionic Mobile Development.” Altexsoft.com. Available: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/>
- [5] “Benefits of Ionic Framework in Mobile App Development.” Biz4group.com. Available: <https://www.biz4group.com/blog/benefits-of-ionic-framework-in-mobile-app-development>
- [6] W. Rozwadowski, “Pros & Cons of Flutter Mobile Development.” Futuremind.com. Available: <https://www.futuremind.com/blog/pros-cons-flutter-mobile-development>
- [7] M. Berka, “Flutter Pros & Cons – Should You Use it in Your Project.” Invotech.co. Available: <https://invotech.co/blog/flutter-pros-cons-should-you-use-it-in-your-project/>
- [8] K. Shah, “Advantages and Disadvantages of React Native Development in 2022.” Thidrocktechkno.com. Available: <https://www.thidrocktechkno.com/blog/pros-and-cons-of-react-native-development-in-2021/>
- [9] S. Vidjikan, “Xamarin App Development: Advantages and Disadvantages.” Softjour.com. Available: <https://softjour.com/insights/xamarin-app-development-advantages-and-disadvantages>
- [10] “Xamarin.” Microsoft.com. Available: <https://dotnet.microsoft.com/en-us/apps/xamarin>
- [11] S. Watts, and M. Raza, “SaaS vs PaaS vs IaaS: What’s The Difference & How to Choose.” Bmc.com. Available: <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/>
- [12] <https://newsroom.intel.com/wp-content/uploads/sites/11/2017/08/movidius-myrriad-xvpu-product-brief.pdf>
- [13] <https://www.seeedstudio.com/ODYSSEY-X86J4125864-Win10-Enterprise-Activated-p-4917.html>
- [14] <https://www.sparkfun.com/products/14718>
- [15] https://en.wikipedia.org/wiki/Canny_edge_detector
- [16] <https://www.mygreatlearning.com/blog/introduction-to-edge-detection/>
- [17] <https://stackoverflow.com/questions/45322630/how-to-detect-lines-in-opencv>
- [18] https://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf
- [19] <https://www.ieee.org/about/corporate/governance/p7-8.html>
- [20] <https://www.osha.gov/sites/default/files/publications/osha3075.pdf>
- [21] <https://inst.eecs.berkeley.edu/~ee122/sp07/80211.pdf>
- [22] <https://www.pcmag.com/encyclopedia/term/cellular-mod>
- [23] <https://store.hologram.io/store/nova-global-cellular-modem/36/>
- [24] <https://www.hologram.io/pricing/flexible-data>
- [25] <https://www.quectel.com/company>
- [26] <https://www.verizon.com/internet-devices/>
- [27] https://en.wikipedia.org/wiki/IP_Code
- [28] <https://rainfordsolutions.com/products/ingress-protection-ip-rated-enclosures/ip-enclosure-ratings-standards-explained/>
- [29] <https://ubuntu.com/blog/what-is-an-ubuntu-lts-release>

- [30] <https://www.baesystems.com/en-us/definition/what-are-single-board-computers>
 [31] <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=578032>
 [32] <https://www.protoexpress.com/blog/ipc-j-std-001-standard-soldering-requirements/>
 [33] <https://pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>
 [34] https://en.wikipedia.org/wiki/IEEE_802.3
 [35] [https://en.wikipedia.org/wiki/Power over Ethernet](https://en.wikipedia.org/wiki/Power_over_Ethernet)

Table 4: Camera Comparison

| Model | OAK-1 PoE | OAK-D PoE | OAK-D Pro PoE | ANNKE C800 | RC-810A |
|----------------------------|------------------------|--------------------------|--------------------------|----------------|----------------|
| Type | Microprocessor | Microprocessor | Microprocessor | IP Camera | IP Camera |
| Unit Price | \$249.00 | \$299.00 | \$399.00 | \$79.99 | \$84.99 |
| Size (WxHxD) | 81.9 mm x 81.9 x 31 mm | 130 mm x 65 mm x 29.9 mm | 111 mm x 47 mm x 31.1 mm | 155 mm x 70 mm | 192 mm x 66 mm |
| Video Resolution | 12 mp | 12 mp | 12 mp | 8 mp | 8 mp |
| Speed | 60 fps | 60 fps | 60 fps | 30 fps | 25 fps |
| FOV (degrees) | 81 | 81 | 81 | 102 | 87 |
| Power | PoE 802.3af | PoE 802.3af | PoE 802.3af | PoE 802.3af | PoE 802.3af |
| Weatherproof Rating | IP67 | IP67 | IP67 | IP67 | IP66 |
| Easy Implementation | Yes | Yes | Yes | No | No |