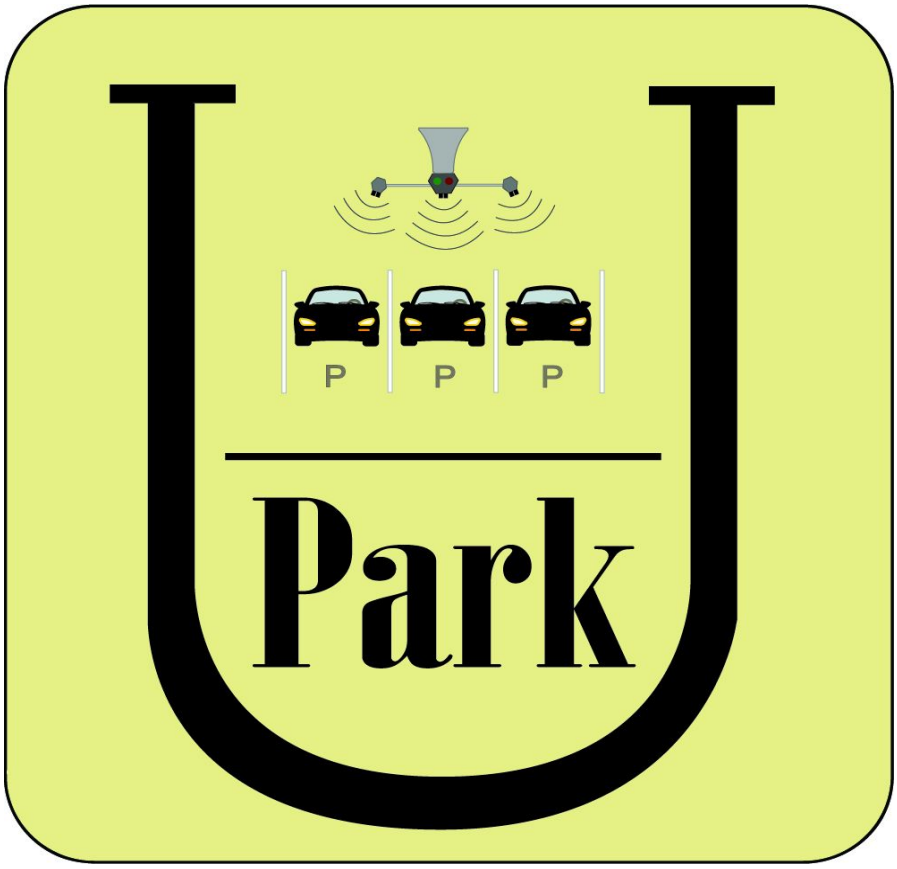
U-Park

Real-Time Parking Information Solution



Spring/Summer 2016 (April,28th 2016)

(Team 9) Member:	**Major:**
Them Le	Electrical Engineer
Danny Russell	Computer Engineer
Carlos Pereda	Computer Engineer
Roddey Smith (PM)	Computer Engineer

Table of Contents

1. Executive Summary

Across the world today there are thousands of parking garages, and there are millions of people who must deal with the frustrations of driving around aimlessly through said parking garages attempting to find a spot. This parking problem is especially apparent to any student who attends classes on the main campus of the University of Central Florida. At the beginning of the fall and spring semesters, this problem only worsens, and often results in driving around in circles for 45 minutes or more. Some people resort to waiting by the stairs in the garages to follow a returning student to their vehicle and take the parking spot as soon as it becomes available. This is not only inefficient, but also results in traffic jams inside the garages which only further exacerbates the stress of finding a spot, and still making it to class on time. This is only one example of inefficient parking garage management, but the problem is prevalent in over-utilized garages all across the country. The U-Park system, which this senior design team will be designing, and testing will aim to alleviate these stresses by supplying users (students) with real-time information of available spots in the garages in which the system is implemented. With students being able to know where spots are available, and which garages have the most spots available, these students will be able to better plan which garage to enter first. Students will be able to spend less time getting to class, and more time being in class.

The proposed solution consists of an array of sensors detecting whether or not there is a car in each parking space in a particular garage. The sensors will be positioned above the cars from the ceiling, so as to have access to the power lines for the lights, and to avoid any tampering, or anybody accidentally driving over one of the sensors. One module will monitor three adjacent spaces using ultrasonic sensors. The modules for the top floor will be slightly different. They will be mounted in front of the parking spots, and they will only detect two cars each. Each of the sensors will point downwards at a space. It will be set to a threshold distance from the ground to be able to detect whether a car has parked in the space or the space is still open. The module on which the three ultrasonic sensors are connected to is based off of the Arduino open-source platform. Instead of using the plug-and-play prototype board, the U-Park solution will simply use a microcontroller chip. The sensors modules will be connected together in a mesh network in order to communicate all of the information for each of the parking spots in the garage down the line to a control module connected to a router. This then sends all of the status information for the spots on each floor to a central database over the internet.

Such a system would be of no use unless users can gain access to this information in an easy and quick way. Since the true problem with the parking situation at UCF, in particular, seems to stem from a lack of information about available parking in the garages. The fix to this problem is to eliminate the

information deficit. Users can access real-time information through a web site which is optimized for mobile use by using dynamically scalable CSS libraries which will optimize the display elements for each screen size. A future goal of the system would be to also develop mobile applications for iOS and Android, which would be able to send users a notification as to what garage to park in, as soon as he/she arrives on campus.

U-Park is a complete solution to a problem which is pervasive in college campuses and large event venues across the country. This system, properly implemented in such locations, will not only save user's time, but will also, in the process, save fuel by minimizing the time driving around searching for parking in a full garage rather than simply driving to another garage with empty spots that is right down the road.

2. Project Description

2. 1 Motivation and Goals

When group nine started brainstorming ideas for senior design project, there were many different ideas from each member in the group. There are three Computer Engineers and one Electrical Engineer in the group, so some of the project ideas were related to computer and some related to electrical engineering. But overall, those ideas were improved solutions or solutions to existing problems that have already been solved efficiently. Even though all the ideas were extremely great, one idea needed to be chosen that all four members would be interested in and able to contribute work to the project. After many group meetings discussing the possible projects, group nine finally came up with the U-PARK system.

The readers may be asking, what exactly is U-PARK, and how does it work? U-PARK refers to the use of sensing devices to determine the occupancy of parking garages. With over sixty thousand students, the University of Central Florida is one of the largest schools in the United State by enrollment. The demand for parking spots has been increasing as the number of enrolled students keep growing. Unfortunately, the current number of parking spots are not enough for all students, leading to the problem that many students are wasting their time driving around looking for a place to park. Students have no knowledge about which parking garage is open or which floor has unoccupied spots. Many students arrive at classes late just because they spend over thirty minutes searching for parking or waiting for other students to get out of class. Even though a possible solution would be to build more parking garages in order to make a more efficient supply to demand ratio, this solution has not solved the problem thus far, and the cost of constant new construction is huge. If students were informed of the location of open spots, the problem could be fixed with a much smaller investment. The cost-effective solution consists of an information

system that provides up to the minute updates to people seeking a spot. In this way, a person who needs to park can determine not only how many spots are available in a particular parking garage, but also know with certainty how many spots are available on a particular floor.

U-PARK is the project that can solves the problem that University of Central Florida's students are facing every day. There will be a lot of benefits that this project can bring if it is successfully implemented. Some of the critical advantages will be discussed in this paragraph. The first benefit is to help students find an open spot faster and provide guidance to optimize the use of existing spaces. All they have to do is to look at their mobile app to get the information about availability and capacity of all parking garages. Secondly, this project helps reduce traffic congestion and therefore improve safety. If students know certain parking garage is full, they will not waste their time to drive there. This will diminish the number of cars in the driveway. Thirdly, all four members in group nine will have the opportunity to apply the knowledge from many classes to this project as well as research and learn more about how to make this project work.

Group nine was excited to take up the challenge. Just thinking about how this project might benefit the school as well as all four members really motivated group nine into researching and planning on how to design the project. In order to start the researching process, group nine divided the tasks to each member in the group depending on each person's interest and major. It turned out that this project is a perfect blend between software and hardware. Group nine will also prototype and research at the same time so that any existing errors will be fixed early. Doing so will help group nine keep everything on track and be able to take control over the project.

Solving the parking problem is not a new subject. It has been a real frustration for many students. Especially in recent years because of the increasing amount of student enrollments. Parking on campus needs improving. This is what makes the U-PARK project is important. Group nine's goal is to implement a better solution than existing ones. The project that group nine will build will not only work for University of Central Florida, but it can also be expanded to a larger scale. The other important goal is to minimize the cost of the whole system so that other customers can see that it is a reasonable cost and worth the money to install the system.

2. 2 Objectives

The primary objectives of the U-Park system are focused around improving the efficiency of parking garages. The problem arose from years of frustration with the garages at UCF in particular, but is rampant at parking garages around the world. There seems to be a lack of information about where there is actual available parking and, when the information is available, it is often wrong and misleading. UCF has a system which has lights or indications on signs at the entrance of the parking garages to tell drivers whether a parking garage is full or whether it has spots available. But this system is often inaccurate that it is more accurate to believe exactly the opposite of what the sign states. U-Park aims to bridge this gap by providing users with up-to-date information about parking by actually giving users the exact number of spots available in a particular garage.

Parking systems in populated areas are only effective if they relay information that is accurate and immediate. For instance, it does users no good to know that a spot was available 5 minutes ago, and if current systems are not able to account for cars pulling out of a spot and immediately having the spot taken by another driver, the system will have such a high level of inaccuracy that the system may as well not even exist at all.

U-Park will be a system which takes into account these intricacies of parking garages, and will provide users with the information that they need, when they need it. By doing this job correctly, the U-Park system will reduce student frustration at UCF by enabling students to get to class faster. And, by avoiding having students/users driving around aimlessly until a driver happens to be pulling out a spot as that student was driving by, and wasting gas. The system will also be beneficial to the environment due to the decrease is fossil fuel emissions. With implementation costs of such parking systems likely being passed on to it's users (in the case of UCF through tuition), students would likely be willing to pay for a system which would save large amounts of headache and would save them money in fuel costs in the long run.

In this document, the U-Park system will be described, and will show how such a system would be implemented. The system is designed around the problems at UCF, but can easily be scaled to meet the needs of parking garages around the world. The document will list the requirements of the U-Park system along with research on how the design team plans on creating the system.

2. 3 Requirement Specifications

The U-Park system's primary mission is to be a cost effective and reliable solution for parking management, while allowing the application's users clear parking availability information. The below requirements reflect this mission and delineate the specific criterion for making this mission a reality.

- The unit cost per sensor module will cost no more than $50.
- Each sensor module will contain sensors to monitor three adjacent parking spots.
- Each sensor module will contain a wireless transceiver to communicate parking availability data to a central hub.
- Data on the user interface will be no more than 3 minutes out of sync with actual parking availability.
- The sensor modules will be able to monitor parking 24 hours a day, 7 days a week and 365 days a year, however will likely only be used for around 16 hours to increase the lifespan of the components.
- Each module will pull no more than 0.5 watts of power.
- The U-Park system will rely solely on the use of standard 110V AC power and will convert to the required DC voltages used in the system.
- The U-Park system will be able to operate in the Florida climate.
- The user interface to check parking availability will allow users to see current parking availability and will have a mobile-friendly interface.

The **Figure 2.1** shows the high level system schematic. The below system is the basic overview which the U-Park design team will be aiming to replicate in the final product.

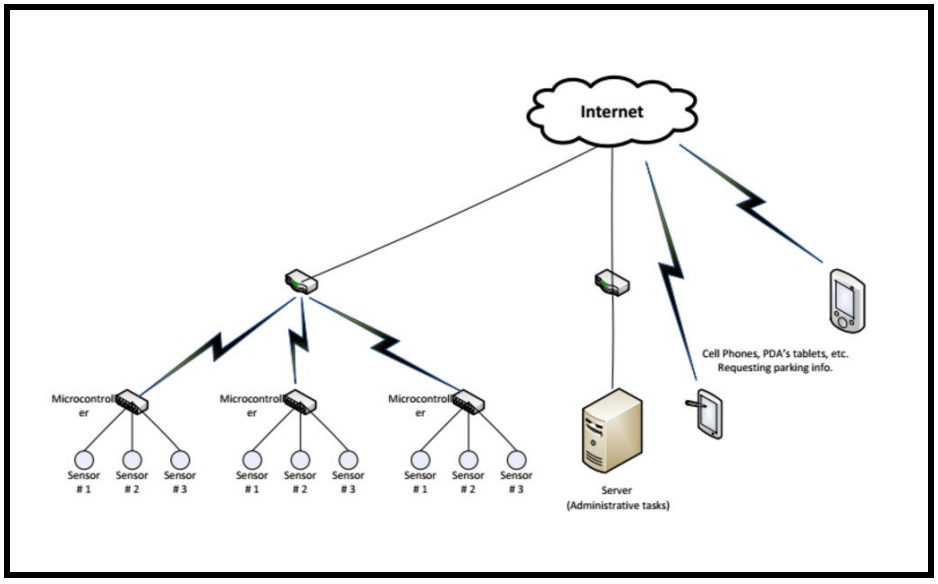


Figure 2.1 - High-Level System Schematic

3. Administrative Content

3. 1 Budget and Financing

The U-Park system is comprised of multiple integrated parts ranging from electrical wiring, sensors, microcontrollers, wireless communication modules, and power supply components. These are used to mount the hardware and case-design hardware. Each sensor module contains three ultrasonic sensors and thereby covers three parking spaces. The three sensors are wired through the mounting hardware to a single microcontroller and power supply. A primary requirement of the U-Park system is that each module will remain as inexpensive as possible in order to stay competitive in the parking sensor market and to reduce the cost of deploying such a system in a parking garage. The system will need a large array of these units to fully implement the design. By keeping the costs for the modules as low as possible, the U-Park system gains a competitive edge on competing products which retail on average about $100.00, and only detect in one parking space. On the next page is a table containing the cost of all of the components needed to build a sensor module. Effectively, this is the cost of sensing three parking spaces.

Table 3.1 - U-Park System Pricing:

Component	Number Required	Component Cost (each)	Total Cost
ATMega 328p-pu	1	$3.38	$3.38
22 pf Capacitor	2	$0.015	$0.03
HC-SR04 Ultrasonic sensor	3	$1.99	$5.97
120V to 12V Transformer	1	$5.00	$5.00
Switching Regulator	1	$2.00	$2.00
LED	2	$0.05	$0.10
16 MHz Crystal Oscillator	1	$0.58	$0.58
Fuse	1	$0.97	$0.97
1N4007 Diode	4	$0.43	$1.72
220 uF Capacitor	2	$0.26	$0.52
10uF Capacitor	1	$0.02	0.02
2.2 kΩ Resistor	2	$0.055	$0.11
10 kΩ Resistor	1	$0.05	$0.05
Wire (misc.)	N/A	$0.50	$0.50
PCB Board	1	$15.00	$15.00
Mounting Hardware	1	$2.00	$2.00
Aluminum Arm	1 x (3ft Section)	$3.15	$3.15
3D Printed Housing	1	$0.00	$0.00
Total:	**26**	**$35.45**	**$41.10**

Total Number of Components:	**26**
Gross Cost: (of Detecting Three Parking Spaces)	**$41.10**

The project is self funded by the development team, and therefore needs to remain cost-effective. The requirement for the cost per module to be less than $50.00 is clearly met as shown by the figures below. The total cost per sensor module is only $41.10. While many senior design projects are funded by external sources and development cost is less of an issue, the U-Park system meets the challenge that all individual test modules must be purchased by the designers. While this can be a problem due to the limited resources of the group and the fact that the development team is made up of students, this also allows for more design freedom. It focuses more attention on keeping the system not only affordable for the future customers, but also sets limitations on the cost for the team itself.

At the quoted cost, the U-Park system remains extremely competitive price-wise. Compared to the cost of other systems, the U-Park system is able to monitor three parking spaces at a cost three times less than other systems which only detect one single parking spot for $100.00 or more. This leads the U-Park system to be around nine times (9x) more cost effective than other products in the same category.

The most costly aspects of the U-Park system are the PCB Board, the Aluminum arm which extends two of the sensors out to the side to detect the adjacent parking spots, and the three HC SR-04 Ultrasonic sensors. The largest cost being the printed circuit board is unavoidable as having the PCB designed and printed is not able to be much cheaper until the boards are put into mass production. Purchasing only three boards at a time from companies such as OSH Park results in a huge markup since the boards must be custom built. If the U-Park system were to be purchased by a customer, there would be cost savings with increased production levels according to the economies of scale.

3. 2 Milestones

The U-Park design team decided from the beginning that sticking to a strict schedule was very important to the success of the final design of the system. For this reason, the team set early deadlines and goals for when parts of the various aspects of the project would be ready for team review. The team worked very hard to consistently meet these goals to ensure that, because everyone on the team is a student, everyone inherently has different schedules and assignments for other classes. Deadlines were set three to four days before official Senior Design deadlines to ensure that in a worst case scenario, additions and changes could be made without putting the team under undue stress at the last minute. Below in **Figure 3.1** a Gantt chart showing the timeline for the U-Park design team is shown. The figure shows the timeline starting at the initial group formation at the beginning of the Senior Design one semester, all the way through until the final demonstration to faculty during the summer of 2016 in Senior Design two.

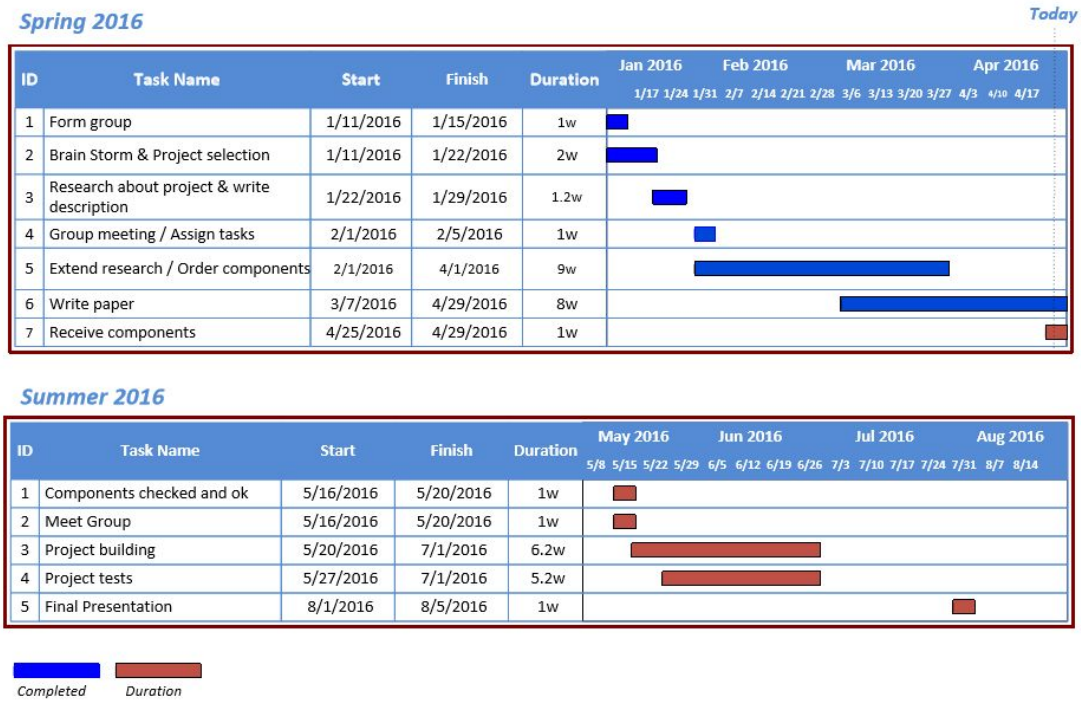


Figure 3.1 - Timeline (Gantt Chart) for the Completion of the U-Park System

3.3 U-Park Design Team

Group nine divided the tasks for each member in the group based on each person's interest and major. **Figure 3.2** is the primary block diagram where it explains specifically the role of each member. Roddey Smith and Them Le are responsible for most of the hardware side, such as the PCP board and sensor. Carlos Pereda is responsible for the database. Danny Russell is responsible for the web application. All the members are required to spend enough time to research on their sections. Whenever someone in the group has a difficult time of understanding something, group nine will have a group meeting to discuss about that problem. Group nine also meets every week or, other week to report on the research status to make sure that everyone is on track.

Figure 3.3 is the input/output of the PCB board. It explains what needs to be connected to the PCB board and the output from the PCB. Basically, this figure is the brief summary of the project about the interaction between hardware and software.

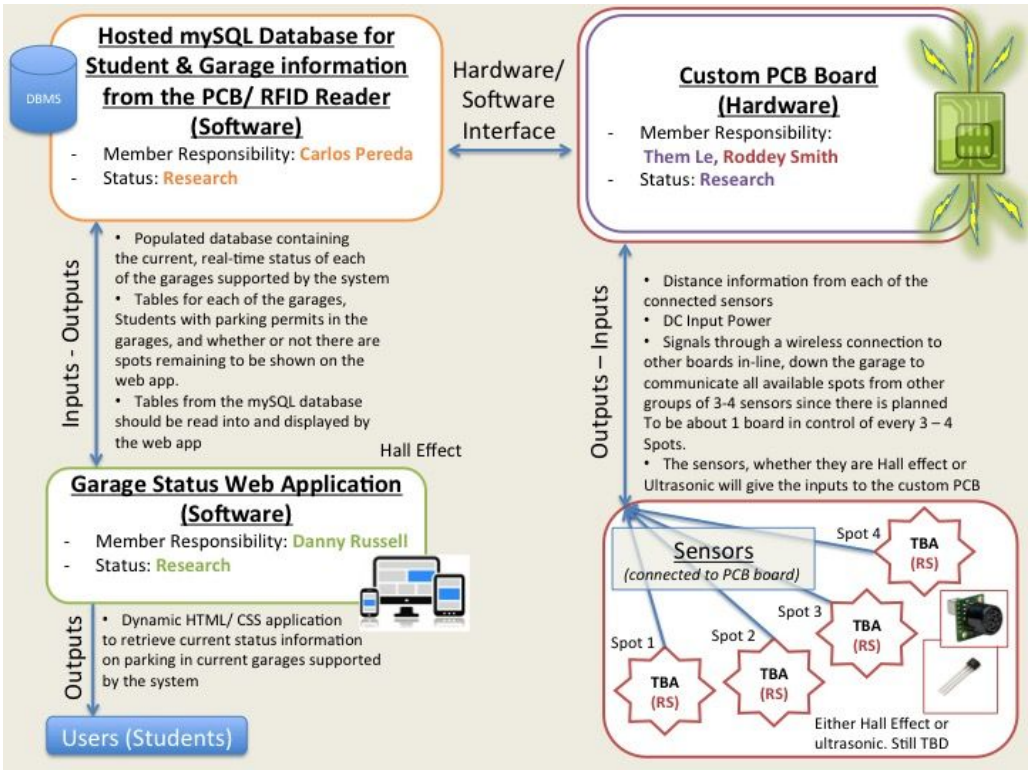


Figure 3.2 - Primary Block Diagram

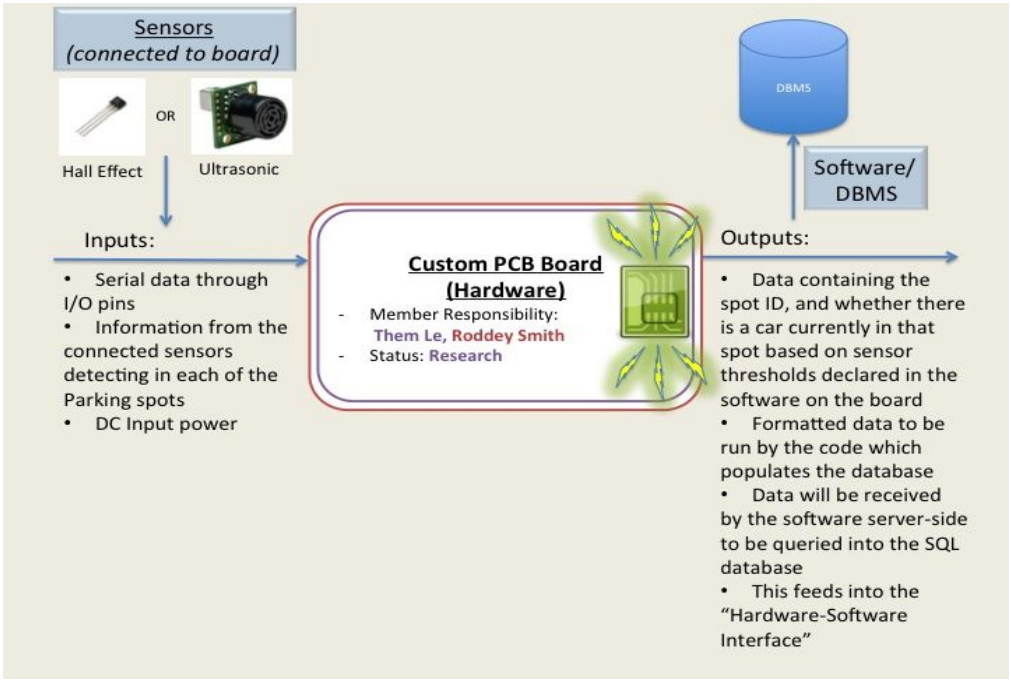


Figure 3.3 - Input/Output for the Custom PCB with Connected Sensor

4. Research

4. 1 Existing Solutions

4. 1. 1 Baltimore/Washington International Airport

The first smart parking garage system in the United States was installed at the Baltimore/Washington International airport. This smart parking system is installed to help travellers find an open spot in a garage that has over 13,200 parking spots. This system uses embedded sensors in each parking spot to detect if the spot is occupied or vacant. As drivers enter the garage, they can view the number of parking spaces available on each floor through displays in the garage. Someone driving into the garage will find messages such as "follow green arrows to vacant spots," as shown below in **Figure 4.1**. This allows people to find spots quickly once they have entered the garage. This is a great design for use in only one parking lot, but if there were multiple unconnected parking garages, it would be difficult to relay the information to drivers about which garages were empty.

Figure 4.1 - Display at Baltimore/Washington International Airport

4. 1. 2 Disney Springs Parking Garage

A similar parking solution can be seen at the parking garage at Disney Springs in Walt Disney World. This system has a sensor above every parking spot that detects if a car is parked there or not. When entering the parking garage, a sign can be seen, this sign gives the amount of spots available on every floor. Each floor has signs that point to how many spots are available in each row, and each spot has a red or green light showing whether or not the parking spot is available. This system is meant to help guests of Disney Springs find a parking spot quickly once they arrive. They do not have any connection to an outside system where people can check the status using a website application. All of their information is displayed using signs in the garage itself.

4. 1. 3 Denmark Automated Parking Garage

A completely different form of parking solutions has been implemented in Denmark. This system does away with the normal parking garage and replaces it with a completely automated system. A person will drive their car onto a specialized pallet inside a specialized parking area. Once they have left the car, the car is automatically taken on the pallet and stored inside a completely automated structure. When someone comes back from their car, the pallet is called back up to allow the person to get their car back. This system catalogues how many cars are in the garage, so if it is full, someone looking for a spot just has to find another garage. There is no need to know what spot to go to since the system automatically places the car in an open spot.

4. 2 Sensors

4. 2. 1 Sensors General Description

A sensor is a device that detects and responds to different types of input. The inputs can be light, heat, pressure, etc. Sensors provide various types of output, but typical use are electrical and optical signals. Sensors are used widely around the world nowadays. The significance of sensor technology is constantly growing that allows more and more sensors to be manufactured. Sensors allow us to monitor our surroundings in ways we could barely imagine a few years ago. New sensor applications are being identified every day which broadens the scope of the technology, and expands its impact on everyday life.

Selecting the correct type of sensor for the project can be an intimidating process. There are literally thousands of models available in the market, so having a good starting point to narrow down the field is essential. In order to decide the appropriate sensor for our project, we have to consider many crucial factors. One of the important factor is the range that it can detect. We want to make sure the sensor that we choose has to be able to read at least a distance

from the ceiling of the parking garage down to the car. Another factor is the working condition (temperature). The sensor has to provide accurate measurement for the environmental condition that it will be subjected to. Power consumption, and lifetime of the sensor is also important because we always want to choose the ones that consume less power, and last longer, but still give us the best quality. Those sensors that last under a year and consume too much power should not be taken into consideration. How fast the sensor would respond to an issued command is also a significant aspect. For example – if a microprocessor is programmed to retrieve information from a pressure sensor every 5 minutes, but it takes the pressure sensor a couple minutes to respond, this would obviously not be a good sensor for the proposed application due to the substantial inaccuracy caused by the time delay. If the sensor does not meet all of the listed requirements above, then it needs to be removed as a viable option.

Out of all the requirements and specifications, cost is one of the most significant aspects. Most of the sensors are not high-priced, but since a sensor is needed in each parking spot, then the cost will be huge if we multiply it out to the whole parking garage. Therefore we need to minimize the cost as little as possible so that the consumer will by the idea to apply the project to real life.

In this section, different types of sensors, and a brief description of the functionality of these different sensors will be discussed. The following will also compare and contrast between sensors, and will conclude with the reasoning behind using the ultrasonic sensor in this application.

4. 2. 2 Case Study: Applicable Sensor Types

4. 2. 2. 1 Hall effect sensor

A Hall Effect sensor is a magnetic field sensor. The basic principle of the Hall Effect is based on a thin metal sheet of semiconductor material that carries the flow of an electrical current. The output connections are perpendicular to the direction of the current. When there is no magnetic field, current distribution is uniform and the potential difference across the output is zero volts as shown in **Figure 4.2**. When a perpendicular magnetic field is present and positioned at a right angle to current flow, a voltage disturbing the current is exerted across the semiconducting metal plate. This potential difference in voltage across the output is termed the Hall voltage as shown in **Figure 4.3**. The Hall voltage is proportional to the vector cross product of the current and the magnetic field.

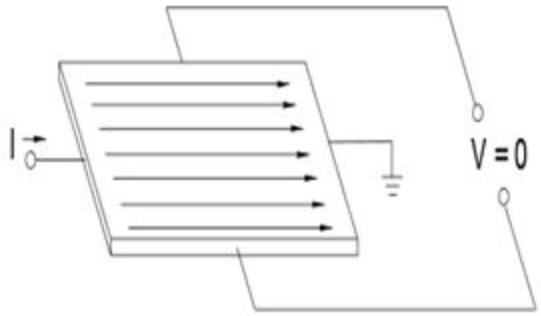


Figure 4.2 - Hall Effect Sensor When no Voltage is Applied

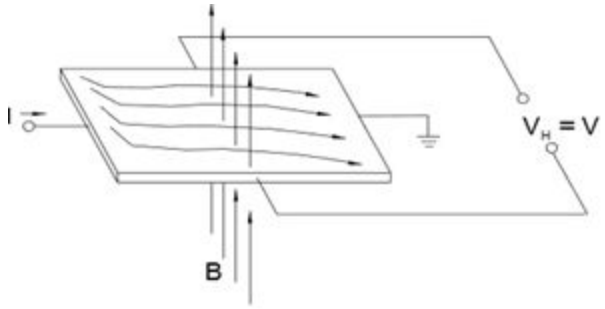


Figure 4.3 - Hall Effect Sensor When Voltage is Applied

A Hall Effect sensor could be used in our project to detect whether the parking spot is occupied or available by sensing the magnetic field around the car. Some features of Hall Effect sensor are listed below in **Table 4.1**

Table 4.1: Hall Effect Sensor Specification:

Parameter	Value
Supply Voltage	3.5 V- 24 V
Supply Current	5 mA
Maximum Switching Frequency	10 KHz
Operating Temperature	-50 to 150 Celsius

Advantages of using Hall Effect sensors are:

- Low noise output
- It does not suffer from contact bounce because a solid state switch with hysteresis is used rather than a mechanical contact.
- Hall Effect sensors are not affected by ambient conditions, such as dust, humidity, and vibrations and are due to are insensitive to some ambient conditions based on the principle that these sensors display a constant flow of an electrical current making their characteristics constant over time.
- Can measure a wide range of magnetic fields
- Hall sensors work in a wide temperature range, provide highly repeatable operation, and are capable of measuring a large current.
- Hall Effect sensors do not have contact with neighboring mechanical parts, making these sensors strong and sensitive enough to detect movement. These sensors do not wear over time thus maintain quality and unlimited use.

Disadvantages of Hall Effect sensors are:

- The Hall Effect sensor is not capable of measuring a current flow at a distance greater than 10 cm
- Hall Effect sensors work on the principle of a magnetic field, making it possible for external magnetic fields to interfere with this and bias the measurement of a current flow.
- Temperature affects the electrical resistance of the element and the mobility of majority carriers and also the sensitivity of Hall Effect sensors.

4. 2. 2. 2 Ultrasonic Sensor

An ultrasonic sensor is also known as a sonar sensor. Its common use is for distance measurement. An ultrasonic sensor has two parts: a transmitter that sends out a signal, and a receiver that reads the signal. Ultrasonic is a sound wave beyond the human ability of 20 KHz. The main idea behind the ultrasonic is that it emits an ultrasonic wave in one direction, and start timing when it is launched. Ultrasonic spread in air, and will return immediately when it encounters obstacle on the way. Finally, the ultrasonic receiver will stop timing when it received the reflected wave. By measuring time lapse and the spread velocity is known (340 m/s), we can calculate the distance between the object and transmitter. Thus, the principle of ultrasonic distance measurement is the same with radar.

Distance measurement formula is expressed as: L= C x T. In this formula, L is the measured distance, C is the ultrasonic spreading velocity in air, and T represents time (T is half the time value from transmitting to receiving). This distance is relevant to our project because it would be able to detect if the parking spot is occupied or available. If the sensor output the signal, but does not

receive any return, then there is no object for the frequency to detect, this parking spot is available. However, if the frequency returns to the receiver, then the parking spot is currently occupied.

An ideal target surface for ultrasonic sensor is hard and smooth because it would reflects a greater amount of signal than a soft and rough surface which is suitable for our project because our target is cars. If the object is small and far away, the sensor will receive a weak echo and this will reduce accuracy. The shorter the distance from the ultrasonic sensor to an object, the stronger the returning echo is. Therefore, as the distance increases, the object requires better reflective characteristics to return a sufficient echo. The sound is a longitudinal wave; therefore, when the obstacle is not perfectly in front of the module, sounds are deflected and echo signal may not reach back the sensor or reach it very attenuated and hence not being detected.

In respect to the proposed project, it is important to analyze the minimum and maximum distances as well as the weather conditions that the sensor can detect. Group nine's research found that the range that the ultrasonic sensor works is between 2 cm-4m. For our project, the average distance from the ceiling to the car is within this range, so the sensor would be adequate to detect a parked car.

As discussed in previous sections, cost is always one of the main concerns. The price is varied in different websites, team nine was able to find sensors as low as $1.10. Even though the price is reasonable, but if this project is applied to the whole parking garage, the total cost will be high. The only problem with this sensor is that it can only detect one car as a time. Multiple sensors would be needed if we want to detect many cars. If this sensor was able to sense many cars as the same time, then the cost can be reduced.

The advantages of using the ultrasonic sensor are:

- An ultrasonic sensor's response is independent upon the surface color or optical reflectivity of the object.
- The response of analog ultrasonic sensors is linear with distance. By interfacing the sensor to an LED display, it is possible to have a visual indication of target distance. This makes ultrasonic sensors ideal for level monitoring or linear motion monitoring applications.
- Stable performance, accurate distance measurement.

The disadvantages of ultrasonic sensor are:

- They need to be mounted in a down-looking configuration as perpendicular as possible to the target,
- Difficulty in identifying lane-straddling vehicles and vehicles traveling side by side, and susceptibility to high wind speeds.
- Targets of low density, like foam and cloth, tend to absorb sound energy; these materials may be difficult to sense at long range.

For this project, team nine has considered the use of a sensor called HC-SR04. A list of the features within the HC-SR04 ultrasonic distance sensor was taken from the product's data sheet with permission from the manufacturer. The list of features is provided in the tab. The dimensions of this sensor is extremely compatible with our design. With the entire detection system being placed on the ceiling, the small dimensions and light weight of the HC-SR04 Ultrasonic Distance Sensor make for a perfect fit within our design.

Table 4.2 - HC-SR04 Ultrasonic Distance Sensor Specifications:

Parameters	Values
Working Voltage	5V DC
Working Current	15mA
Maximum Range	4m
Minimum Range	2cm
Measuring Angle	15 degree
Trigger Input signal	10uS TTL pulse
Echo output Signal	-25 to 125 celsius
Dimension	45*20*15 mm
Operating Temperature	Output TTL level signal, proportional with range
Operating Frequency	40KHz
Resolution	0.3 cm

With the ultrasonic advance and the electronic technology development, the applications of ultrasonic has been increasingly widespread, such as: public security, level detection, parking detection, etc.

4. 2. 2. 3 Sharp Infrared Sensor

Unlike other infrared (IR) sensors, the Sharp IR sensor is a more specialized detector that not only determines if there is an object in range, but it can also measure how far an object is and return an analog value of the distance. There are two major types of Sharp's infrared sensors: analog rangers and digital detectors. Analog ranges provide information about the distance to an object in the ranger's view. Digital detectors provide a digital (high or low) indication of an object at or closer than a predefined distance.

The detector in the Sharp IR sensor is similar to the imaging sensor found in digital cameras. The detector and the IR LED have a fixed distance, the distance of the object will affect the angle at which the light from the IR LED hits the receiver. By looking at where the light hits the detector, it is possible to calculate the angle of the light and from that angle derive the distance to the object.
The **Figure 4.4** below characterizes each sensor by minimum and maximum ranges. Depending on the purpose of each project to choose a specific type of sensor.

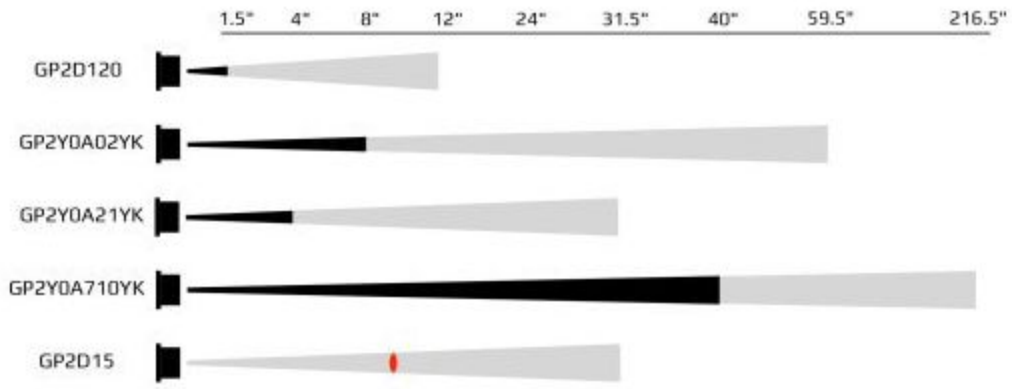


Figure 4.4 - Comparison Chart for the Sharp IR Rangers

Some of the advantages of infrared sensors are that they can be operated during both day and night, and they can be mounted in both side and overhead configurations. Disadvantages are that infrared sensors can be sensitive to inclement weather conditions and ambient light.

Even though Sharp Infrared Sensor can be used to detect whether or not a car is parked in a certain spot, the distance that this sensor can detect does not meet the range that group nine specified. Therefore, group nine considered this type but does not use it.

4. 2. 3 Sensor Decision

There is obviously a wide range of sensor technologies available for vehicle detectors. Some of the most common sensors and how they work were described in the above sections. Each type of sensor has its own specification and work condition. Some of the sensors can only detect an object in a small range while other sensors can detect in an object farther away. There are also many advantages and disadvantages that each type has. After many group meeting discussing about a suitable type of sensor that would work for group nine's project, a decision was made that Ultrasonic sensor is the one. There are many reasons that group nine chose Ultrasonic sensor over other types. These are some of the main reasons. First, the range of the ultrasonic sensor can detect falls into the range that group nine looking for. Second, Ultrasonic sensor not only can operate outside but also can function under various weather condition. Finally, Ultrasonic sensor consumes low power and the cost for each sensor is within the group's budget.

4. 3 Transceiver

4. 3. 1 Bluetooth

The primary viable solution for bluetooth communication using a microcontroller would be the "HC-05 Wireless Bluetooth Transceiver Serial Communication Module" which is shown below in **Figure 4.5.**

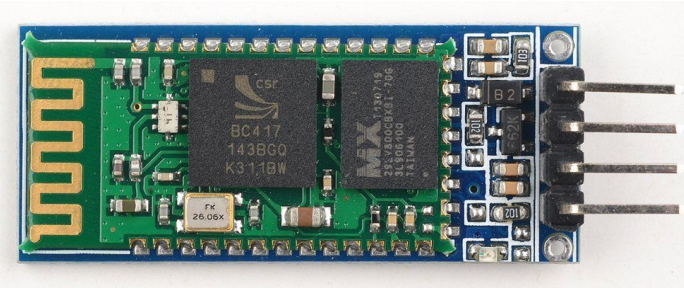


Figure 4.5 - HC-06 Wireless Bluetooth Transceiver Module (purchased/ tested)

This sensor operates within the 83 MHz wide, 2.4 GHz ISM band. This is the same frequency band as Wi-Fi. According to HP, "Bluetooth uses Frequency Hopping Spread Spectrum (FHSS), and is allowed to hop between 79 different 1 MHz-wide channels in this band" (9). This is different from Wi-Fi because it uses *Direct Sequence Spread Spectrum* (DSSS). These are fundamentally the languages used by the transmitters and receivers, to format the way the bits are transmitted over a distance. Bluetooth's method of "Frequency Hopping", is a means to avoid interference from other signals in the same band. If, for instance, there is a competing signal which is interfering with the bluetooth device, the bluetooth device simply jumps to the next available channel until the interference subsides. Unlike Wi-Fi, which remains locked on a 22 MHz-wide band, Bluetooth can "hop" between 79 independent bands, each being 1 MHz wide. This difference can cause problems as 22 of bluetooth's 79 channels will be contained

within the Wi-Fi band. While bluetooth and Wi-Fi have much in common, their delivery method and software interface have different goals and purposes. Bluetooth is based on the IEEE 802.15 standard, and is being further developed by the bluetooth SIG (Special Interest Group) to continue working on efforts to ensure reliability and compatibility with newer systems.

One of the main issues for bluetooth is the presence of Wi-Fi interference and and an inability to develop easy to control mesh networks of devices which would also be able to communicate with the internet. For the U-Park project, one of the fundamental abilities of the system is to communicate whether there is an open spot to the server over a network. While there have been experiments done, and mesh networking software has been made available for bluetooth 4.0 devices and higher. The system would be made more complicated by switching between a bluetooth mesh network for serial communication between the different modules in the garage, and then transferring the data through Wi-Fi modules to the required routers which would communicate with the UCF LAN, and then to the U-Park servers. While such a solution is technically possible, there is a strong possibility of overwhelming interference due to all of the various Wi-Fi and bluetooth signals that would be present.

Pros:

- Simple 4-pin serial tx/rx modules.
- Low power consumption.
- Independent 1 MHz channels .
- Methods such as TDMA (Time Division Multiple Access) to reduce interference from other sources such as Wi-Fi.
- Ease of use.
- The HC-06 has a built-in antenna.

Cons:

- Mesh networking which would be required for U-Park system is in it's infancy .
- Interference issues with wireless LAN's .
- Bluetooth tends to be slightly "glitchy" in practice and the rate of error may be too high for a system such as U-Park which relies on accurate up-to-date parking data to be usable at all.
- Lack of compatibility with Wi-Fi.
- Limited to basic communication such as serial between modules.

The HC-05 module mentioned at the beginning of this section would be the primary candidate if a bluetooth module were to be used for the U-Park System. This board takes an input voltage of 5V and transmits and receives over 3.3V. The baud rate for this board in 9600 by default, with 8 data bits, 1 stop bit, no parity bit, and no handshake protocol. By default the board is set to DATA mode,

where the module is able to transmit and receive data to/from another bluetooth module. Onboard the chip there are 8 Mbit of flash memory.

Since the design team already has experience using the 802.11 standards for setting up networks, and the interference between the required Wi-Fi transceivers and bluetooth transceivers is likely to be a problem, it is very likely that an alternative besides bluetooth 4.0 transceivers will be used to implement the U-Park system. While Bluetooth does exhibit strengths in many applications, this project would likely encounter unneeded hiccups if bluetooth was used, and will likely not be included as the serial communication module for device-to-device communication.

4. 3. 2 Wi-Fi

Wi-Fi is a wireless computer networking technology whose name was coined by the Wi-Fi Alliance, and it's mainly based on the IEEE 802.11 standards. Due to the fact that IEEE does not certify any of its standards, products based on the standard 802.11 were not so operatively compatible. To face that problem a group of technology companies (from which is important to mention Cisco, Harris, Lucent, Apple, Microsoft, among others) decided to form the *Wi-Fi Alliance* by the late 2000. The goal of this alliance is to establish a number of test a product must pass in order to obtain the Wi-Fi certification.

Today, due to its cost, easiness of use, and wide acceptance Wi-Fi is used in airports, colleges, hotels, and any other places where the need to connect into a network (mainly Internet) is a must. Wi-Fi allows multiple connections (mesh) using radio frequency (RF), in the range of both 2.4, 3.6 and 5.0 GHz. The standard 802.11 relates to the physical Layer, and wireless Medium Access Control (MAC) -OSI standards-, and its variants a, b, g, and n will indicate the frequency and the speed of transmissions.

Pros:

- Use of standards.
- Even though prices may be higher compared to other technologies such as Bluetooth, its relationship price/performance makes Wi-Fi the best alternative.
- Wi-Fi distance range can be up to 300 feet indoors and 600 feet outdoors
- Larger distances may be covered by the use of extenders and boosters.
- Wi-Fi speeds range from 11 Mbps (802.11b), to a speed of up to 600 Mbps (802.11n).
- Ability to connect multiple access points reliably in a mesh network

Cons:

- Due to its wireless nature, it's easier to be infiltrated by hackers.

- Speed easy to degrade if other technologies share the same frequencies (such as Bluetooth, etc.)

One of the primary requirements for team nine's project is the ability to establish a reliable wireless communication between the micro-controllers (they control the sensors, that will determine in time, if a parking spot is available or not) and the router (that will in turn send the data collected to the router). The data to be sent by the microcontroller basically consists of two bytes:

Byte 1) The first byte contains the ID of the micro-controller and
Byte 2) Two bits encapsulated into a byte where the status of the three sensors will be reported.

Team 9 has chosen to implement the wireless part of the project using Wi-Fi technology due to the following reasons:

- Distance. The project contemplates to install one or more routers per floor at each and all of the parking garages. They will serve all micro-controllers installed in said floors. At this point it's important to point out that in a real life situation, UCF has already installed Wi-Fi in all parking building, thus eliminating the need for extra routers.
- Standards. The project adheres to the OSI standards, and requires TCP/IP in order to establish a telnet session between the micro-controller and the server (via router). This telnet session, consists of microcontrollers sending data, and server receiving and acknowledging upon receipt.

At this time different boards are being tested, and the most efficient solution will be used in the final deliverable product.

4. 3. 3 Other Notable Options

As an alternative for wireless connections, team 9 has considered the use of wired connections. The option to evaluate is 1000BASE-T also known as Gigabit Ethernet, which is based on the IEEE 802.3ab standard. The reason for considering this alternative are based on pros and cons on the following page.

Pros:

- It increases the distance between the microcontrollers and the router. Team nine estimated a maximum distance of 40 meters using a wireless

system. By adopting 1000BASE-T this distance can be increased to 100 meters (large enough for this project), this is achieved by using Category 5 cable or better.
- Throughput is increased to nearly one gigabit.
- Security:
    - Data transmission is less vulnerable due to RF interference.
    - Access to data is restricted to a wired media.

Cons:

- Price
    - Connecting each and every microcontroller to a router would increase the cost of the project is several times its original cost, due not only to the cost of cable per se, but also the cost of pipes, connecting boxes, clips, etc.
- Maintenance.
    - Fixing a broken connection requires higher costs (tools, more cable, testers, etc.) than fixing a wireless connection.

After debating over cost/performance, it was determined that even though the advantages obtained with wired connections improve the quality of communications, in order to meet our desired budget the project requires the use of wireless technologies.

4. 4 Networking

4. 4. 1 Networking Models

There are many different network configurations that could be used for the wireless communication in the U-Park system. These networking models provide the rules of how each node, and each connection is made to create a network. Each model has its own benefits and shortcomings. The following are the networking models that were considered for the U-Park System.

4. 4. 1. 1 Linear Connection

A linear network consists of nodes that are each connected only to their adjacent nodes. As it can be seen in **Figure 4.6** below, this type of network allows data to move along through each node, one at a time.

The main advantage of this model is that each node is relatively close together. As a wireless model, this would mean that each node would only need to be able to reach its closest neighbors. If this model were used in a garage, the network would be able to snake its way through each parking sensor without having to worry about being blocked by the walls or ceilings. The main disadvantage of this type of network is that if one node, or link goes down; then the entire network (or

a big chunk of it) will be taken offline. Since there is only one route for the data to take, if the chain is broken, no data will be able to flow until the module is repaired. Another disadvantage is that each node must have a large enough memory buffer to transfer the data through the network. As the network grows larger, there is more traffic going through each individual node. There is also a high probability of data collisions if there is too much data flowing through this type of network.

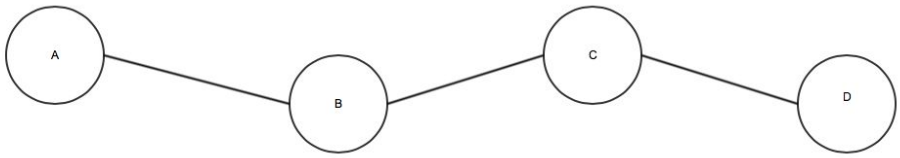


Figure 4.6 - Linear Connection Network Topology

4. 4. 1. 2 Mesh Network Topology

Mesh networks are similar to linear connection networks, except that each node will have a connection with up to four surrounding nodes. The nodes in the corners have two connections while the nodes on the edges have three. This creates a pattern similar to a grid as can be seen in **Figure 4.7** on the next page.

The mesh network has similar advantages as the linear chain except that the data does not have to flow in one direction. This allows for easier routing of data and a smaller chance of collision. A mesh network will still be able to work if one or more of the nodes or connections fails. The data will simply be routed around the failed connection. The disadvantage of a mesh network is creating a routing solution. Each node must know from where to send and receive data. This makes routing much more complicated than if there were only two connections like the linear chain. Mesh networks also require a large number of connections, which adds to this complexity.

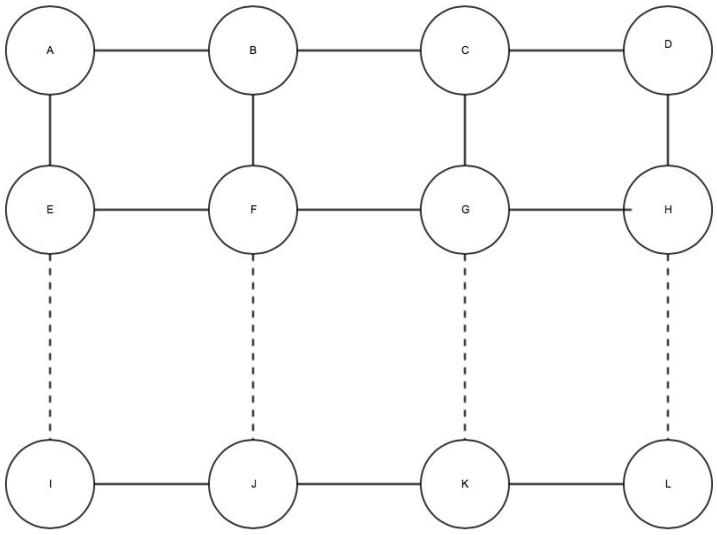


Figure 4.7 - Mesh Network Topology

4. 4. 1. 3 Star Network Topology

The star network, as seen in **Figure 4.8** consists of a central hub that every other node is connected to. Each outside node will send and receive information only through the hub. This allows a network with a minimum number of connections where each can send data.

The main advantage of the star network is that it will continue to work if any of the nodes of connections fail, except for the hub. It is also a good model to use if the communication is mostly one way. If each of the outside nodes is sending and not receiving from the hub, this network can be created easily.

The obvious disadvantage of this network model is that the entire network will go down if the hub fails, but this in turn makes a failure easier to diagnose. The hub also has to be able to handle the data from the entire network. In a wireless environment, such as the U-Park network, the outer nodes must be able to reach the hub. This means that obstacles such as walls and ceilings will come into play.

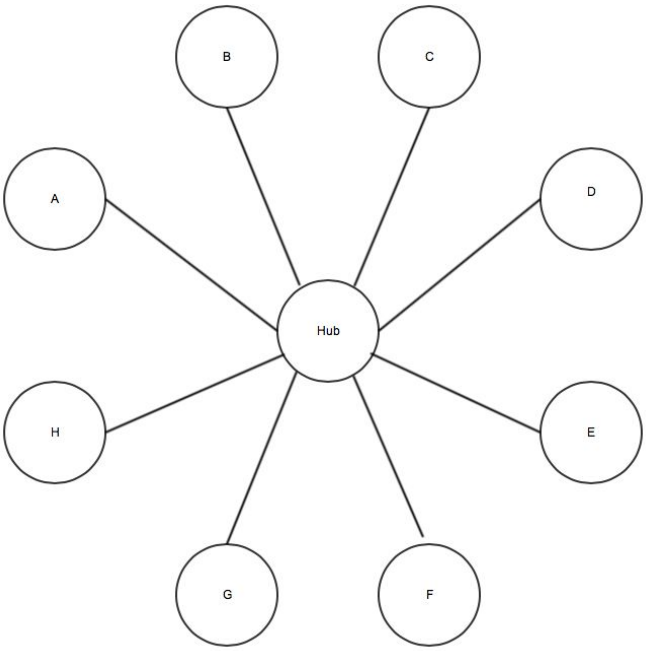


Figure 4.8 - Star Network Topology

4. 4. 1. 4 Ring Network Topology

A ring network is similar to a linear network except that there are no end nodes. Every node on the network is connected to two other nodes. This allows data to move in either direction. This can be be viewed in **Figure 4.9**.

The advantage of this network configuration is that it can withstand the loss of a single node. If one node were to be lost, the data could be routed the other direction to meet its destination. If this were used in the garage, the data would be able to travel through each parking sensor, and it would be able to continue working if one of the nodes went down.

If two nodes that are not connected to each other fail, then the network will fail. This is the main disadvantage of the ring network. It is more reliable than the linear network, but it will still fail with relatively low broken nodes. There is also the same disadvantage as with the linear network of possible collisions.

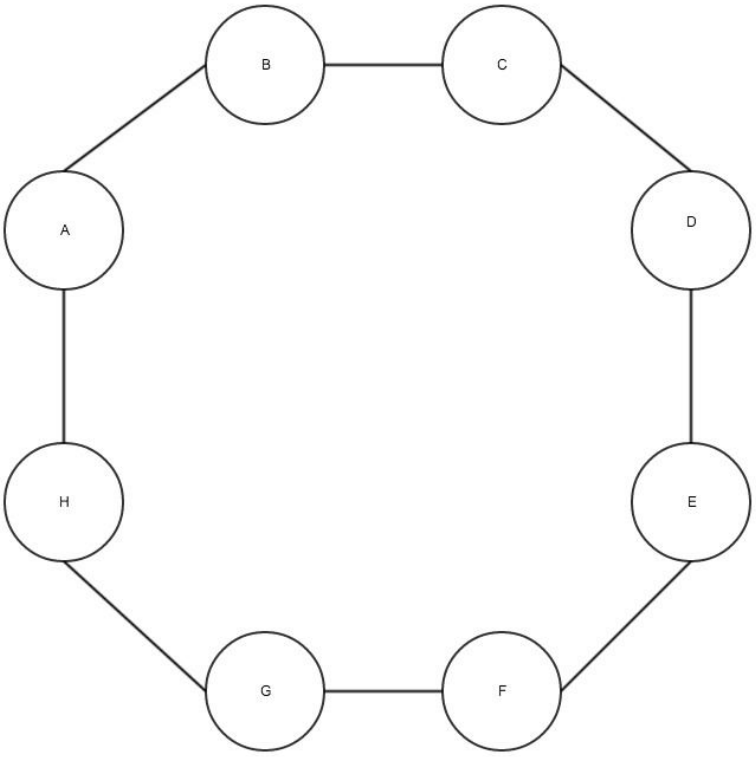


Fig 4.9 - Ring Network Topology

4. 4. 2 U-Park Network

The network model that will be utilized for the U-Park sensor network will be a combination of a wireless star network with a mesh network as depicted in **Figure 4.10**. Each of the sensor modules will use a star network to wirelessly connect to a hub. Since the modules do not need to receive data and only need to send it, it makes sense to use this type of configuration. Each module will continuously update the data to the hub and the hub will relay the information to the surrounding hubs.

The hub network will use a mesh network topology. The hubs will pass the data received from their sensors to each other, and they will then pass that information out to the server. In order to create this network and avoid collisions, the hubs and sensors will have to be placed in a designated order. Each hub will read a distinct signal from each sensor. The signals from each sensor ring network will use the same frequencies. In order to avoid collisions, each of these star networks must be placed so that sensors using the same frequency will not overlap with each other. This will allow the signals to be broadcasted and connect to the correct hub.

Team nine believes that this is the best network to be utilized because it allows each module to only send data in one direction. The modules will use a Wi-Fi

connection with the hub to send the state of each of the sensors. The hubs will be more robust FPGAs with a Wi-Fi and Ethernet connection. This will allow it to accommodate the data that is coming in and route it to the servers and other hubs via either an outside Wi-Fi network or wired network.

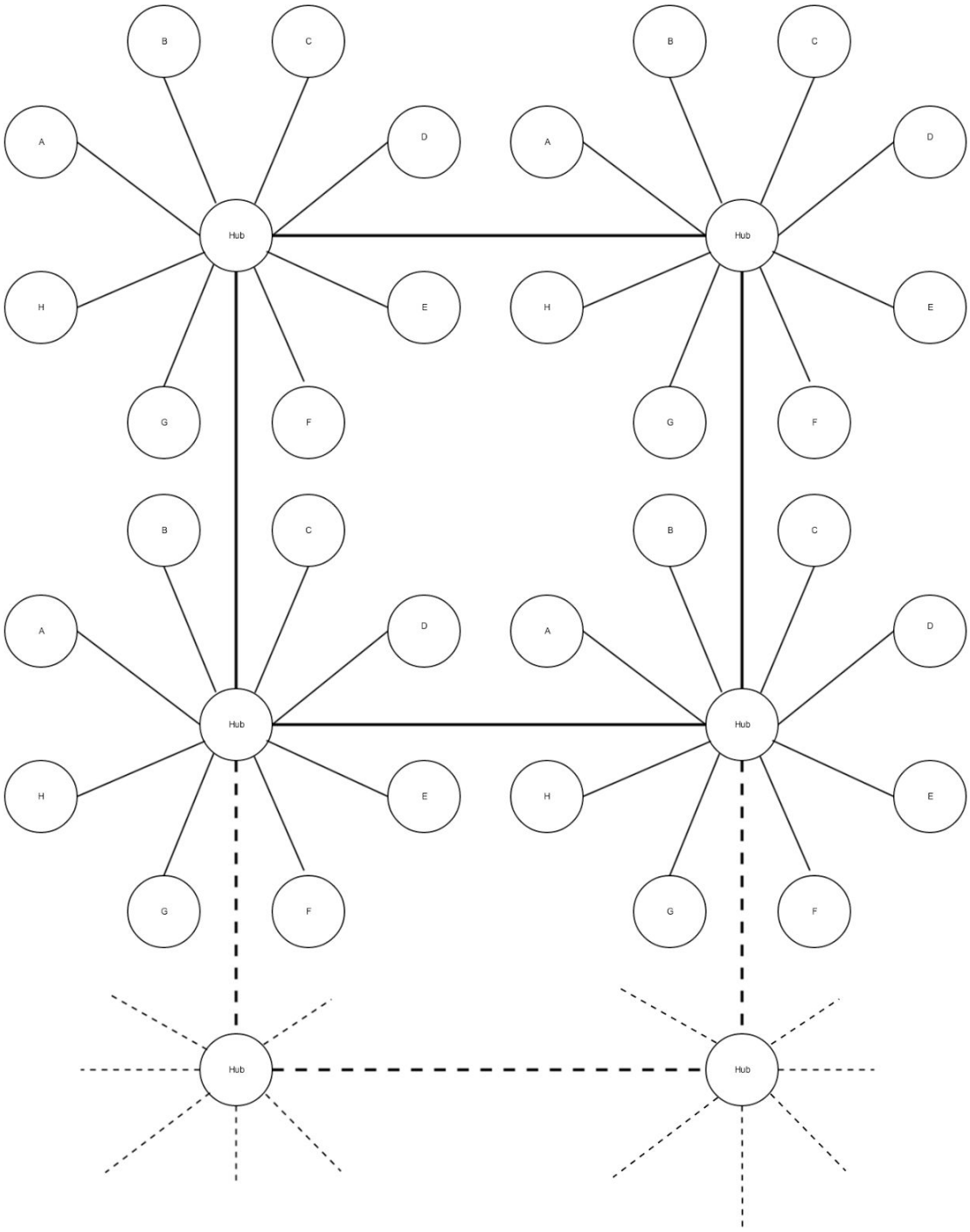


Figure 4.10 - U-Park Network Topology

4. 5 Power

4. 5. 1 AC to DC converter

The microcontroller that team nine will design can operate on an external power supply of 6V to 20 V. Using over 12V, the voltage regulator may overheat, and damage the board. Using 7V, the 5V pin might supply less than its 5V requirement, thus making the board unstable. Therefore, the recommendation is to supply power in the rage between 7V to 12V. The microcontroller can be powered by USB connection, or external power supply.

Powering a microcontroller from a USB is the method used by most people because it is quick and convenient. The advantage is when a microcontroller is connected to a computer to load code, it also provides power for testing. The disadvantage is that this method can only be used for temporary testing because in real life testing, the project is mobile and not in close proximity to a computer, so a different power option will need to be considered.

The microcontroller also can be powered by AC adapters. These are widely used to supply power for small and portable electronic devices. There are some advantages that these AC adapters include:

- Using an AC adapter is fundamentally safer due to its design.
- Heat reduction: A separate power supply removes the sources of heat from the apparatus.
- Replacement: AC adapters are easy to be replaced by the qualified maintenance crew without the need to have the powered device repaired.

While AC adapters are useful for many purposes, there are some problems with these type of power supply:

- Size: Power supplies that plug into the mains directly without using a plug on a cable are bulkier than bare plugs. Sometimes, they are too large to plug into the power socket with restricted space.
- Weight: Some AC adapter are heavy and exert excess weight on the power socket.
- Inefficiency: Some power is wasted when the AC adapter left running while the power equipment switch is off.

Team nine design contemplates to build our own power supply to provide power for many boards. Moreover, team nine also hopes to apply the concepts that all

the members in the group learned from electronics, and power classes to create our own power supply for the board.

External power can come from AC to DC transformer or battery. There are two methods to convert AC to DC that group nine considered:

Method 1)
The first method is to use a transformer (as the one shown in **Figure 4.11** below).

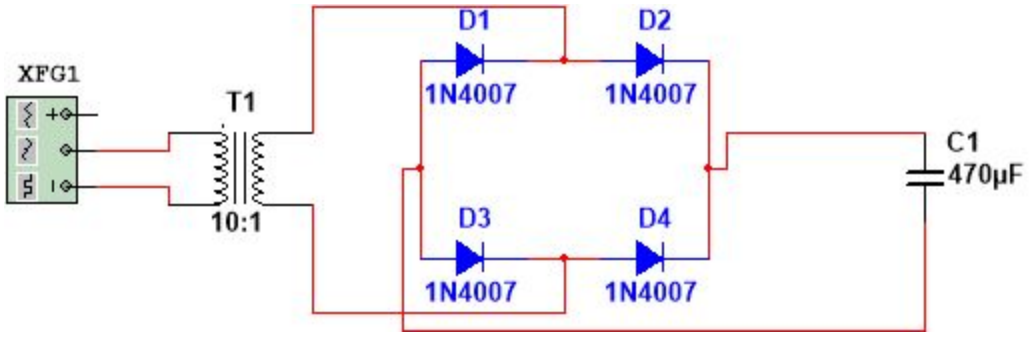


Figure 4.11 - AC to DC with transformer

Transformers are devices that change the voltage of power supplied to meet the individual's needs of power consume. A transformer is based on a basic principle fact about electricity: when a fluctuating electric current flows through a wire, it generates a magnetic field. The strength of magnetism is directly related to the size of the electric current. The larger amount of current, the stronger magnetic field. The primary winding is connected to a 120V AC, 60Hz source. The magnetic field (flux) expands and collapses about the primary winding. The expanding and contracting magnetic field around the primary winding cuts the secondary winding and induces an EMF into the winding. When a circuit is completed between the secondary winding and a load, this voltage causes current to flow.

The voltage may be stepped up or down depending on the number of turns of conductor in the primary and secondary windings. In this design, the transformer step down the voltage from 120V AC to 12V DC. Step down transformer is the one that the secondary windings are fewer than the primary windings. In other words, the transformer's secondary voltage is less than the primary voltage. The ratio of the winding in the primary and secondary is 10:1 because it follow the basic formula of a transformer that is V1/V2= N1/N2. The advantage of using transformer is the ability to transfer power from one circuit to another where power loss is negligible. Power into the transformer is considered equal to power out.

There are many different sizes of transformers that are being sold in the current market. Depending on the needs of customers to pick the right type of transformer. In this project, team nine picked to get 12V DC from 120V AC. However, there is one more factor that the user need to be consider is the current that the electronic component draw in order to choose that appropriate type. The board that team nine designed draws 200mA. In order to accomplish this task, team nine selected the step down transformer that provides 500mA (just as a precaution in case other features are added to the board later on). The cost of these transformers are roughly about five dollars. In real life, designers will use transformers with an extra current capacity.

Method 2)

The second method is to step down the voltage without using the transformer. The schematic is shown in **figure 4.12** below.

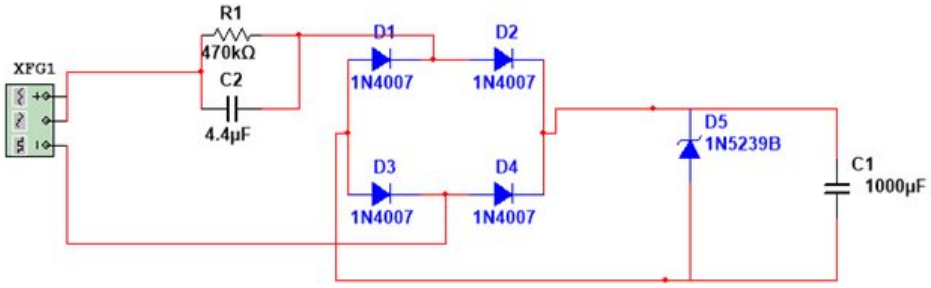


Figure 4.12 - AC to DC Without the use of a Transformer

When a capacitive transformerless power supply is disconnected from the AC mains, there is no guarantee that the capacitor will be in a discharged state. This creates a high voltage safety hazard on a circuit that would normally be presumed as safe when disconnected. To mitigate this concern, a high value resistor is usually placed in parallel with the capacitor. The resistor needs to be rated to handle at least the peak voltage of the circuit.

In both methods, four diodes were used to build a bridge rectifier. The input voltage is a sinusoidal so using bridge rectifier will flips the negative half of the signal up into the positive range. When used in a power supply, the bridge rectifier allows us to convert almost all the incoming AC power to DC. The main advantage of this bridge circuit is that it does not require a special centre tapped transformer, thereby reducing its size and cost.

The four diodes labelled D1 to D4 are arranged in “series pairs” with only two diodes conducting current during each half cycle. During the positive half cycle of the supply, diodes D2 and D3 conduct in series while diodes D1 and D4 are reverse biased and the current flows through the load. During the negative half cycle of the supply, diodes D1 and D4 conduct in series, but diodes D2 and D3

switch “OFF” as they are now reverse biased. The current flowing through the load is the same direction.

In reality, during each half cycle the current flows through two diodes instead of just one so the amplitude of the output voltage is two voltage drops (2 x 0.7 = 1.4V) less than the input V_{MAX} amplitude.

For full-wave rectification to be effective, the rectification must be performed before the Zener diode. This is because the Zener diode will only generate the Zener voltage output whenever a reverse voltage is applied to it. Full wave rectification ensures that Vin is positive, which allows the Zener voltage to be generated. If full-wave rectification were added after the Zener diode, then the negative portion of the AC waveform would simply result in forward conduction through the Zener, which does not generate useful output voltage. Therefore, for full-wave rectification, blocking diodes must always be present before Zener diode.

To make to output of the full-wave rectifier not bumpy, the smoothing capacitor is connected to the output. Now, the output of the power supply will be much smoother. The size of the ripple can be reduced by choosing a larger smoothing capacitor. However, there are two important parameters to consider when choosing a suitable smoothing capacitor, and these are its working voltage which must be higher than the no-load output value of the rectifier, and its capacitance value. This determines the amount of ripple that will appear superimposed on top of the DC voltage.

The 12V, 2W Zener diode (2EZ12D5-TP) and the 1000 μF capacitor are connected in parallel to the circuit to smooth out the output and also bring the output voltage to 12V. In order to choose the correct type of Zener diode, team nine considered many factors. The Zener has to be rated to the power that is larger than what the board consumes so that the Zener will last long, and will not be burnt. Moreover, team nine use the switching regulator that take 12V DC input so the Zener diodes has to be rated to 12V. Zener are very useful; they are widely used, especially in safety circuits where the voltage signal cannot exceed a limit that might damage the circuit. Zener diodes have very high reliability, very sharp reverse characteristic, and low reverse current level.

The advantage of using a bridge rectifier, smoothing capacitor and Zener diode to create a power supply over the use of transformer is cost and size. Each diode costs $0.14 cents and each Zener diode costs $0.22 cents. The total cost of this power supply is about $3.00 dollars compared to a transformer costs $5.00 dollars/each. When building and testing, this power supply will be built in the same board as the microcontroller because the size of this circuit is small. Whereas if the transformer is used, it has to be built in a separate board because

the transformer is big and heavy. The magnetic field around the transformer might affect the microcontroller.

After the 120V AC is stepped down to 12V DC, it needs to be connected to a voltage regulator to give to output voltage of 5V. The reason the switching regulator was chosen instead of the linear regulator (LM7805) is because the linear regulators are only great for powering very low powered devices. Even though they are cheap, easy to use and very popular; the way they operate, makes them inefficient.

The linear regulator works by burning up the difference between the input and output voltage through the release of power by heat transfer. The larger the difference is, the more heat is produced. In most cases, a linear regulator wastes more power stepping down the voltage than it actually ends up delivering to the target device. As a general rule of thumb, if a regulator wastes less than 0.5W then it is considered good. For example, if someone wants to step down the DC voltage from a 12V battery to 5V to power a microcontroller that draws 5mA, and a n ultrasonic sensor that draws 50mA then the power wasted if using LM7805 is (12V-5V)*(0.055A) =0.385 W. The power loss of 0.385W is considered acceptable. The LM7805 can handle this without heatsink. Now if the user wants to add two servos that draw an average of 0.375A that also run off of 5V then the power wasted would be (12V-5V)*(0.055A+0.375A+0.375A)= 5.635 W. The power (5.635 Watts) is lost due to *waste heat*. Without a large heatsink, the LM7805 would get so hot that it could melt the breadboard. With high input voltage, driving loads over 200mA with linear regulator is extremely impractical. The switching regulator would be very useful in this case.

The switching regulator is better choice because it works by taking small chunk of energy, bit by bit, from the input voltage source, and moving them to the output. This is accomplished with the help of an electrical switch and a controller which generates the rate at which energy is transferred to the output. The energy losses involved in moving chunks of energy around is relatively small and the result is that switching regulator typically have 85% efficiency. The switching regulators are usually used in devices like portable phones, video game platform, robots, digital camera, and computers.

The switching regulator that team nine decided to use is TL2575-05. **Figure 4.13** is the schematic of the TL2575-05 and the block diagram is shown in **Figure 4.14**. The input capacitor (electrolytic, Cin>47uF) needs to be located as close as possible to the regulator for the purpose of stability. For operating temperature below -25 Celsius, Cin need to be larger in value. For both loop stability, and filtering of ripple voltage, an output capacitor is required. Also, the diode is placed close to the output to minimize unwanted noise. Schottky diodes have a fast switching speeds and low forward voltage drops, and thus offer the best performance, especially for switching regulators with low output voltages. Proper

inductor selection is the main factor to the performance-switching power-supply design. The type of inductor chosen can have advantages and disadvantages. If high performance is a concern then the more expensive core inductors are the best choice. The inductor never should carry more than its rated current. Doing so may cause the inductor to saturate, in which case the inductance quickly drops, and the inductor looks like a low value resistor.

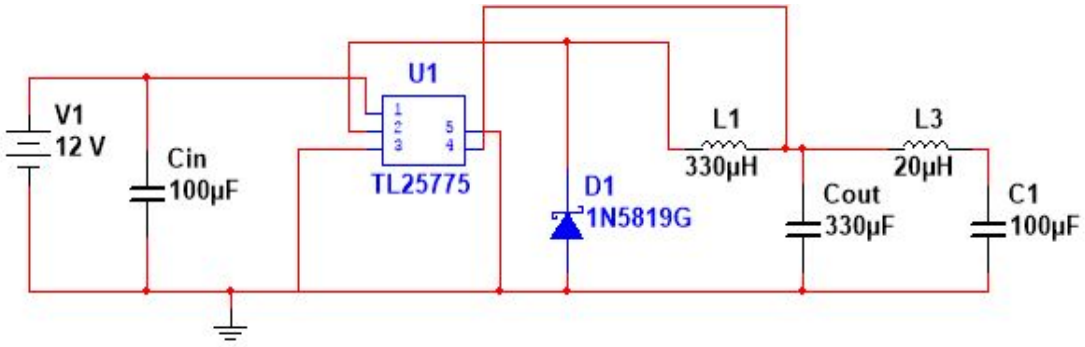


Figure 4.13 - Schematic of Switching Regulator

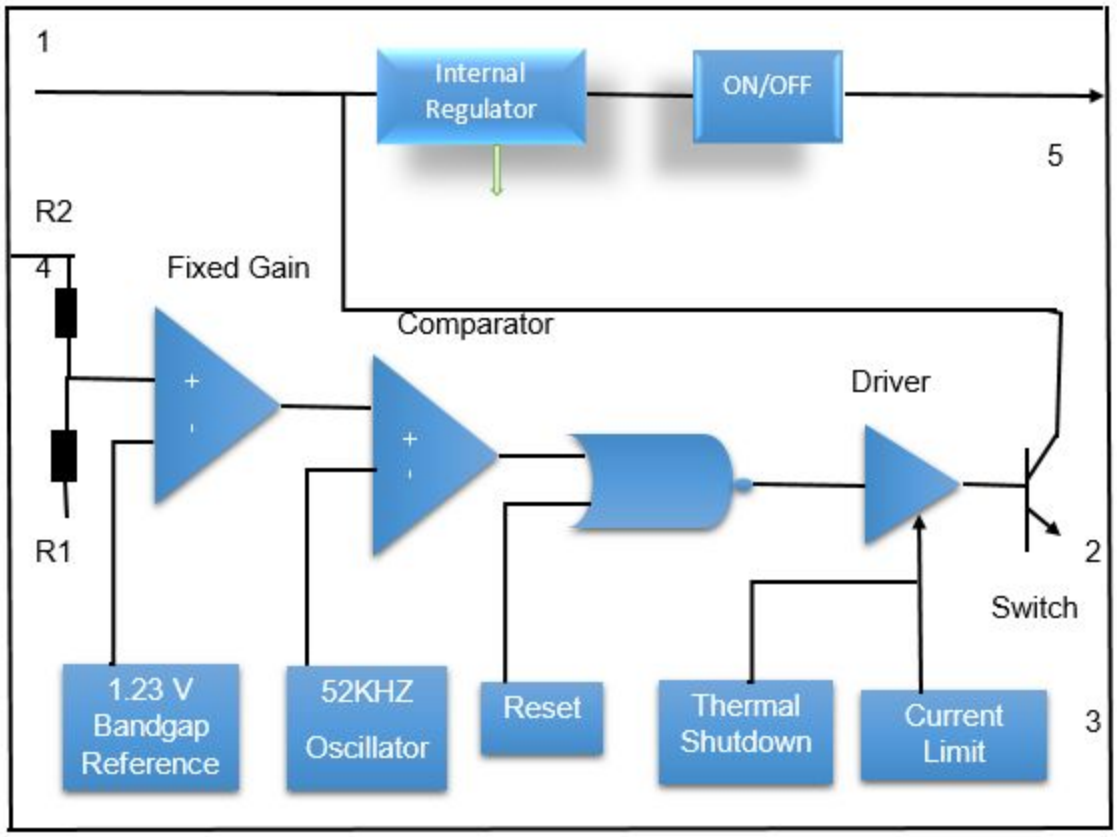


Figure 4.14 - Block Diagram of Switch Regulator

Note:

- 1 is connected to DC input voltage.
- 2 is the output voltage.
- 3 is ground.
- 4 is the feedback.
- Options, feedback need to be connected to Vout. For the adjustable version, feedback must be connected between the two programming resistors R1 and R2.
- 5 is the ON/OFF and must be connected to the ground.

Table 4.3 - Shows the characteristics of a TL2575-05 switch regulator:

Parameter	Values
Input voltage	12V
Output voltage	5V
Efficiency	77%
Output current	1A
Operating Temperature	-40 to 125 Celsius

The TL2575 accepts a wide input-voltage range of up to 60V and available in fixed voltages of 3.3 V, 5 V, 12 V, 15V, or adjustable-output version. The device also offers internal frequency compensation, a fixed-frequency oscillator, cycle-by-cycle current limiting, and thermal shutdown.
Even though team nine can design a power supply for the board transformerless with less cost, team nine did not choose that option. There are many disadvantage that the transformerless power supply has, which are listed below:

- There is no isolation between the high volt AC and the load DC so failure of power supply will destroy the device and also introduce safety issue
- Main voltage is not always steady so this will reflect the output voltage

- If the transformer is not used, it is really dangerous when testing. The board is power from the main source 120V AC, if the users accidently touch any of the component, the will get electronic shot and might cause death.

The following are some advantages of transformerless power supply:

- Significantly smaller in size and weight than the transformer power.
- Supply is easier to come by than transformers.
- Lesser in cost compared to using a transform.

4. 5. 2 DC Power (Batteries)

The board that team nine is designing can also be powered by DC power, which typically is supplied by batteries. Since the board consumes 5V DC, a 5V supply could be obtained from a 9V battery or a set of four AA batteries (1.5V each). Even though a 9V battery seems to be an ideal solution, using this type of battery is inefficiency. The problem is that, in order to obtain 5V from the 9V source, a voltage regulator is needed. The board will draw approximately 200mA, which means 0.8 W is being dissipated by the linear regulator which creates heat which must then be dissipated safely. Furthermore, the typical 9V battery has a relatively low capacity and will drain in a short period of time. If four AA batteries were used, the total voltage that these batteries will provide is 4x1.5V = 6V. Therefore, 0.2W will be burnt in the regulator. While this method is still inefficient, the relative energy dissipated is less than when using a 9V battery.

Table 4.4 - Shows the capacity of standard alkaline batteries:

Battery type	Nominal voltage	Rated capacity	Voltage cut-off	Rated load	Discharge C-rate
9V	9 volts	570mAh	4.8 volts	620 Ohm	0.025
AAA	1.5 volts	1,150mAh	0.8 volts	75 Ohm	0.017
AA	1.5 volts	2,870mAh	0.8 volts	75 Ohm	0.007
C	1.5 volts	7,800mAh	0.8 volts	39 Ohm	0.005
D	1.5 volts	17,000mAh	0.8 volts	39 Ohm	0.0022

A battery can be used to supply power in the testing process. Such as: testing to see if the microcontroller works, testing if the sensors are functioning, or providing the power for the Wi-Fi modules. However, using batteries to power the board is not an efficient method in the long run to implement the U-Park system. The reason is that a lot of power will be wasted through the voltage regulator, and using batteries alone would require much higher maintenance costs as the batteries reached the end of their life. Since the project will be applied to the

whole parking garage, the power supply used for the U-Park system needs to last as long as possible.

4. 6 Software packages

4. 6. 1 Programming Languages

4. 6. 1. 1 AVR-C (Arduino)

A microcontroller is nothing more than a small computer, or a *System On a Chip* (SoC) that typically features a CPU, flash memory, RAM & ROM memory, and programmable I/O peripherals.

The microcontroller selected for this project is the ATmega328P-PU, which according to the manufacturer, contains an eight bit CPU, 32Kb flash memory, 1Kb EEPROM, and 2Kb RAM memory. Microcontrollers are commonly programmed using Assembler, C, or C++ programming languages. The tool chosen by the team to program the microcontroller is AVR-C (a form of C++), for its simplicity of use, and royalty free.

AVR-C consists of a GCC (GNU compiler collection), GNU binary utilities, and the AVR-libc which contains the standard C library. To complete the set of tools for programming the microcontroller, an IDE must be chosen, alternatives to be review are the Arduino IDE, Makefiles, and AVR studio.

4. 6. 1. 2 Java

“Java is a programming language and computing platform first released by Sun Microsystems in 1995” (1). It’s widely used in ATMs, cell phones, laptops, websites, game consoles, embedded systems, and other devices are programmed using Java.

Java provides a multi platform operability (Java code designed for Windows, may be used under IOS, Linux, and other OS) thanks to the use of a Java Virtual Machine (JVM). Java is a free tool; both the Java Runtime Environment (JRE), and the developer (Java SE) are free for downloading from the Oracle website.

Pros of the Java Programming Language:

- One of the most used platforms, constitutes by itself and asset, for it’s quite easy to find staff to maintain the application.
- Royalty free.
- As previously stated, it runs on virtually any platform.
- The IDE contains excellent tools for documentation

Cons of the Java Programming Language:

- GUI for the IDE is not very user-friendly. It is not as easy to program as other tools available, and graphics elements not as 'good looking' as IDEs of other languages.
- The fact that runs on all platforms, makes its more vulnerable to attacks.

4. 6. 1. 3 PHP

PHP is a general purpose scripting language. It is designed to be used for web development, but can also be used as a general purpose programming language. PHP can be embedded in HTML code for a website. This allows the programmer to add dynamic content on their webpage. It is an object-oriented language with each variable declared as objects and assigned a type when they are used instead of when they are declared.

The language PHP is helpful in connecting a website to a database. It allows for the connection to be made as well as make database calls. This allows the website to dynamically display information from a database.

One of the major downsides to PHP is that it can make websites vulnerable to hackers. In order to combat this, many safety updates have been implemented to prevent outside users using PHP calls inside of website for malicious reasons. The programmer must also make sure to close any loopholes that would allow such intrusion.

4. 6. 1. 4 Javascript

As the name implies, JavaScript is a scripting language. It is used by a majority of websites and as such is allowed in almost all web browsers. JavaScript supports object-oriented, imperative, and functional programming styles. On the client side, it is usually implemented as an interpreted language (compiled into an executable before use), but on web browsers, it is often uses just-in-time compilation. This allows it to be portable to almost any system, but it does cause it to run slower than an executable file.

4. 6. 1. 5 Visual Basic

Visual Basic (VB) is considered the flagship language for developing from Microsoft. This easy to learn/use language starts with a versatile IDE which allows development of applications oriented to PC's (forms), standalone applications (Services), web development, and in more recent versions development for the Android platform.

VB is an object oriented language that allows the developer to write programs from drivers (using Windows system calls), up to the most complex database

oriented applications, thanks to the ability to connect to many available DBMS interfaces such as Access, SQL Server, MySQL, SQLite and many others.

Among many programming tools available; all of them with similar characteristics as VB, VB was selected by Team nine for the development for the server interface, due to the fact that is has the best GUI and IDE available today. This GUI not only contains the largest object library (buttons, checkboxes, drop-down list, etc.), but also the GUI 'writes the forms code' as the programmer 'draws' (places objects) onto the screen.

With regards to the aspect of investment security, VB permits the developer to continue using legacy applications, or develop new ones with the latest technologies available in the market. Should the server platform change in the future? Visual Studio's Xamarin permits any Windows based application (VB), be transported to other environments such as IOS, and Android.

4. 6. 1. 6 Bootstrap CSS

Bootstrap is the most popular HTML, CSS, and JS framework for developing mobile first project on the web. It allows the programmer to create web pages that will scale to almost any size screen. It uses programs webpages in HTML. The main benefit of Bootstrap CSS is its portability.

People now use many different devices to connect to the Internet. Bootstrap allows the programmer to create web pages that will look correct on almost all devices. It does this by dynamically scaling the content to the size of the screen. This saves a lot of time in developing web pages by eliminating the need to create different code for each device the web page is viewed on. Bootstrap CSS makes it easy to create web-based applications that can be access on a desktop computer or a cellular phone.

4. 6. 2 Programming Design Patterns

4. 6. 2. 1 Observer

The Observer pattern consists of a subject, an object and its observers. The subject object maintains a list of observers. When there is a state change, the subject automatically notifies the observers by calling methods in their class. This can be seen in **Figure 4.15.** This is a good method for event handling because the subject can register an event that has occurred and it will be updated by all the observers. This provides loose coupling as a change in one class does not ripple effect the others. The downside of this pattern is that subscribers may get unnecessary updates, and it can lead to memory leaks. This pattern could be used in the U-Park system for event handling.

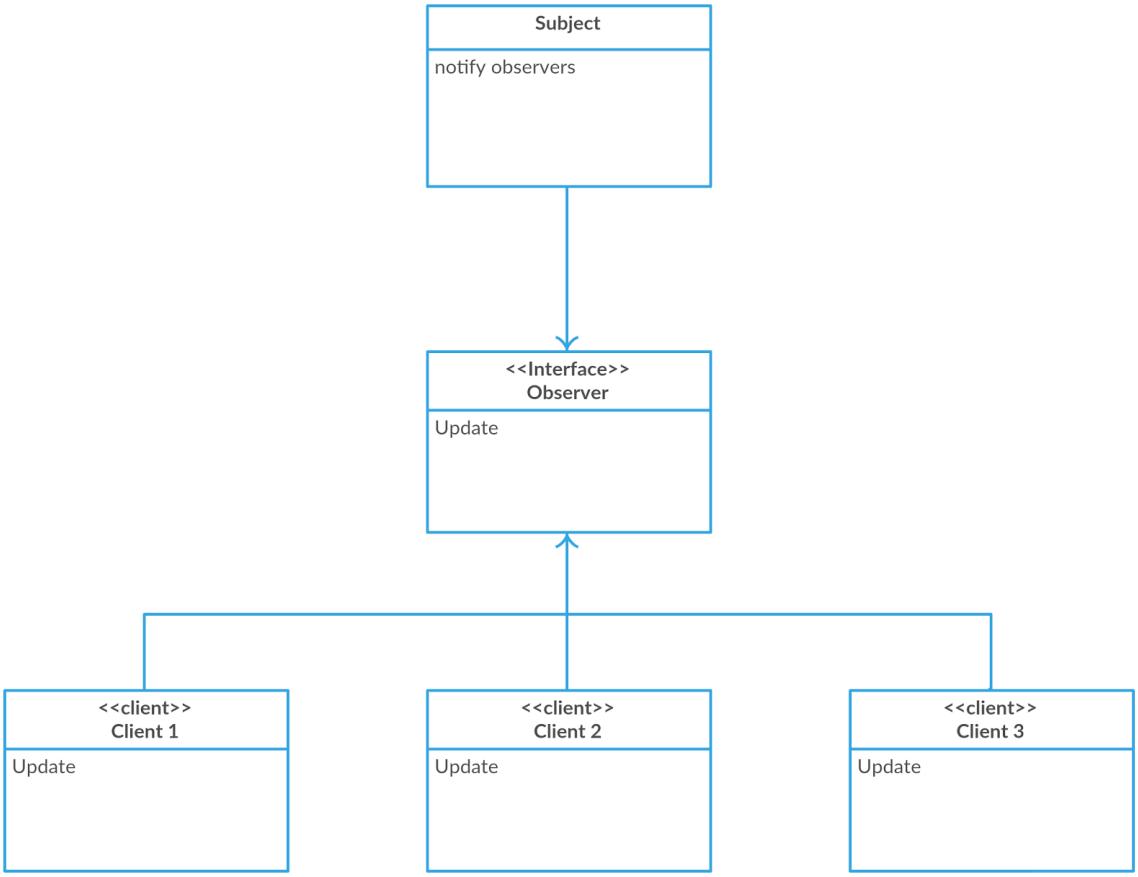


Figure 4.15 - Observer Pattern

4. 6. 2. 2 Factory

The Factory pattern is useful if the programmer needs to create a large number of object and does not know ahead of time how they will act. The factory class produces an object according to the command it receives from the client. An example of this creation can be seen in **Figure 4.16.** This is a good pattern to use in order to centralize class selection and encapsulate object creation. This pattern is not necessary for the U-Park application. The application will only require a few object, and their use will be known ahead of time.

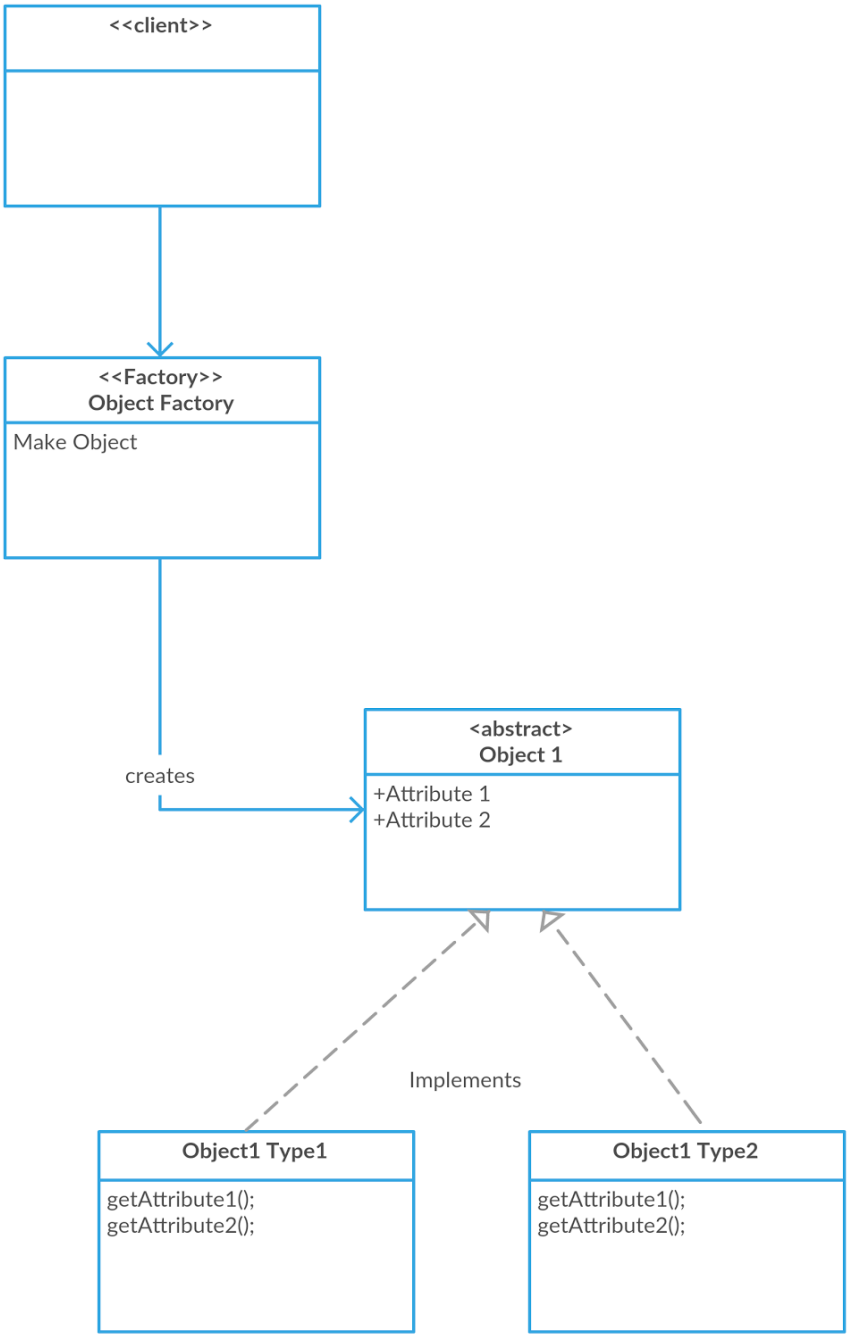


Figure 4.16 - Factory Pattern

4. 6. 2. 3 Singleton

The idea behind the Singleton pattern is to have only one instance of any class running at a given time. This is shown in **Figure 4.17.** This means that only one object of each type can be created. This can be accomplished by marking the class constructor as private, and adding a get_instance function that can only be called once. This pattern works well for embedded operations and other programs where each class has a single purpose.

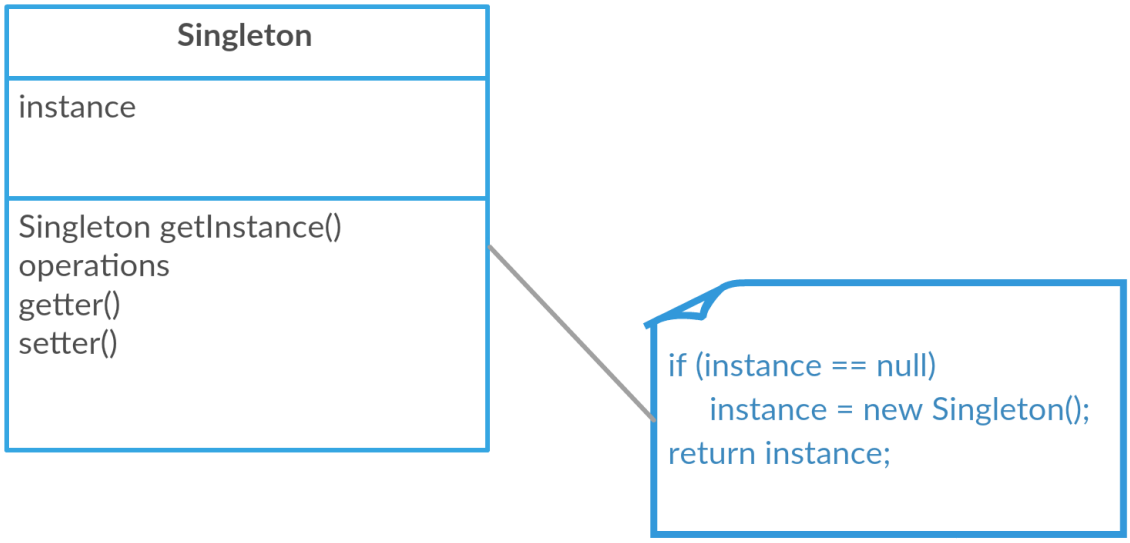


Fig 4.17 - Singleton Pattern

4. 6. 2. 4 Facade

The Facade pattern is based on the idea of hiding any complex actions behind the scenes. The user will see a very simple interface without any access the the complicated functions that are used to create that interface. All of the complex functions are hidden by the facade as shown in **Figure 4.18.** This is a good pattern to use for the U-Park application. The application should be simple to use, and the user should be able to get the information they are looking for as quickly as possible. In order to give this information to the user, a lot of background functions have to be implemented. The user only sees the simple interface without ever having to know what is happening behind the scenes.

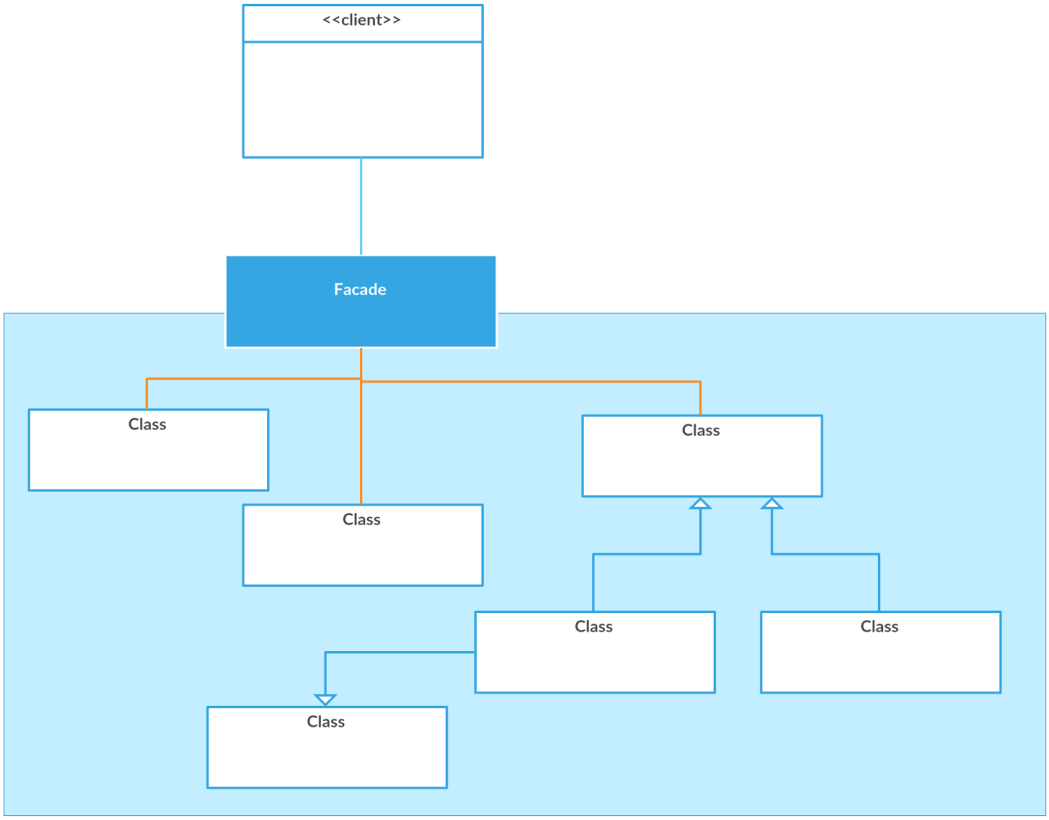


Fig 4.18 - Facade Pattern

4. 6. 3 Database Management System (DBMS)

Besides doing fast calculations, the main function of a computer consists in storing data for later access and use. There has always been discussions about *the best* mechanism for storing and retrieving data; sequential files, index sequential files, and DataBase Management Systems files which can be hierarchy, networked, and relational. Throughout the pass of time, relational models have become a standard, and as prices for software licensing have come down, relational DBMS (RDBMS) can be found from a SOHO network to large corporation mainframes. Its simplicity to design and program, yet its powerful features, makes RDMS a tool of choice for system designers and programmers. "*SQL (Structured Query Language] is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).*

4. 6. 3. 1 MySql

MySQL is a RDBMS (relational database management system) from Oracle Corporation. Developed in 1995, and later acquired by Oracle it's one of the most used RDMS worldwide due to its versatility, cross-platform (both operating

systems, and programming languages), and easiness of use among other characteristics. Facebook, and Youtube can be mention as an example of large customers.

MySQL is distributed in different editions based on the user's needs. Team 9 is evaluating *MySQL Community Edition* which is a free version under a GPL license, and according to the manufacturer's website, it provides the following elements (2):

- Pluggable Storage Engine Architecture
- Multiple Storage Engines:
- InnoDB
- MyISAM
- NDB (MySQL Cluster)
- Memory
- Merge
- Archive
- CSV
- MySQL Replication to improve application performance and scalability
- MySQL Partitioning to improve performance and management of large database applications
- Stored Procedures to improve developer productivity
- Triggers to enforce complex business rules at the database level
- Views to ensure sensitive information is not compromised
- Performance Schema for user/application level monitoring of resource consumption
- Information Schema to provide easy access to metadata
- MySQL Connectors (ODBC, JDBC, .NET, etc) for building applications in multiple languages
- MySQL Workbench for visual modeling, SQL development and administration

From the server side an application written in Visual Basic will interact with MySQL, the mobile app will connect the project database using PHP. In both cases the interfaces are acquired thanks to the use of connectors.

4. 6. 3. 2 MS SQL Server

Just like MySQL, Microsoft SQL Server is an RDBMS based on SQL Codasyl standards. For this project team nine will evaluate the *SQL Server Express Edition,* this due to the fact that is a free product with the same capabilities as the standard edition.

Main components:

- SQL Server Database Engine (Server)
- SQL Service Manager (Server)
- SQL Profiler
- Data Transformation Services (Client)
- SQL Server Enterprise Manager (Client)
- Query Analyzer
- Replication (Server)

The vendor (Microsoft) offers many versions of MS SQL Server each one oriented to a specific necessity. For this project, the version MS SQL Server Express 2014 is the one chosen and being evaluated against other DBMS.

Despite the fact the components presented for MySQL might not exactly match those of SQL Server, both products present similar characteristics –data storage, SQL syntax, replication, backup, etc,.

The pros that could be enumerated for using SQL Server are practically the same as those of MySQL, so in this case attention will be put on the cons that would affect this project. First of all, in a rough estimate of data to store based on a 10,000 car parking, and transactions being made every 30 seconds, the applications would require approximately 20Gb storage per year. This number would exceed the maximum limit of the 'free' version (SQL Server Express), and the project would incur on new expenses not considered for this project.

4. 7 Application

4. 7. 1 Website

The website will be the place that the users will go to view the status of the garages in order to decide the most convenient place to park. The most important aspect of this application is that it will allow users to view a network of garages. For instance, if a user is looking to park in either a city or campus containing multiple garages, they will have instant access to the status of each garage and the floors inside the garages. Users will be able to make a decision quickly based on the status of the garages, and which garages has open spaces closest to the area they would like to get to on campus. This will save people time and help to alleviate the stress of finding a parking spot in an area where garages are spread out.

The application will be scalable to almost any device to make it easy for users to view the website. There are a number of different types of devices that people use to connect to the internet and view web pages or applications. The U-Park application will scale to be usable on almost any size screen using CSS libraries. The interface will allow a user to view the garage status at home from their

computers or tablets and then check for any updates on their smartphones once they arrive at their destination. The web application must be built using a script or language that allows for scalability. This will be one of the most important aspects for choosing the programming design for the website.

The website will be designed for administrators as well as everyday users. The administrator accounts will be created for the “owners” of the garage, such as parking authority at a university. The administrators will be given more access to the garages than the standard users. If an administrator is logged into their account, they can adjust the status of their garages to correct for errors or inconsistencies. They will be able to mark garages closed or reserved. They will also be able to mark certain floors closed or reserved as well. This will allow for the owners of the garages to communicate to all the other users the state of their garages. For example, if there was construction or a special event that would call for the closure or reservation of a garage or certain floors, the administrator will be able to let other users know of the changes.

Standard users will log in the same way as an administrator. They will be able to choose which garages that they want to have saved as their viewable garages. Once this is done, their homepage will show them the parking spots available in the garages they have chosen. They will also be able to expand on a garage and view which floors have free spots. This will be used in the case of a user not wanting to park on a roof or any other floor of a given garage. They will be able to make a quick decision in which garage and on which floor they would like to park.

The website application will need to be connected to a login database on a server. This database will hold the credentials for users to log in. It will also keep track of which accounts are administrators and which are standard users. The passwords will have to be encrypted for the safety of the users. The web application will allow people to create accounts, and will check to ensure that no duplicate accounts are created.

The website will display the current parking spots available on each level of each garage. If a garage is full, the website will display a colored “busyness”indicator. There are times when a garage will be full, but when people are constantly leaving and their parking places being filled by others looking for a parking space. The “busyness” indicator will show how likely a person would be to find a spot in a full garage. The indicator will show if a full garage is more or less likely to have spots open up quickly.

A database will be used in order to keep track of the open parking spaces. This database will be populated by the information from the sensors in the parking garages. The website application will query this database to provide the information to the user. In the case of the administrator, the database will allow the admin to write to the tables to change the status of a garage or a floor.

4. 7. 2 Smartphone Application

While the smartphone application will only be implemented if time permits in senior design two, the information describing the proposed application is listed in the following paragraphs. The smartphone application will eventually have the same functionality as the website application. The smartphone application will give users quick access to the information they need. This will save users time when searching for a parking spot on their phone. The application will be able to be opened directly without having to pull up a website.

This application will remember the user's credentials so that they will not have to type in their username and password every time they want to view the U-Park system. This will save the user time, and it will be much safer if they are using the application while driving.

Once inside the application, the users will be able to view everything as if they were on the website. There will be small differences because the view will be created specifically for cellular phone screens, and the application will also make use of the touch interface used on smartphones.

The smartphone application have the exact functionality as the website. It will query the same database and use the same calls as the website application. Using the same database for both platforms will allow for both the website and smartphone application to display the same data simultaneously as it is updated in the database.

4. 8 Project Scaling

One of the most important aspects of the U-Park system is that it has to be designed with upgradability and scalability in mind. Parking garages come in all kinds of configurations however, to develop a working prototype, the U-Park team has designed its system with special consideration for the parking garages at UCF. Because to U-Park system is not based on UCF specific features though, the system should be able to grow to meet any other parking situation, so long as there is an available network connection to relay the parking information from the sensors.

Each sensor module contains three sensors to detect parking availability in three spots, and each of these modules positioned every three spots communicates with one another in a star-type network. By forming this star network, all of the parking sensor modules need to communicate with router or connection to the internet. This system allows modules to be added as needed and simply assigned an ID number, and connect to the router when added to the other

sensors. This allows the total number of connected devices to only dependent on the number of clients the chosen router can support and the effective range of the router's antenna.

To allow for a greater range on the router both a omni-directional antenna and a patch antenna could be used for diversity. The omni-directional antenna, shown below in **Figure 4.19**, has a much shorter linear range than the omni-directional antenna, but is able to connect to devices, as indicated by the name, 360 degrees around the router. The omnidirectional antenna has a much farther range, but since the grand majority of garages are constructed as rectangles, this may not be the most efficient method for garages. Instead a patch antenna, shown below in **Figure 4.20** can be aimed down a line of receivers and reach the sensor modules at the other end of the parking garage that would otherwise be out of range of an omnidirectional antenna. The caveat of the patch antenna is that it must be aimed in the direction of the clients that it supports, and will not be able to pick up on modules beside or behind the router. This is where the omni-directional antenna comes in to bridge the coverage gap.

Figure 4.19 - Omni-Directional Wi-Fi Antenna

Figure 4.20 - Uni-Directional/Patch Wi-Fi Antenna

For the test implementation and demonstration, only an omni-directional antenna will be used due to cost limitations, and due to the fact that routers do not come with antennas other than omni-directional antennas out of the box. However, to achieve the benefits of having a patch antenna, all that is required is unscrewing one of the standard antennas, and replacing it with a patch. Below there are two figures which show the two different types of antennas that would be used for the U-Park system. This design allows for better scalability of the system since the

length of the garage would be less of a factor, so long as the length of the garage is within the effective range of a path antenna. The range of directional antennas can be as high as a quarter of a mile with a 9 DBi rated patch antenna. The other solution to the problem of implementing the U-Park system in larger parking garages would be to use higher power omni-directional antennas. While this solution “gets the job done”, there also exists the result of having a large amount of signal “bleed-out” past the limits of the garage boundaries, with could result in unwanted interference with external networks.

Connecting all of the devices in an area in a garage to the same router could present issues with an increasing number of clients, but seems to be the best implementation for the U-Park system which is designed primarily around the issues of parking in the UCF parking garages. The U-park system is then able to connect all of the various locations in the garages (the access points) in a mesh network which is described in more detail in section 4.4.1.2. The proposed solution should be more than enough to deal with the specifications of the UCF garages, but is also able to grow to meet the needs of other parking garages, wherever they may be due to the size of the total system being able to be increased by adding a new access point to the mesh network.

4. 9 Microcontroller

4. 9. 1 Processor Speed

The clock used for the ATMega328p-pu microcontroller is a 16.00 MHz crystal. Many times in software a multiplier is put on the raw crystal frequency to allow for different processing speeds. However, the ATMega328p-pu chip actually uses the 16.00 MHz clock frequency as-is. As previously stated, the frequency is often altered for many applications, and the ATMega328p-pu is no exception to this ability. The clock speed is able to be raised for more performance, but there is a substantial degradation in product life, and heat becomes a greater concern as the chip is passively cooled by exposure to its surroundings. More commonly, the clock speed is lowered. One way to accomplish this is through the IDE software for writing the “sketches” (programs), and the base-clock speed can be halved by using the ATMega328p-pu’s internal 8MHz clock.This allows for a reduction in power consumption and slightly improved life-expectancy of the chip. While this does lower performance, lowering the base-clock frequency for the chip can be beneficial in some systems which are reliant on battery power, and benefit from any extra power the system can save through lowered consumption.

For the application of the U-Park system, power consumption is not a primary concern as the power is being received over the AC wiring in the parking garage, and then converted to usable DC in the system. Lowering of the base-clock speed in accomplished by applying a “prescaler” value, and writing code to be included in a header file of the executable program on the board. Applying a

prescaler does cause some headache when writing the code which, since the U-Park is not a low-power application, would add an undue complication to the development process. According to the Arduino website "*when one changes the clock prescaler, time-based methods millis() and delay() become void, since time needs to be multiplied by the prescaler" (10).* For the U-Park system the standalone 328p-pu microcontroller will use the base clock of 16.00 MHz. The clock connected on both leads to the common ground by 22pf ceramic disc bypass capacitors. The pins 9 & 10 are then connected to pins on the ATMega328p-pu microcontroller.

4. 9. 2 Peripherals

The U-Park system is fairly self contained as it requires only two external peripherals to monitor a parking spot. One of the peripherals is the Wi-Fi Module discussed in detail in **Section 4.3.2**, and the other is the set of three ultrasonic sensors which are discussed in more detail in **Section 4.2.2.2**. Other components which could be considered peripherals are the router for Wi-Fi, and the cables to connect the various components together, but since these components are not directly connected to the sensor system, they will be discussed as part of the networking components in other sections in this document.

There are four components which make up the parking sensor system: the microcontroller, power supply, sensor modules, and Wi-Fi module. The Wi-Fi module and sensors will be bought while the microcontroller and power supply system will be built by team nine. This decision was made because building a Wi-Fi module and/or sensor is not only impractical, but also nearly impossible due to the level of intricacy, tiny surface mount components, and the machinery that are required to build these devices. To build and design a router is also incredibly impractical, and would only result in more development cost and headache since routers are already prolific in the market and come with many features above what the team could develop in a short period of time. The U-Park system takes some modules which are available as commercial off-the-shelf products, and integrates these components to an in-house sensor and communication system which is specifically designed for the task at hand,with no more features than are specifically required by the project specifications. In this way the development team is able to streamline the design and ensure compatibility with existing systems, while keeping costs low.

4. 9. 3 Other Microprocessor Information

Over the course of testing and building a stand-alone ATMega328p-pu microcontroller, the team mistakenly connected the microcontroller's clock pins to ground with the incorrect capacitor values. 2.2pf ceramic disk capacitors were

used instead of the required 22 pf capacitors. The Effect of this was that the clock did not work correctly, and would not run the test program to blink the LEDs on pins 12 and 13. This was a learning experience, and showed the effect that the correct values of bypass capacitors have on the valid clock output. The problem that the board was not functioning correctly due to the incorrect capacitor values was difficult to troubleshoot due to the fact that the board was still receiving power and all of the corrections “seemed” correct. Also the capacitors that small do not have an obvious labeling since the ceramic disk area is so small. This was a valuable learning experience as it showed that the rise and fall of voltage on the 16.00 MHz clock is very reliant on specific capacitors, whereas other circuits may not be as sensitive to minor changes like this.

4. 9. 4 Part Selection

Components used in the development of the U-Park system were chosen based on price, part availability, use in the marketplace and whether there was good documentation to support in the integration of the components into the system. Also, another reason some parts were chosen such as the Wi-Fi communication module, was from testing. The team discovered that some Wi-Fi modules compatible with the microcontroller did not contain integrated antennas, and an external antenna had to be used. While this was an easy fix, The addition of having to add an external antenna was an easily avoidable issue by simply using a WIFI module with an integrated antenna.

For the router, selection was made by using an unused router a team member already had. While a fully-implemented system would require a router with much higher specifications, and would require external antennas, This was not implemented in the final senior design deliverable because the router is one of the items that can be replaced as needed depending on the physical requirements of the garage which the system is installed in. Also, the router is an independent component of the system and other components do not “care” which router is used to relay the sensor data to the network.

Other parts such as the individual capacitors, resistors, etc. were chosen based off of availability and price off of various websites. The basic building blocks of the system did not have any special requirements and therefore did not require special consideration in the selection process.

The ATMega328p-pu microcontroller was selected due to its ease of development, and its open-source hardware and software architectures. There is an exorbitant amount of documentation available for all compatible modules such as the sensors, Wi-Fi modules, and the microcontroller itself, which aids in development, and minimizes the amount of time it takes to troubleshoot errors, and reduces the overall development time and demand on the team, which may have little prior experience working with similar embedded applications.

4. 10 Summary of Features

This section will explain the U-Park system as a whole from a higher-level perspective. It will describe the system's primary features and will cover how users will interact with the features of the U-Park system, and how the users will be able to be disconnected from the hardware, so that the only features the user will have to be concerned will have to do with the web interface.

The hardware for the sensor modules is reliant on AC power, and is therefore mounted on the ceiling of the garage. However, this presents a problem for the sensor modules on the top floor of the garage where there is no roof. In this case the modules will be mounted on the wall in front of the parking spaces, and power will be run between the modules by a separate wire. On the main floors of the garage where there is a ceiling, vehicles have a restricted height requirement to be able to enter the garage, and therefore the implemented system does not have to be overly restrictive on the amount that the module extends not from the ceiling. The parking garage ceiling has a structure where "ribs" of concrete are around a foot and a half lower than the actual flat portion of the ceiling. This is where the sensor modules will be mounted. This puts the system well out of the way of where users will come into contact with the modules and makes tampering with the modules much more unlikely.

On the top level of the garages, where the sensor modules will be mounted on the walls in front of the parking spaces, the modules are mounted in between sets of two spaces. This ensures that even if a car decides to pull into a spot extremely close to the wall, the sensor will not be hit by the car or the car's bumper. This does present the issue that the sensor module will be more able to be tampered with, however, because the system will be enclosed inside of a hard 3D printed shell, the sensors will be protected from a grand majority of would-be vandals. The team is also relying on the system being useful and provide useful data to prevent users from wanting to vandalize the system in the first place. After all, who would want to harm a system that is useful and helps users with the rampant parking problems presented in parking garages, not only at UCF but all around the country.

The sensor modules are connected to a router wirelessly through Wi-Fi modules. Each router is responsible for only a small number of modules in a area surrounding that router. The routers "talk" to each other in a mesh network configuration. This allows the system to be expanded to cover larger garages, and ensures that the system is not overloaded with too many clients on each of the access points.

The user interface is the only aspect of the U-Park system that is pertinent to its end-user. The user for the system would initially presumably be a student at UCF. The student will simply be able to open a browser and access the web site, which displays parking information for a garage. This web interface is designed and built using Bootstrap CSS libraries to ensure that the content displayed is optimized for any screen size. Whether the student is on a laptop and want to check before they leave for class if a garage has available parking, or if the student is already on campus and wants to see which garage currently has the most available parking by the building that the student need to go to. The user will be able to see real-time information on where to park based off of current availability in the garages monitored by the U-Park system. This makes driving into a garage to search for parking a thing of the past, and allows drivers to save not only time when searching for parking, but also gas as the user does not drive around aimlessly looking for a spot. This will save the user money and will help lower the amount of greenhouse gas emissions from cars driving around in full garages.

The web interface will be simple and uncluttered, only supplying the users with the garage information that they need. This will improve the response time of the web page as well and avoiding unnecessary distractions from irrelevant information. At a glance, users of the U-Park system will be able to see all current information on the garage availability and will be able to make a decision on where to park based on that information.

5. Methods

5. 1 Research Methods

Research for the U-Park system done by the design team was an iterative process. In principle, research for intricate projects such as these can not simply be confined to the initial design period. All throughout the design process, specifications were created, and new considerations had to be thought about. Even initial design ideas had to be reconsidered once new information came to light.

For instance, the team considered implementing the parking management system through UHF RFID tags, where traffic and parking passes would be monitored by keeping track of these RFID tags. However, the team realized that there would be significant privacy and ethical concerns with implementing such a system in a school setting. Also, there would exist the problem of drivers entering the garage without an RFID tag. This problem would require the creation of a separate system to take pictures of someone’s licence plate who entered without a RFID hang-tag or sticker. This, among other computational challenges, presented more ethical issues. Because of these hurdles and ethical issues, the

team then decided that creating a system to just detect the individual spaces directly would be a better solution.

Research on components and tutorials of how various aspects of the project will continue to be sought-out until the system is fully fleshed-out. Even after the design phase, as small problems may come to light, small changes will be made to improve the system. There is always a need for continuing exploration of new ways to do things to ensure the product keeps improving. **Figure 5.1** below shows a visual representation of the research process the U-Park design team. It shows how the process is iterative, and how the process is independent of the point the team is in the design process.

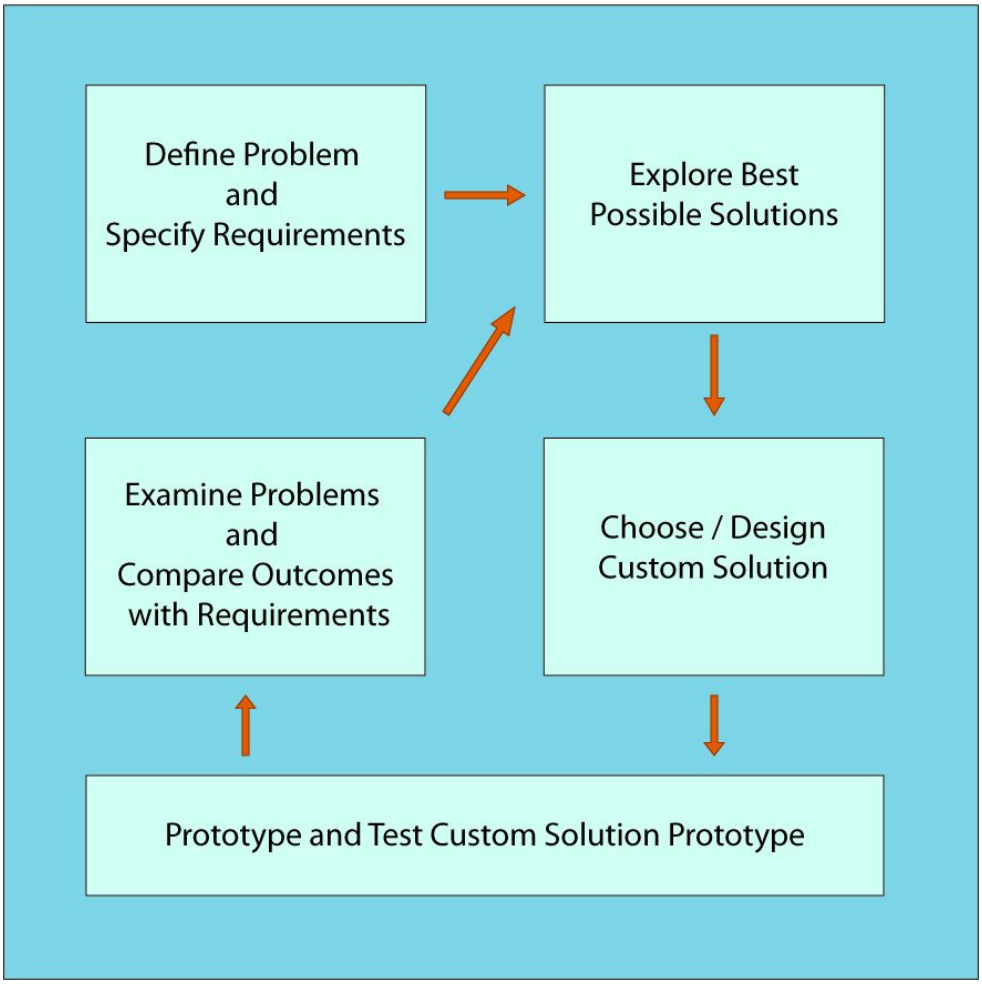


Figure 5.1 - U-Park Team Research Method

5. 2 Design Methods

The Process through which the U-Park system has been created is similar to the methods used for research shown above in the research methods section (**Section 5.1**). The design method was also an iterative process where, during examining possible solutions for the given problem, caveats to the proposed

solutions were examined. However, during the design process, input from mentors and Dr. Richie were also taken into account.

For example, the team was initially decided on using a simple 5V DC linear voltage regulator. It was brought to the design team's attention that this solution would work, but would not be an elegant solution for the problem, in that linear voltage regulators are extremely inefficient. In linear voltage regulators, the voltage difference between the input voltage and the steady 5V output is dissipated through heat transfer. Instead of using a linear regulator the team sought out help from Professor Reza Abdolovand, who suggested using a switching regulator, which does not suffer from this lack of efficiency since it basically switches off and on quickly to only build up the required voltage output required and then shuts off through a feedback system. This process repeats itself to produce a constant output voltage without having to dissipate the excess energy through the transfer of heat.

While using the switching regulator is more efficient, using a linear regulator accomplishes the task of voltage regulation well enough for the U-Park system. Because using the linear regulator saves space inside the sensor module housing, it was chosen by the design team to be the best solution, as efficiency is not much of a concern. However, the design process benefitted greatly from examining the caveats of using a linear regulator, and allowed the team to weigh the pros and cons of different options more thoroughly. The design pattern the U-Park design team followed in the development process is shown below in **Figure 5.2**.

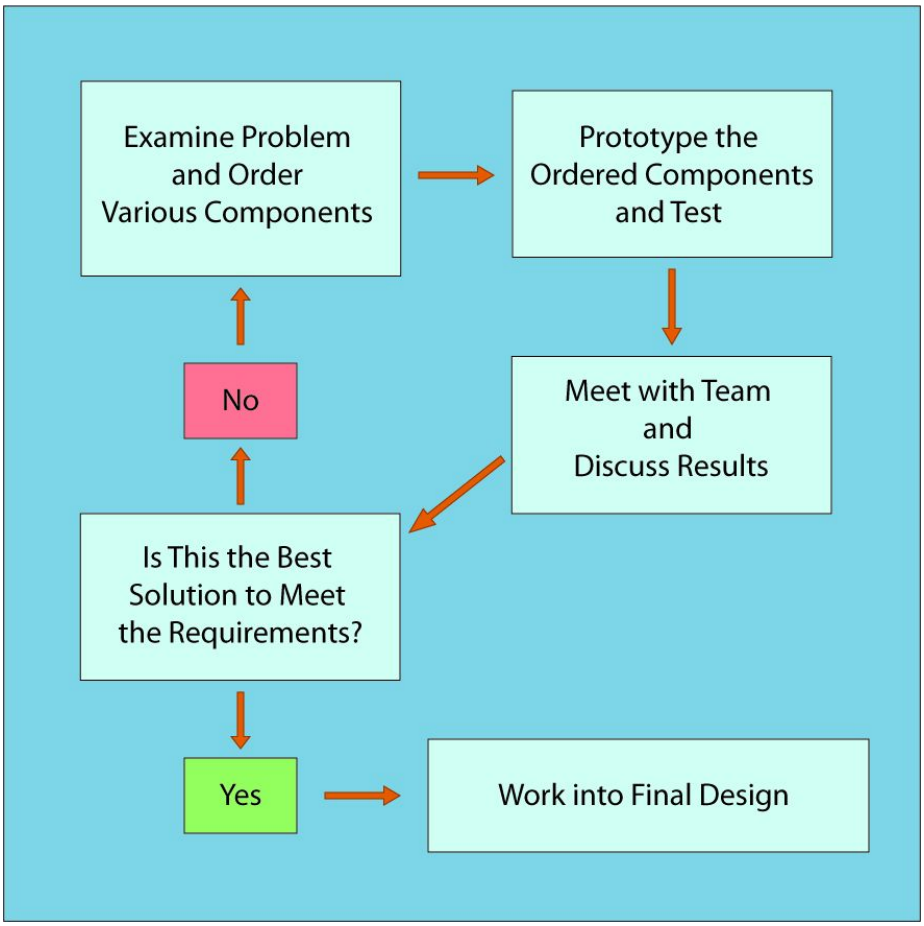


Figure 5.2 - U-Park Team Design Method

5. 3 Project Management

Effective management of the design team is fundamental in the success of delivering a quality final product that meets, or exceeds the initial requirement specifications of the system. Throughout the design process, developing the team and ensuring that expectations were clearly stated and ensuring communication was always open, was very important. For this reason, the software used to ensure this openness and ability to collaborate had to be chosen carefully to act as a catalyst for effective collaboration. If an inefficient procedure was used from the start, changing half way through would be an added hurdle that, under a timeline as short as that in senior design one and two, would add undue stress to the development process, and could jeopardize cooperation and participation required by the U-Park design team's members.

To ensure that the team was able to have 24/7 open communication the team decided to make use of the GroupMe mobile app. This app is available on both android and iOS devices, which ensured universal usability between the team's members. The interface also allowed for communication which would be

separated from the team members personal phone messages. GroupMe provides a differentiation from team members personal and “work” messages, and ensures that whenever a notification from the GroupMe app appeared, team members can see at a glance that attention needed to be paid to communication specific to the project’s development.

The other piece of software pinnacle to the management of the U-Park documentation was the use of Google’s cloud services. Namely the use of Google Drive and Google Docs. During the initial assignments such as the table of contents submission, the team created a shared document on the Google drive containing a universal table of contents for the project documentation. This table of contents was color coded by team member so that each team member was able to see what sections each person was responsible for. This also allowed for efficient design review sessions during the documentation process, as while walking through the document the table of contents could be used to examine the total completion of the document, and the completion status of each of the team members. One other important feature of Google cloud services is the ability to examine edit history. This feature ensures that incorrect changes can be easily reversed.

Team meetings were scheduled at least once every week during the senior design one semester to ensure that the team remained engaged, and was able to begin growing as a group. While group unity is often an overlooked aspect of design teams, the amount that team members feel that they are worthwhile contributors, and the ability to share concerns and opinions can either make or break the quality of the final project. Throughout the design process members were encouraged to share any concerns or problems that they were having. This ensured that any problems or concerns could be solved before they turned into more significant issues, and that group members would always feel that any personal concerns were able to be reasonably explored and resolved.

While the management style required during Senior Design two, will be different than that required in Senior Design one, the overall communication aspect and group collaboration aspects must remain unchanged. The reason for the difference in management styles stems from a different set of problems the team is likely to face. The team will likely require more face-to-face meetings, however, unlike during the initial design document creation process, the entire team will not always be required to attend these meetings. For instance, if members who are working on the electrical components of the system are having issues, those working on the database and software of the U-Park system would not be required to attend the hardware-specific meeting. Instead these other members could use that time to continue working on other aspects of the final design. While the team will still have meetings where the entire team needs to collaborate and be present, these meetings will be reserved for the fundamental design reviews.

5. 4 Implementation

Implementation of the U-Park system happened in multiple phases. The first of which was during the initial design process, the second was during the experimentation and initial prototyping and the third phase will be during the final implementation of the U-Park system during the Senior Design two semester in the summer of 2016. While these various implementations are slightly different, they all combine to develop the final implementation at the end of senior design two which will be demonstrated to the faculty.

The first implementation of the U-Park System was performed during the first couple weeks of the design process. This implementation included a proof of concept prototype for various methods which could be used to detect cars. The implementation which was decided to be the most effective by the team was a simple ultrasonic sensor program using an Arduino UNO microcontroller board. An ultrasonic sensor was connected to the board and a simple test program was written to measure distances away from the sensor in centimeters.

The first test prototype led to other prototypes and tests of other equipment. Namely, the various Wi-Fi modules available on the market were tested. It was discovered that some of the Wi-Fi modules were easier to develop with than others. Even though the other wifi modules are slightly more expensive, the extra quality and programmability is necessary to ensure that the final prototype is a reliable product.

The final implementation presented to the faculty and demonstrated during the senior design expo will have a polished design. The sensor modules will have custom 3D printed bodies and will exist as a fully integrated system. The U-Park system's final implementation will be demonstrated through the use of a scale model of a parking garage and will allow the audience to move the model cars in the garage to observe the changing status on the web interface. This demonstration will prove the concept and will show that the system meets the stated specifications. Furthermore, this demonstration will provide a stress test for the system as the system will be relying on a UCF wireless network notorious for poor connectivity. The majority of the testing will not be carried out in the engineering 2 atrium where the Senior Design two expo will be held. And, since the wireless network is so "spotty", the design team will have to take this into account. The U-Park design team must be able to supply its own wireless internet connectivity in the case of network problems in the atrium.

6. Realistic Design Constraints

6. 1 Standards

To implement this project, a careful observation, test, and selection on standards was made. These standards made the implementation of the project a much easier process both in design as well as in costs.

6. 1. 1 Wireless Standards

In the search for a mean of communication between the microcontroller and the server, several alternatives were evaluated. There are many alternatives in the market to accomplish this task. The most common are based on IEEE standards for data communications. The following standards were evaluated:

- IEEE 802.3-2008 (Giga-byte Ethernet or 1000Base-T). This standard is based on a 4 pair twisted cable allowing speeds of up to 1 Gbps.
- IEEE 802.15.1 (Bluetooth - Standard not maintained). This Low power consumption solution permits the communication of two devices in a short distance. This is a low cost solution, however it was discarded due to distance limitations, and a lack of consistent standards. The latest version 4 LE allows throughput of up to 24 Mbits, but is not universally recognized.
- IEEE 802-11 (WiFi). Commercially known as WiFi this standard uses 2.4Ghz ISM band and allows speeds of 54 Mbps. When implemented along with TCP/IP, this standard becomes the best alternative for replacing “wired networks”. In the case of this particular project this was the best alternative, and was selected to be implemented with the microcontroller.

6. 1. 2 Other Relevant Standards

- TCP/IP Embedded in the Wifi module TCP/IP is used as the transport protocol.
- Telnet Application layer. Telnet is used to send/receive data between the microcontroller and the server applications.

6. 1. 3 De facto Standards

Even though these standard are not industry recognized as *de jure standards*, *De Facto standards* are used due to the popularity imposed by their manufacturers. In some cases these standards are as much recognized as any *de jure standard.* In this project the following *De facto standards* were used.

- Microsoft Visual Basic. Used in the programming language in the design of the server application.
- AVR-C. Even though it is based on the popular C language, this version is adapted to be used on the ATMega-XXX family of microcontrollers. It is used to program the microcontroller used in this project.

6. 2 Economics and Time

This section will discuss the economic cost of the U-Park system and the time considerations associated with designing and implementing such a system. This section will also provide an overview of the ideas surrounding some fixed costs required to implement the U-Park system.

To design a system, which could be reliable enough to be successful in an industrial setting, presents some challenges. These challenges are exacerbated when such a system is to be fully fleshed-out in the short time available in the the two semesters given to developing, building and testing an entire working project. Because of this, to fully deploy a viable version of the U-Park system, much more time to fine-tune the final product would be required. The last thing a prospective customer would want would be an expensive system constantly requiring updates and repairs due to oversights in the development process. Also, more testing of the sensors would be required to fully ensure that, over time, the sensors would be reliable, and that normal wear and tear would not compromise the overall quality of the system.

The economic impact of the U-Park system changes with the size and restrictions of the garage in which the platform is implemented. Smaller garages would have a higher cost per spot than a garage with the same infrastructure and network resources. This is due to the overhead of installation costs, the costs of the design team to customize the placement of the access points, and the cost of retrofitting the garage with the necessary electrical and network routing and wiring. For instance, if the U-Park system were to be implemented in a garage lacking the required network connection, the owners of the garage would be required to factor in the continuing cost of an internet package with an internet service provider (ISP). This would disproportionately affect the implementation of small garages more than larger garages because the fixed cost of the subscription would be proportionately larger compared to the installation cost of

the entire system. While network access is a caveat of the proposed system, in the ever-increasing connected world, this cost reduces over the lifetime of the system as new networks are made available.

The initial costs of the U-Park system are fairly minimal, and the cost of maintaining such a system are mainly reliant on: the cost of electricity to power the sensors and routers, and the cost of network access to be able to communicate with the network interface. The only other time and cost considerations associated with the proposed system would be maintenance. So, to ensure this cost would be as low as possible, the system would need to undergo in-depth testing and reliability analysis over the life cycle of the designs. Also, continuing user-studies would need to be performed to ensure that users of the U-Park system are able to consistently receive the information that they need to avoid the problems currently prevalent in parking at highly populated environments such as UCF.

6. 3 Ethical, Privacy, Health and Safety

This section will explore the ethical, health and safety considerations for the U-Park parking information system and its components. Because the users of U-Park do not directly interact with any of the hardware of the system the negative ethical, health and safety aspects are minimized. Below, these aspects will be explored in more detail.

There are very few ethical concerns for a system such as U-Park. Most of the ethical problems would come from the users, weary of using the system due to privacy concerns. In today's day and age, privacy is often an afterthought for many embedded systems. Consumers have become aware of this fact, has left them with a feeling that most technology that makes life easier, must also come with a cost of giving some level of privacy. Because the users would not directly be paying for the services U-Park would have to offer, some would assume that the cost for the users is privacy. This fear is unfounded in reality, however, as the only costs associated (ethical or financial) with the U-Park system are the installation cost, and the small sum of money required to keep the servers for the database up and running. The U-park system was designed with security in mind and users are not tracked. In fact there is no possible way for users to ever be tracked due to the types of sensors that are being used. The sensors only tell the server if someone has pulled into a parking spot, not who the person is.

Privacy concerns of the U-Park system are fundamentally non-existent and relate only to the concerns associated with the aforementioned ethical concerns. Because the users of U-Park are not tracked by any physical or software means by the system itself, the users don't have to worry about any privacy intrusions from the system. The one main area where users would be at risk of privacy being compromised would be on the online web interface through the IP address

used to access the interface. For this reason, appropriate security protocols must be adhered to when developing the application, ensuring that malicious users would not be able to compromise the security of the website itself.

Ensuring the health and safety of U-Park users is not compromised is fundamental to the success of such a system. Any failures resulting in injury or the damage of personal property could be a huge liability for the owners of the garage and the designers of the system. The obvious safety concern would be one of the modules, which are mounted on the ceilings of the garages, falling on someone's car, or worse, falling on someone. To minimize this risk, all components mounted on ceilings would need to be secured using concrete screws with maximum weight thresholds far exceeding the weight of the installed modules. This would ensure that as time goes by, and the parking structure ages, the ceiling-mounted components would not fall and result in injury or the destruction of personal property. Other than components falling from height and injuring users, the only other health and safety risk would come from electric shock. Because the modules are powered with AC power from the garage, new connections would need to be made into the power of the garage. These new connections would need to abide by electrical safety standards to ensure that no users could accidentally come into contact with any exposed wiring. Also, on that same note, measures would need to be taken to ensure that no shortcuts were taken during installation, which could develop into safety issues as time progresses.

6. 4 Manufacturing and Sustainability

Manufacturing and maintaining the U-Park system is a very important consideration during the design process. For this reason, the design needs to be as simple as possible. This will reduce not only the cost of the components, but also simplify the system in terms of maintenance. For instance, if a module is not communicating over WIFI, a module could simply be swapped and in many circumstances, a costly repair or recall could be avoided.

Manufacturing of the U-Park system would initially be carried out in custom batches. For the prototype demonstrated at the Senior Design two expo at the end of the summer of 2016, the hardware will be custom ordered from suppliers such as OSH Park. OSH Park is a custom PCB designer which allows customers to submit a design file, and within around two weeks, OSH Park will send out three custom PCB's. While there are a few options for custom PCB designers in the marketplace, the U-Park has decided to use OSH Park due to its cost and, speaking with past senior design groups, other teams have had good experiences using this supplier.

If the U-Park system were to be implemented in a full scale purchase for a parking garage, the small custom orders would simply not provide the

cost-effectiveness and volume required. For full-scale implementations, a company that specializes in larger orders would be needed. One example of such a company is the Captor Corporation, which offers much more customization in the way of materials, and would be able to supply the quantity of custom boards required to fully outfit a parking garage.

Production of other components such as the 3D printed housing of the sensor module would also need to be scaled up in a full-size garage implementation. The housing would not be able to be 3D printed as 3D printing takes far too long, and is not very wear-resistant, especially in the hot florida sun and humidity variations. The aluminum used for the sensor arms for the two sensors on the sides which detect the side parking spaces would also need to be ordered in bulk. Custom cutting and finishing individual aluminum sections for each garage implementation would not only be impractical, but would also not be financially viable. The aluminum would instead need to be purchased directly from the manufacturer.

Manufacturing of the final prototypes, due at the end of the Senior Design process, will be made in-house and will be built by the design team itself. The team will use facilities offered by UCF and will obtain the materials either from the facilities at UCF, or will purchase the supplies locally or online. The supply chain of the prototypes is vastly different than a full-scale production run, and much of the process would need to be optimized to meet full-scale product orders. One hurdle that the U-Park system would have to overcome is that there are no such thing as “small” orders in a full-scale implementation, as outfitting a garage would be an “all or nothing” deal, and there are not really any small garages in existence as the entire idea by constructing parking garages is to fit as many parking spots into as small of a geographic location as possible.

Because the U-Park system needs to be maintained after installation, the goal of the design team was to keep the system as simple as possible. The system is designed such that much of the system is modular in that when a component fails, it’s place in the network can simply be replaced with a new component. Sustainability was a significant aspect of the U-Park design, however, until the system has gone through more extensive testing, the common faults and problems remain unknown. For this reason the system must be as perfect and streamlined as possible before a single sensor module would be installed in the first garage.

7. Design

7. 1 Design Specifications

7. 1. 1 Sensor

The following are specifications for the ultrasonic sensor that the parking system will use to detect whether or not there is a car parked in a given parking space. While other options such as hall effect sensors were considered, the ultrasonic sensor was chosen due to its accuracy and simplicity. Also, the ultrasonic sensor is able to be mounted in any orientation as long as it can detect whether there is an object (i.e. a car) that has entered its detection path, and has crossed the distance threshold set by the distance parameters in the source code. The specifications for this sensor are shown in the tables below and on the next page.

Table 7.1 - Sensor Pins:

5V DC - 15mA power supply
Trigger Pulse Input
Echo Pulse Input
0V DC Ground (connected to the 5V power supply)

Table 7.2 - Hardware Specifications:

Working Voltage	5V (DC)

Working Current	15mA
Working Frequency	40 kHz
Maximum Detection Range	~ 4 - 5 meters
Minimum Detection Range	~ 2 centimeters
Measuring Angle	15 degrees
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and range in proportion
Product Dimensions	(.45 x .20 x .15) centimeters
Operating Temperature	-40˚ C to 120˚ C
Overall Accuracy	$(95\%)^{+}$
Life Span	2 years (minimum)
Weight	8.5 grams

7. 1. 2 WIFI - LAN Transceiver

Wireless communication is a primary requirement of the U-Park System. To this end, the group has chosen to use a transceiver that works on the 2.4 GHz WIFI band. This will allow the modules to communicate directly with the access point without the need for extra intermediary communication hardware/software. Because the modules are arranged in a star network configuration (described in more detail in section **4. 4. 1. 3**), each module must have a way to communicate with a central access point. To accomplish this the team has chosen to use the ESP8266 WIFI Transceiver (shown in the **Figure 7.1** below).

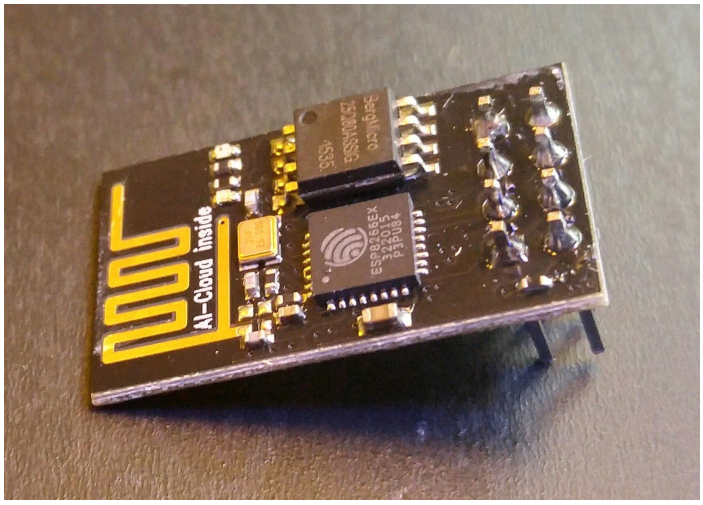


Figure 7.1- ESP8266 WIFI Transceiver Module

This transceiver has a built-in antenna, and an eight pin header pin connector for power and data-transfer. A problem which was encountered during preliminary research was that the team ordered a set of transceivers which required external antennas (not included). This became an issue as it proved difficult to find external antennas which would fit the connector on the receiver, and which would be properly matched for this particular transceiver. This mistake, however, did help the team better justify the slightly higher costs associated with transceivers with built-in antennas.

The Input voltage required for this module is 3.3V or 5V depending on the model purchased. Therefore the team opting to use the 5V version for the design, and can therefore avoid the extra component cost to regulate down to 3.3V. The 5V option is able to be supplied by the same 5V source that the ATMega chip and rest of the components of the system are. The module contains the necessary Tx (transmitter) and Rx (Receiver) pins and three GPIO pins which will connect to pins on the ATMega328p-pu microcontroller. The pin arrangement is shown in **Figure 7.2** on the next page. Although this is the pin-out for the 3.3V versio the pins are all the same except for the +VCC pin.

Figure 7.2 - ESP8266 WIFI Transceiver Module Pin Arrangement

One benefit of this transceiver is that it is a very common module for hobbyists, and therefore has ample amounts of available documentation, and tutorials are available to help in the development process. Helpful programming libraries are also common for the module. Because there would be hundreds of modules per garage when outfitted with the U-Park system, the code and libraries need to have reliable and tested files to ensure reliability of the system. If for instance the system had an unknown bug that caused errors to occur in the WIFI transmission, resulting in the sensors not being able to transmit any data, the entire system would effectively be useless. While other communication systems such as bluetooth were considered, the ESP8266 WIFI module was chosen due to its size, power, available development resources and its ability to communicate directly with the U-Park server.

7. 1. 3 Power

Below are the list of all the specifications for the power of the board.

- Input Voltage Limits:
- Recommended: 7~12
- Absolute: 6~20V:
    - Below 7V may cause the 5V levels on the board to waver, fluctuate, or sag, causing board instability and less accurate
    - Sustained voltage above 12V will cause additional heating on the linear voltage regulator of the microcontroller, which could cause it to overheat.
- Input/output (I/O) pins: -0.5V to +5.5V
- Output Current Limits:
- When powered by USB: total of 500mA
- With external battery or power supply: total of 500mA~1A
- 5V pin: same as above: 500mA or 500mA~1A
- Each input/output pin: 40m
- Sum of all input/output pins combined 200mA
- Life Span: At least 2 years
- Fit within the package size
- Little to no heat generation
- Capable of a quick power on/power off cycle

7. 1. 4 Microcontroller

The micro-controller to be built by the team; features an ATMega328P CPU, and an Ethernet WIFI module. The inexpensive 8 bit CPU allow the builders to connect up to three sensors, the WIFI module, and LED that will identify the different statuses (on/off, connected/not connected, etc.) that may prevail during operation.

The 5V DC required to energize both the microcontroller and the sensor is supplied an AC/DC converter, designed and built by the team. The function of the microcontroller consists in querying the three sensors connected to it, and determining if a vehicle is located in one of the three parking spots being monitored. Every thirty seconds the microcontroller will query for information. Once the microcontroller has acquired the data it will transmit it to the server through a Telnet session, using the internet or a dedicated network.

7. 1. 5 Server

The hardware requirements for the server will depend on the data being handled, and the connections expected from the clients (users). Fortunately with the

existing technologies, a computer powerful enough to fulfill the most demanding needs, won't exceed $500.

Described merely as an example (but not limited to) the Dell Inspiron (I3847-3538BK) with a street price of $499.99, which includes:

- Processor: Intel Core i5.
- 12Gb RAM memory.
- 2Tb Hard drive.
- 10/100/1000 Ethernet board (in case of extreme demanding accesses, LAN connection could be upgraded to fiber optics).
- Monitor, keyboard, sounds card, USB ports.

The above described configuration meets the requirements needed for the project. As for the main functions to be performed by the server, they can be enumerated as:

- Receive all data sent by all microcontrollers.
- Determine the status of the parking spots.
- Update the system database.
- Web Server from where all clients will get updated parking information.
- Determine problems with a sensor, a microcontroller, or the connection, and notify administrator of the found problems.

In order to accomplish the previous tasks, the server will be running the following programs:

- Monitor (As a windows server).
- A DBMS system, and the required applications to maintain data updated.
- A Web Server package.

7. 1. 6 Application Interface

The application interface is where the user will have direct contact with the system. This is where users will be able to view the parking levels in the garages that are monitored by the U-Park sensors. The application will pull data from the database in order to display the parking levels of each garage.

The system will be designed to have two types of users. These types of users will be administrators and normal users. The administrators will have access to some of the back end features, while normal users will simply be able to see the parking spaces in the garages.

Administrator accounts will be designed for the owners or custodians of the garage. They will log in through the web interface and have the ability to maintain the system. They will be able to mark garages or floors as closed or reserved, run diagnostics to ensure the system is running correctly, and other back end activities that the owner of the garages would need.

When non administrator users log in to the application, they will be able to add the garages that they would like to view. The first time they log-in, they will be prompted to select which garages they would like to personally track. On subsequent log-ins, the system will remember and display the status of the garages that the user has selected.

Users accessing the web application will be able to see the parking situation in each parking garage. The main page will show the overall amount of free spaces in each parking garage in which this system is used. If a garage is full, there will be a “busyness” indicator that will show an approximation of how many people are in the garage trying to find spots. This crowd notification will use an algorithm that determines how busy any full garage is by how many spots are quickly taken up after a car has left. This will allow users to be able to determine which garage they will have a better chance of finding a spot in even if they are all full.

When a user clicks on any garage, they will be able to see a breakdown of how full each level of the garage is. The menu will show how many spots, if any, are open on each floor. This is to allow someone to quickly determine where he or she can go to find the open spots in the garage.

The application will be built first as a web application. It will be built so that it will scale in order to be usable by people on computers, tablets, and phones. An Android application will be created as well for mobile phones.. This application will allow users on their cell phones to quickly pull up the information. It will not have any different information than the web page.

7. 2 Microcontroller

When team nine was first brainstorming the ideas that would lead to the system, the team knew that a microcontroller would be necessity. It would need to have enough processing power and memory to control the ultrasonic sensors and a Wi-Fi module. After many group meetings and research, team nine decided to use a microcontroller commonly used on the Arduino Uno board. The arduino uno board architecture has more functions than what team nine need for the project, and can be optimized for this project. In this section, the design process and function of each component will be discussed.

There are many types of microcontrollers that group nine considered, such as ATmega48A/PA/88A/PA/168A/PA/328/P. The Atmel ATmega48A/PA/88A/PA/168A/PA/328/P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications. They differ only in memory sizes, boot loader support, and interrupt vector sizes.**Figure 7.4** summarizes the different memory and

interrupt vector sizes for the devices. Compared to other microcontrollers, the ATmega328 has a fairly large memory space. Some of the features of this microcontroller include:

- 23 Programmable I/O lines
- Operating voltage: 1.8 to 5.5V
- Temperature Range: -40 celsius to 85 celsius
- Speed grade: 0-4MHZ at 1.8-5.5V, 0-10 MHz at 2.7-5.5V, and 0-20 MHz at 4.5-5.5V
- Power consumption: Active mode 0.2mA, power-down mode 0.1uA, and power-save mode 0.75 uA
- Special microcontroller features: Power-on reset and programmable brown-out detection, internal calibrated oscillator, external and internal interrupt sources, and six sleep modes
- Three flexible Timer/Counters with compare modes
- In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.
- In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The ATmega48A/PA/88A/PA/168A/PA/328/P are low-power CMOS 8-bit microcontrollers based on the AVR enhanced RISC architecture. The block diagram of the AVR architecture is shown in **Figure 7.3** This section discusses the AVR core architecture in general.

The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memory, perform calculations, control peripherals, and handle interrupts. In order to maximize the performance, the AVR uses a Harvard architecture, which has separate memories and buses for program and data. Instructions in the program memory are executed with single level pipelining. While one instruction is being executed, the next instruction is prefetched from the program memory. This concept allows instructions to be executed in every clock cycle.

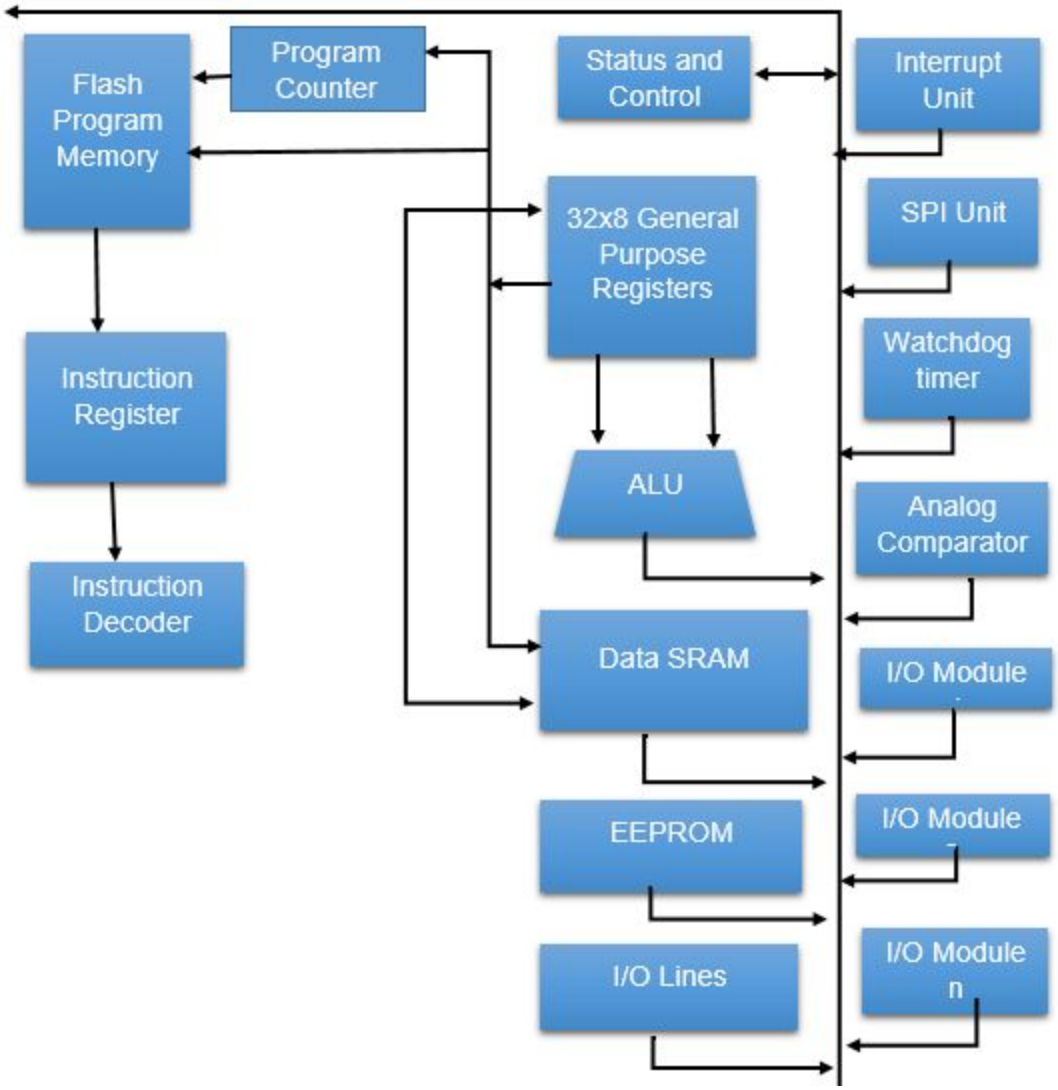


Figure 7.3 - Block Diagram of AVR Architecture

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4 KBytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega48PA	4 KBytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88A	8 KBytes	512 Bytes	1 KBytes	1 instruction word/vector
ATmega88PA	8 KBytes	512 Bytes	1 KBytes	1 instruction word/vector
ATmega168A	16 KBytes	512 Bytes	1 KBytes	1 instruction word/vector
ATmega168A	16 KBytes	512 Bytes	1 KBytes	1 instruction word/vector
ATmega328	32 KBytes	512 Bytes	2 KBytes	1 instruction word/vector
ATmega328P	32 KBytes	512 Bytes	2 KBytes	1 instruction word/vector

Figure 7.4 - Memory Size Table

Below are the list of components will be connected to the microcontroller:

- AT mega 328
- 16 MHz crystal clock
- Two 22 uF capacitor, Two 10uF capacitor
- A Switch
- LED
- Power supply
- Voltage regulator
- 10 kΩ and 200Ω resistor

The main component is the ATmega328. The Atmega328 is a very popular microcontroller chip produced by Atmel. It is an 8-bit microcontroller that has 32 Kbytes of flash memory (0.5 Kbyte occupied by bootloader), 1Kbyte of EEPROM, and 2 Kbytes of internal SRAM. The maximum operating frequency is 20MHz The Atmega328 has 28 pins. It has 14 digital I/O pins, of which 6 can be used as PWM outputs and 6 analog input pins. These I/O pins account for 20 of the pins. The description for each pin of the ATmega328 is shown in Table 7.3 below.

Two important pins are Vcc and GND. The Atmega328 is a low-power chip, so it only needs between 1.8-5.5V of power to operate. Since we are using an external power supply for the board, a voltage regulator is needed to stabilize the 5V DC voltage. An LED is attached to the power input and therefore acts as a power-indicator.

One nice feature is that the Atmega328 chip has an analog-to-digital converter (ADC) built-in. Without an ADC, the Atmega328 would not be capable of interpreting analog signals from the sensors. Three pins are needed for the ADC to function: AVCC, AREF, and GND. AVCC is the power supply, positive voltage, for the ADC. The ADC needs its own power supply in order to work. Therefore, GND is relative to the power supply. AREF is the reference voltage that the ADC uses to convert an analog signal to its corresponding digital value. Analog voltages higher than the reference voltage will be assigned to a digital value of 1, while analog voltages below the reference voltage will be assigned the digital value of 0. Since the ADC for the Atmega328 is a 10-bit ADC, meaning it produces a 10-bit digital value, it converts an analog signal to its digital value, with the AREF value being a reference for which digital values are high or low. The last pin is the RESET pin. This allows a program to be rerun without having to disconnect the power.

Table 7.3 - Pinout for the ATMega328p-pu and Description for each Pin:

1	PC6	Reset
2	PD0	Digital Pin (RX)
3	PD1	Digital Pin (TX)
4	PD2	Digital Pin
5	PD3	Digital Pin (PWM)
6	PD4	Digital Pin
7	Vcc	Positive Voltage (Power)
8	GND	Ground
9	XTAL 1	Crystal Oscillator
10	XTAL 2	Crystal Oscillator
11	PD5	Digital Pin (PWM)
12	PD6	Digital Pin (PWM)
13	PD7	Digital Pin
14	PB0	Digital Pin
15	PB1	Digital Pin (PWM)
16	PB2	Digital Pin (PWM)
17	PB3	Digital Pin (PWM)
18	PB4	Digital Pin
19	PB5	Digital Pin
20	AVCC	Positive voltage for ADC (power)
21	AREF	Reference Voltage
22	GND	Ground
23	PC0	Analog Input
24	PC1	Analog Input
25	PC2	Analog Input
26	PC3	Analog Input
27	PC4	Analog Input
28	PC5	Analog Input

Two of the pins are for the 16 MHz crystal oscillator. microcontroller is designed to handle off-chip crystals that have a frequency of between 4-16 MHz. The

crystal oscillator provides a clock pulse for the Atmega chip. A clock pulse is needed for synchronization so that communication can occur in synchrony between the Atmega chip and a device that it is connected to (sensor in this case). The crystal oscillator are connected across two capacitors (C2 and C3). If the incorrect crystal and external capacitors are selected, it can lead to a product that does not operate properly or fails prematurely.

The crystal equivalent circuit is shown in **Figure 7.5**. The series RLC circuit is called the motional arm. Capacitor C1 represents elasticity of the quartz, inductor L1 represents the vibrating mass, and the resistor R1 represents losses due to damping. Capacitor C0 is called the shunt or static capacitance, and is the sum of the electrical parasitic capacitance due to the crystal housing and electrodes. By using a Laplace transform, two resonant frequencies can be found in this network. The series resonant frequency, fs, depends only on C1 and L1. The parallel frequency fp, also includes C0. The reactance vs frequency characteristic in shown in **Figure 7.6**.

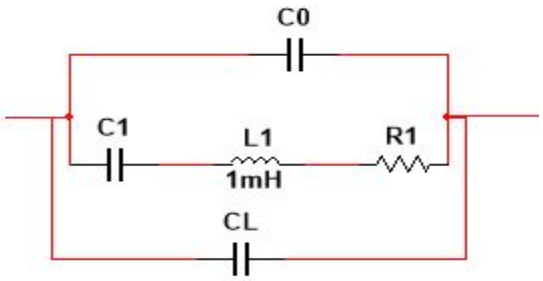


Figure 7.5 - Crystal Equivalent circuit

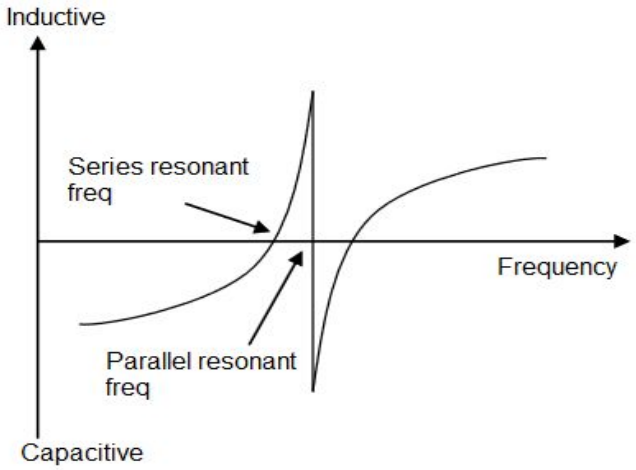


Figure 7.6 - Crystal Reactance Characteristics

In the figure below (**Figure 7.7**), the schematic for the 16.0 MHz crystal is shown according to its capacitance values.

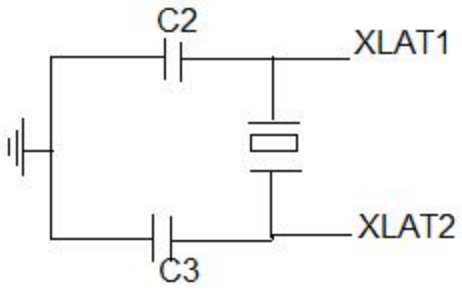


Figure 7.7 - Schematic of External component

The values of C2 and C3 are calculated using the following formula:
CL=C2C3/(C2+C3) +CS
Where:

- CL: The crystal load capacitance (15pF for 16MHz crystal)
- CS: The tray capacitance (2pF- 5pF range)

Typically, the values of C2 and C3 are equal. If we for example use the dummy value C, then C=2*(CL-CS). In this case, the value of C needs to be between 20-26 pF. In our design, we chose C to be 22pF.

7. 3 Sensors

Ultrasonic sensing technology is based on the principle that sound has a relatively constant velocity. The time for an ultrasonic sensor's beam to strike the target and return is directly proportional to the distance to the object. Therefore, ultrasonic sensors are used frequently for distance measurement applications, such as in backup systems of cars. There are four basic components of an ultrasonic proximity sensor: transducer/receiver, comparator, detector circuit, and solid-state output.

The ultrasonic transducer sends sound waves outward from the face of the sensor. The transducer then receives echoes of those waves when they reflect back off of an object. The schematics of the transmitter are shown in **Figure 7.8** and **Figure 7.9**.

The working principles of ultrasonic transceivers are as follows:
:

- 40 KHz signal generated by the microcontroller

- The signal is passed to a resistor for safety when the signal is refracted forward a series of diodes and transistors.
- The signal is fed to a current amplifier circuit which is the combination between diodes and transistors.
- When the signal from the input logic high (5V) then the current will pass through the diode D1 , then the current will bias transistor T1.
- When the signal from the input logic high (0V) then the current will pass through the diode D2 , then the current will bias the transistor T2.

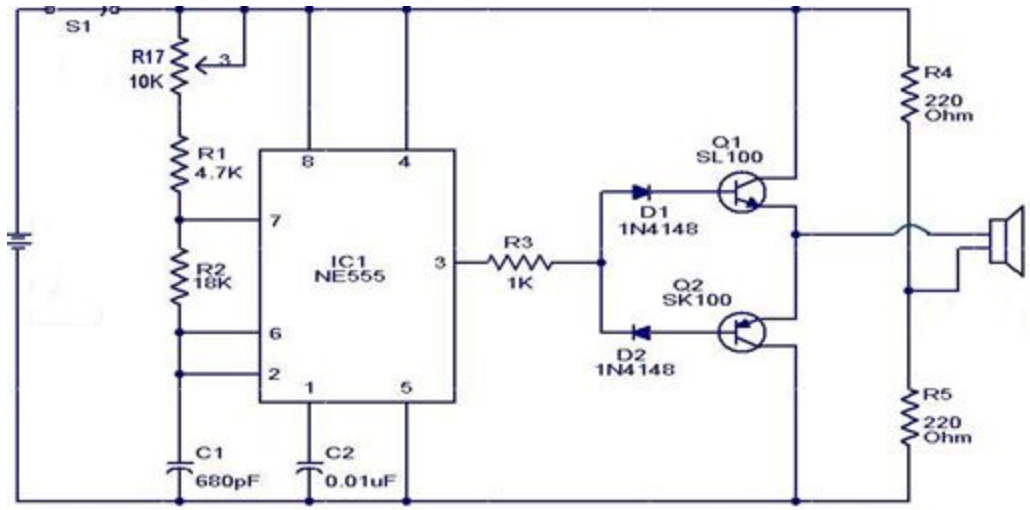


Figure 7.8 - Ultrasonic Transmitter Schematic

The ultrasonic receiver circuit uses a two-stage amplifier, a rectifier stage, and an operational amplifier in inverting mode. Output of op-amp is connected to a relay through a complimentary relay driver stage. When switch S1 of transmitter is pressed, it generates ultrasonic sound. The sound is received by ultrasonic receiver transducer. The receiver converts the sound to electrical variations of the same frequency. These signals are amplified by transistors T3 and T4. The amplified signals are then rectified and filtered by high pass filter at frequency 40 KHz. The filtered DC voltage is given to inverting pin of op-amp IC1CA3140. The non- inverting pin of IC13140 is connected to a variable DC voltage .The inverted output of IC1CA3140 is used to bias transistor T5. When transistor T5 conducts, it supplies base bias to transistor T6. When transistor T6 conducts, it actuates the relay.

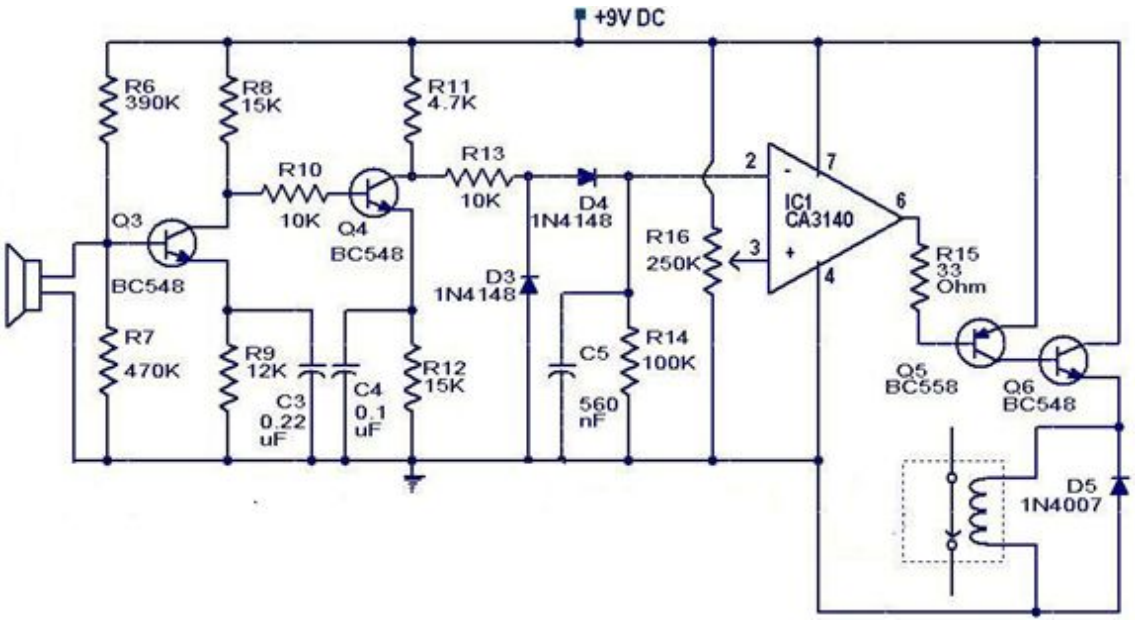


Figure 7.9 - Ultrasonic Receiver Schematic

When the sensor receives the reflected echo, the comparator calculates the distance by comparing the emit-to-receive time frames to the speed of sound.

7. 4 Communication between Microcontrollers and Router

The core object of this project consists in identifying empty/occupied parking spots in any parking area at UCF. To accomplish this task; a series of ultrasound sensors are monitoring each spot in a 30 second interval, 24/7. A microcontroller, designed will be implemented by team 9 to perform the following functions:

1. At boot up, the microcontroller after connecting to the router; will declare itself as a telnet server with a static IP address already allocated and reserved at the router.
2. Controls three sensors (hardwired to the microcontroller) at one time; the sensors are identified as left (1), center (2), and right (3) sensor. Every 30 seconds each sensor is queried about the status of the parking spot; the sensor will return a zero " 0" if it doesn't detect any object taller than one meter from the floor, and will return a one " 1" if it does detect the object.
3. The micro controller broadcasts the information just acquired in a record with the format displayed in figure 2.
4. The micro controller sleeps for 30 more seconds and repeats the cycle at step 2.

Table 7.4 - Message structure:

Pos	Description	Use	Format	Mask	Range
1-3	Microcontroller (MC) unique ID.	Identify the micro-controller among the other. The server DB will associate this ID to a MC placed in a specific building, and floor within the building.	Integer	000	000–999
4	Sensor 1 status	Determines if the parking spot 1 (left) is occupied (1=true), unoccupied (0=false), or an error was detected (@)	Boolean	0	0 – 1
5	Sensor 2 status	Determines if the parking spot 2 (center) is (1=true), unoccupied (0=false), or an error was detected (@)	Boolean	0	0 – 1
6	Sensor 3 status	Determines if the parking spot 3 (right) is occupied (1=true), unoccupied (0=false), or an error was detected (@)	Boolean	0	0 – 1

7. 5 Software

7. 5. 1 Languages

The following languages will be used in the design of the U-Park application

- Bootstrap CSS
- HTML
- PHP
- Javascript
- Java

The website application will be designed using Bootstrap CSS and HTML. The Bootstrap CSS will allow the application to be scalable to all different types of screens. This will allow users to be able to access the website application on the device of their choice. The application will be sized to look the same on any of these devices.

The scripting of the website will be written using PHP and Javascript. Using PHP will allow team 9 to implement dynamic scripting to the website application. This will be used to query the database and return results on the fly. This will allow the user to receive up to date information on the state of the garage they are searching. Javascript will be used for adding other scripts to the web pages. It

will be used to display messages from the system and to dynamically update the tables gathered from the database.

Java will be used to create the future Android application. This is the main language used for creating applications on the Android platform. The desired mobile application will be almost identical in function to the website application. The difference is that Java, as a language, will be used for all functions. One benefit of using Java is that it allows the programmer to display a page, query a database, display the database information, and reload information on the fly without the need for other scripting languages.

7. 5. 2 Programming Pattern

The main programming pattern employed by the U-Park application will be the Facade pattern. The users will be given a simple interface, and they will not see or need to interact with any of the complex functions that are used to create that interface. **Figure 7.10** below shows an example of how the user sees the simple interface, and does not see anything that is happening server side behind the scenes.

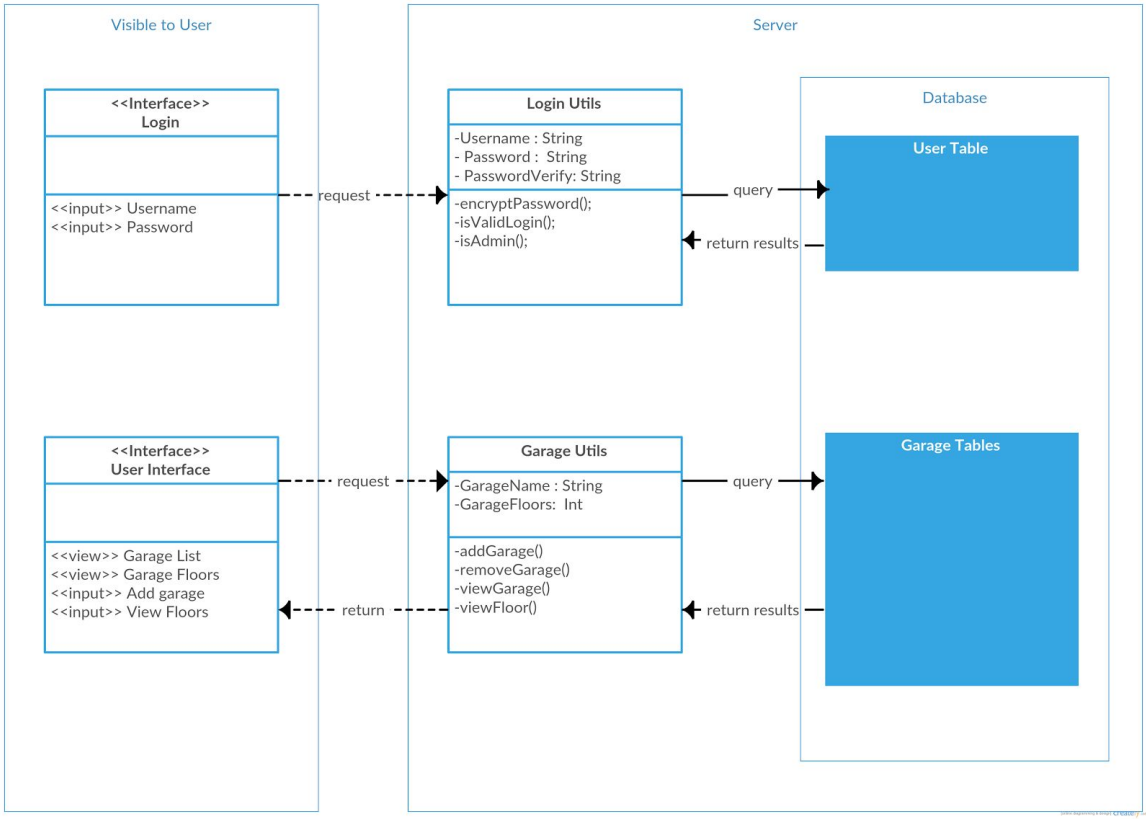


Figure 7.10 - Facade Pattern for U-Park Application

7. 5. 3 Web Application

7. 5. 3. 1 Overall Design Philosophy

The U-Park application will be designed to give users quick access to the status of the garages of their choice. The application will be accessible through a website and an Android application if time permits. The website will be designed so that it will be scalable to all size of screens. This will be accomplished by using Bootstrap CSS.

The application will also feature a login and register page. Using this dynamic language, the application will make a call to the database to check the credentials of a user who is trying to sign in, and will check if the user is an administrator or a standard user. It will also add users who are registering new accounts. A database call will also be made to ensure that no duplicate accounts are created. The passwords will also be encrypted using the sha256 algorithm for the safety of the users.

Once a user has logged in, they will view their interface. This interface will be built using HTML, PHP, and Javascript. The main page will be created with HTML and will use PHP to make any database calls. Javascript will be used for any event messages, and also to dynamically update the information on the webpage without the user having to click refresh.

The mobile application will have the same features as the website application. It will be written in Java using Android Studio. Java allows for similar functionality to the languages used to create the website. Java is the most popular language used for making Android applications.

7. 5. 3. 2 User Interface

Once a user logs into the application, they will enter the user interface. Unless they are an administrator, in which case, they will be directed to the admin interface. The user interface will be built using Bootstrap CSS, HTML, PHP, and Javascript. The user will have the option to add garages that they would like to view. This will use PHP to create a call to the database which will pull up the available garages for viewing and allow the user to mark which ones they want on their personal page.

Once the user has selected which garages they would like to view, their main page will show a list of the garages they have chosen. In order to continuously update this page without the user having to reload, a separate PHP table will be called using a Javascript function. The Javascript function will continuously call the PHP table to ensure that the information is constantly up to date. This will be invisible to the user who will only see the information updated as this page is loaded. The user can expand on each garage to view the number of spaces

available on each floor of the garage they wish to view. This will be implemented using another PHP table and the same type of Javascript function.

One of the more important features of the interface will be for the user to view a "busyness" indicator of any garages that are full. This indicator will use different color to show how busy a parking garage is. If the color shown is red, for example, it will mean that the garage is extremely busy and it will be difficult to find an open spot. If the color is green, it means that although the garage is full, there is a good chance that a spot will open up quickly, and the user will not have to wait a long time to find a parking spot.

The "busyness" indicator will be implemented by keeping track of how many spots are remaining on each floor and garage. This will allow the user to see, at a glance, the approximate amount of available parking in each section. They can then make a decision on where to park based on the more detailed information.

The goal of the interface is to provide accurate information quickly to the user. They will be able to log in and immediately see the status of the garages they have chosen. The design will allow them to quickly make a decision on which garage they should go to in order to find an open parking space.

7. 5. 3. 3 Administrator Interface

Administrators will have access to their own application that will allow them to oversee the garages that they own or maintain. This application will be similar to the user interface, but will allow the administrator extra privileges such as setting a garage to closed or open or adding or editing the microcontrollers used in the garage.

The administrator interface will also show the status of the parking garages that they own and any others they choose to see. This will be created using the same method mentioned above for pulling data out of the database. The administrator will have extra privileges that will allow them to change the status of the garages they own.

Administrators will be able to mark a garage as closed or reserved. This will show to the other users that these garages are not accessible, or in the case of reserved garages, the users will know if it is reserved for them or if they must find parking elsewhere. This will be accomplished by using PHP to write to the database. This will reflect if the garage is closed or reserved on other users that are accessing the database. The user can set a time for the reservation or closure to expire, or can manually enter when the garage will be open again.

If there is a special event that is using one of the garages, the administrator can set the status of this garage to reserved. This will show other users that this

garage is not available to them. If a garage is closed for any reason, such as construction, the administrator will be able to make this change as well. They will also be able to do this for each individual floor of their garage in the case that only one of the floors is closed or reserved.

7. 5. 4 Mobile Application

The mobile application will be designed to have the same functionality as the website application. The difference will be that it will take advantage of the touch screen interface and will be built specifically to work on smartphones. The mobile application will be built primarily using Java. This is the language used to build most Android applications.

Both the user and administrator interfaces will be created using Java to query the database. The mobile application will connect to the same database as the website so that the information will be equal across both platforms. The user will be able to touch the garage that they want more information on, and the application will expand to show the information on each floor.

Since people will want to check the status of their garages while driving to them, the mobile application will be designed with speed and safety in mind. The application will remember the user's login data so that they do not have to type in their credentials each time the use the application. A Java method will hold the information and automatically log the user in when they open the application. This means that anyone using the application will be able to get the information they need quickly without the need to take their concentration away from driving.

7. 5. 5 UML

The following UML diagrams show how the application is designed. **Figure 7.11** shows the state diagram as the website is being used. The Use Case diagram seen in **Figure 7.12** shows how both administrators and regular users use the website. Finally, **Figure 7.13** shows a flow diagram of the pages on the website.

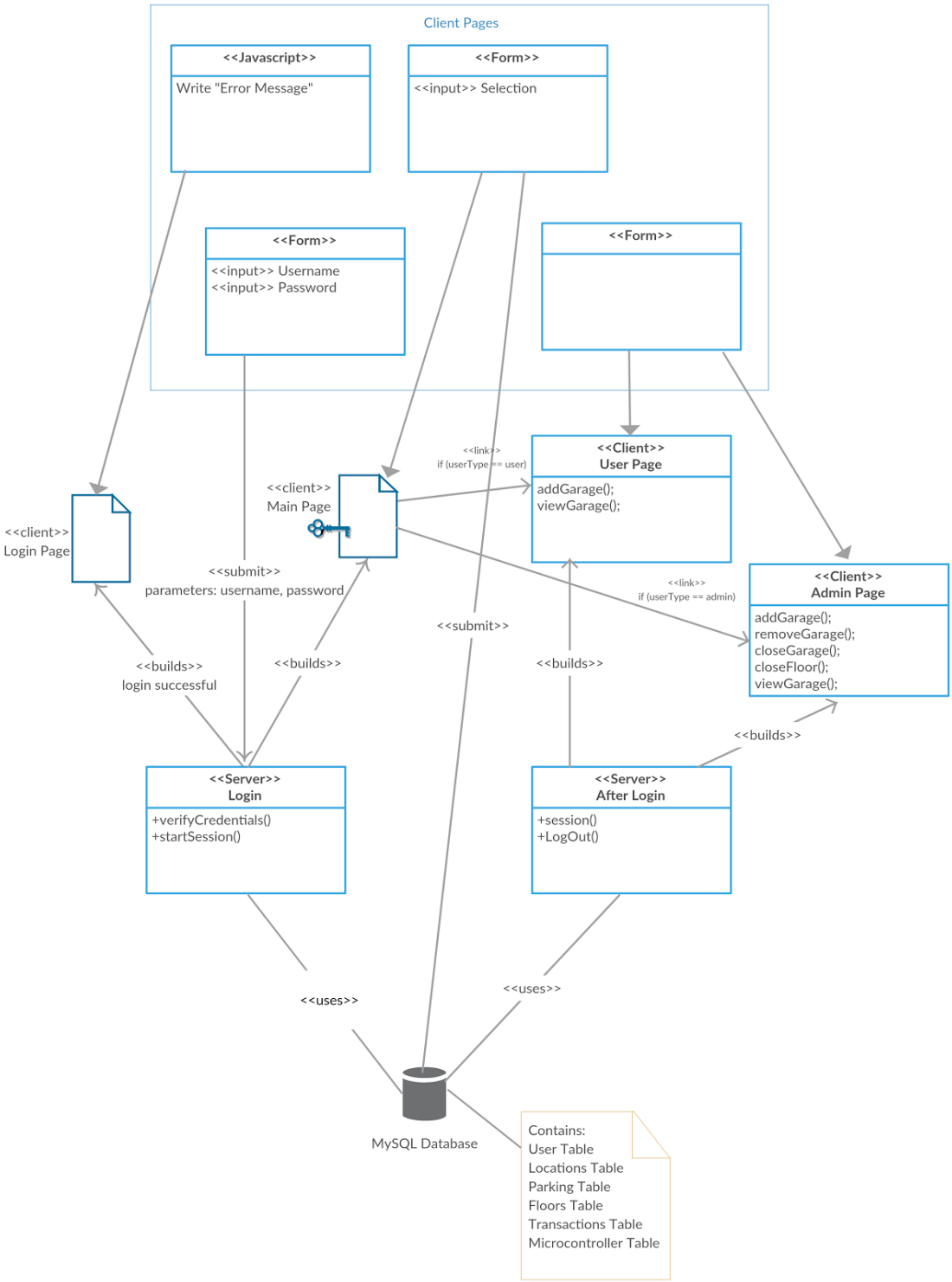


Figure 7.11 - Website State Diagram

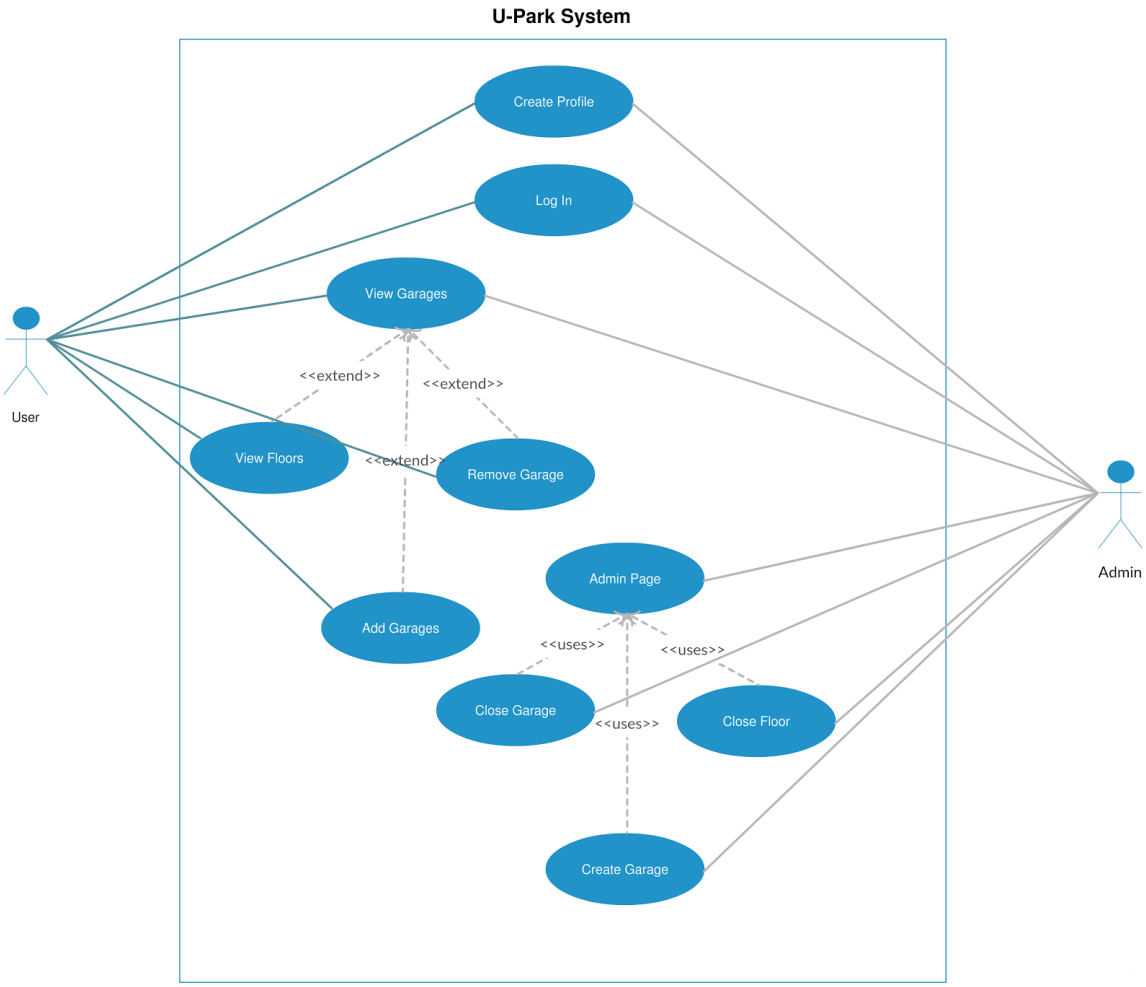


Figure 7.12 - Website Use Case Diagram

Figure 7.13 - Website Page Flow

7. 6 Server

The configuration required to run the application (U-Park) must meet a minimum criteria ample enough to accommodate the operating system, development environment, DBMS environment, DBMS data and the application. Caution must be taken when looking for a hardware alternative, so costs don't exceed what is forecasted.
The selected configuration for the server is:

- Case with a 500W power supply, minimum four 3.5" bays.
- Processor: 64 bit Intel processor with 3Ghz, minimum 8Mb cache, 4 cores/4 tasks.
- Memory:
  - 8 Gb UDIMM, ECC (minimum).
  - 12Gb UDIMM, ECC (desirable).
- Storage:
  - 1 500Gb Hard drive (minimum)
  - 3 500Gb SATA 3 hard drive arranged in a RAID 5 -Block level striping with distributed parity- (Desirable but not required).
- LAN: 1000Base-T LAN Card.
- Ports: 3 USB ports (minimum).
- Display: 1 17" SVGA monitor (minimum).
- Keyboard: US English 101 keys keyboard USB.
- Mouse: USB mouse.
- Operating System: Windows 10 (or newer).
- DBMS: TBA.
- Development platform: TBA.

7. 6. 1 Gateway

In order for the microcontrollers to communicate with the Server, a number of routers are required. The exact number is to be determined based on the quantity of parking spots and the size of a parking garage.

The figure presented in section *2.3. Requirement Specification* clearly indicates how routers interconnect microcontrollers to the server. The idea is to group a number of microcontrollers close to the router. These microcontrollers, once active, will submit information to the router, which in turn will communicate through a LAN -or the Internet, depending on the implementation- to the router located at the server facility. This (server sided) router will communicate to the server, updating the database with the latest parking availability information.

7. 7 Design Summary

7. 7. 1 System Installation and Mounting

Mounting the various components of the U-Park inside the garages presents some challenges. Namely, different locations in the garage will have different locations where the sensors and modules must be mounted. Also, the access points will be mounted in different locations, depending on where space is available. The mounting hardware chosen to mount the components must be attachment securely enough to the garage surface, that will prevent any tampering with the equipment, and ensure that users of the garage are not put in any danger of the various components falling on them or their cars. This section will examine these aspects and provide a detailed overview of how the U-Park system and its components will be installed and secured inside of a parking garage.

Relatively all parking structures are built of steel reinforced concrete. While this can be a difficult material for hardware to “tap into”, the practice is common enough that a multitude of hardware is available to provide a secure and lasting connection. The components mounted on all of the levels of the garages, except those on the top floor, will be mounted to the ceilings of each of the levels. Because the top level of parking garages usually have no ceiling, a slight modification of the sensor modules, and the way the sensor modules are mounted, will be required. On the levels where there is a ceiling, the modules will be mounted to either the ceiling directly, or to the concrete structural “ribs” which extend down from the ceiling. On the next page in **Figure 7.14** and **Figure 7.15** the proposed mounting locations for the modules on the ceiling can be seen. The first figure shows how the sensor modules will be mounted to the “rib structure”, and the second figure shows how the module will be mounted if the ceiling is flat and the structural “ribs” are not present.

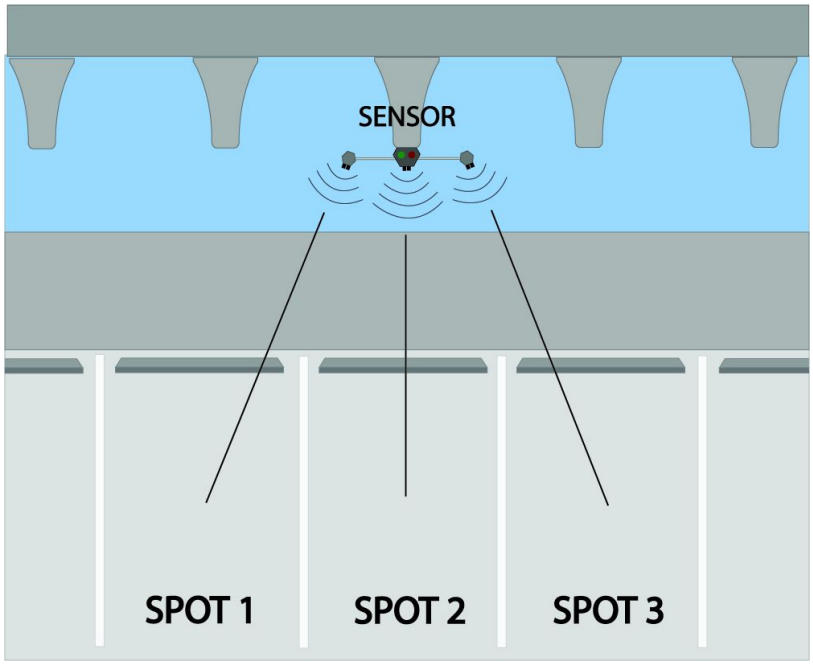


Figure 7.14 - Sensor Mounting Location on a Structurally "Ribbed" Ceiling

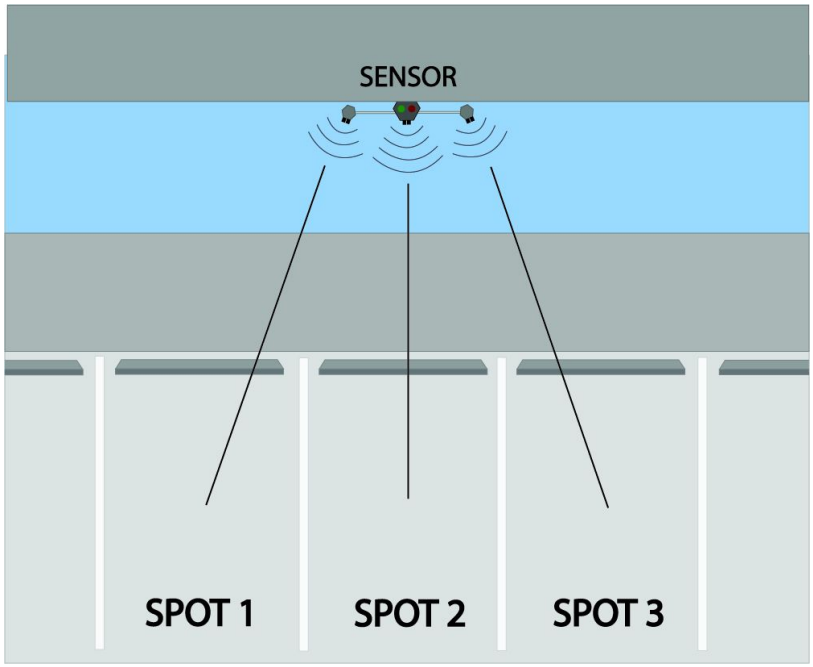


Figure 7.15 - Sensor Mounting Location on a Structurally Flat Ceiling

The sensors mounted on the top level of a garage are unable to be mounted on the ceilings and therefore must be altered so that even from a different perspective, they will still perform the same functionality as those sensor modules mounted on the main levels. The top floor sensors will have the main difference that instead of sensing three parking spaces, these modules will only be able to detect two adjacent spaces. The sensor modules will be located in

front of the spots and will be positioned between two parking spots. The reasoning behind this, and the reason the same three-spot sensor modules will not simply be placed in front of the middle spot is due to a high level of variability in the way people park. People often do not line their car up perfectly in the center of the parking spaces, and will either pull too far forward in a spot, or they will not pull forward enough. In the case of a car pulling too far in, the sensor module runs the risk of being damaged if someone were to hit it. Also, if someone were to pull in too far the side sensors may be blocked and falsely read that the first or third spot is also taken when in reality, it could still be open. Effectively the problem could be that either the sensor could be damaged, or the system could be inaccurate, both of which do not translate into a quality final product. This is especially important in a product such as this, which relies on robust hardware and accurate capacity reporting to work. To this end, placing sensors between the spots on the top level minimizes the risk of being hit by a car and also eliminates the possibility of a car “taking up two spaces” by crossing the threshold of two sensors. **Figure 7.16** below demonstrates this method and shows approximately where a sensor module would be placed on the top floor of a garage.

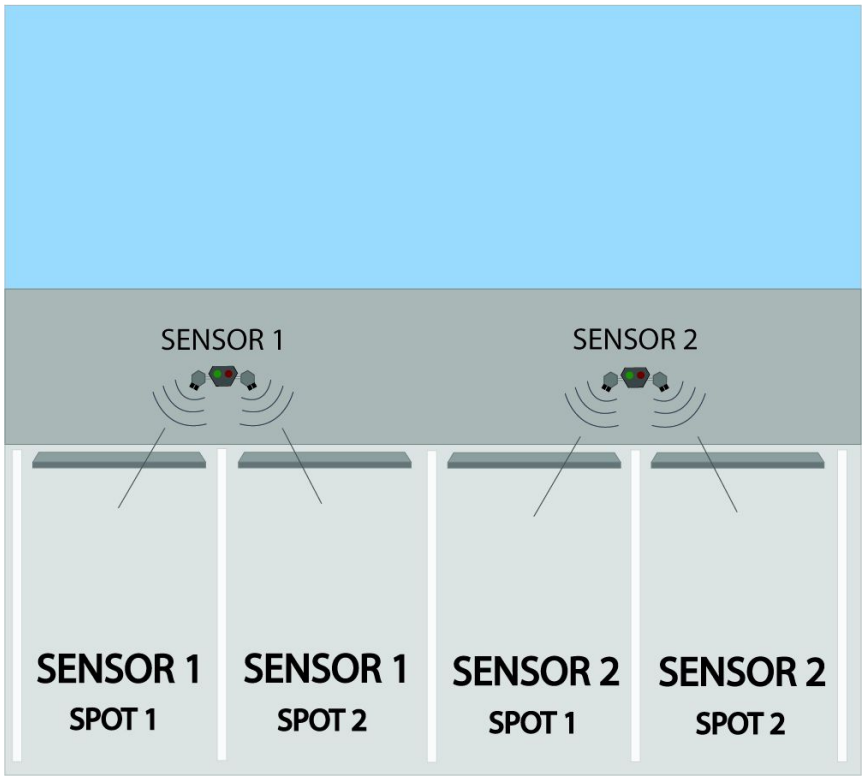


Figure 7.16 - Sensor Mounting Location on the Top Level Without a Ceiling (Wall Mount)

One final problem that mounting on the top floor of a garage would pose has to do with weather. Because all of the components will be exposed to the elements, special attention needs to be paid to selecting weather resistant access points and ensuring that the modules themselves are shielded from the elements. While

these minor changes do require slightly higher costs than are required to implement the system on lower levels in the garage, this is a necessary cost and will ensure the best reliability of the U-Park system.

7. 7. 2 Network Infrastructure

The network for the U-Park system is made of of a multitude of inter-connected components. In this section, the structure of how these components are connected and how the components work is to be discussed. There are two separate network infrastructures that are used in parts of the project and integrate together to create the backbone of the communications network.

The first network structure used by the U-Park system is a star network. This network model is discussed in more detail in section **4.4.1.3**. In this section, the specifics of this network type were discussed from the theoretical perspective of initial research. The benefits of this network topology are that all of the devices are connected to a single access point, and that routers are set-up to work in this configuration by default. However, a problem with this type of topology is that if the router “goes down” or faults, all of the clients connected to the access point are unable to transmit data to the server.

The router serves as the access point to the parking sensor modules. The modules communicate over the Wi-Fi transceivers integrated into the sensor modules. The router can support a maximum of about 10 clients due to the fact that there are only a limited number of independent channels to be used by the 2.4GHz Wi-Fi devices. The number of connected devices must be limited by the number of channels for each router to avoid any interference. Also, to avoid interference between access points across the garage, the access points must be separated by a sufficient amount of physical space. This minimizes the noise received from other Wi-Fi access points.

Because signal from the antennas of a router emanate in a circle surrounding the router, each router/access point needs to be position just out of range of other routers. This will be accomplished by installing a router on each of the four corners of the garage and in the center of the isles of the garage on the long sides of each floor. This means that a total of six routers will be used per floor is the theoretical garage shown on the next page in **Figure 7.17** and in **Figure 7.18**. The power supplied to these routers must also be adjusted accordingly to ensure that the signal strength is low enough to only be able to “see” the modules that are connected to the particular access point. Below, **Figure 7.17** shows the proposed locations of the access points inside of the the theoretical garage on each level. The black dots and the blue rings show the routers and the coverage of the routers respectively. As the figure indicates, the coverage overlaps slightly, however this is unavoidable due to the fact that all spots in the

garage must be covered by at least one router to be able to communicate with the access point.

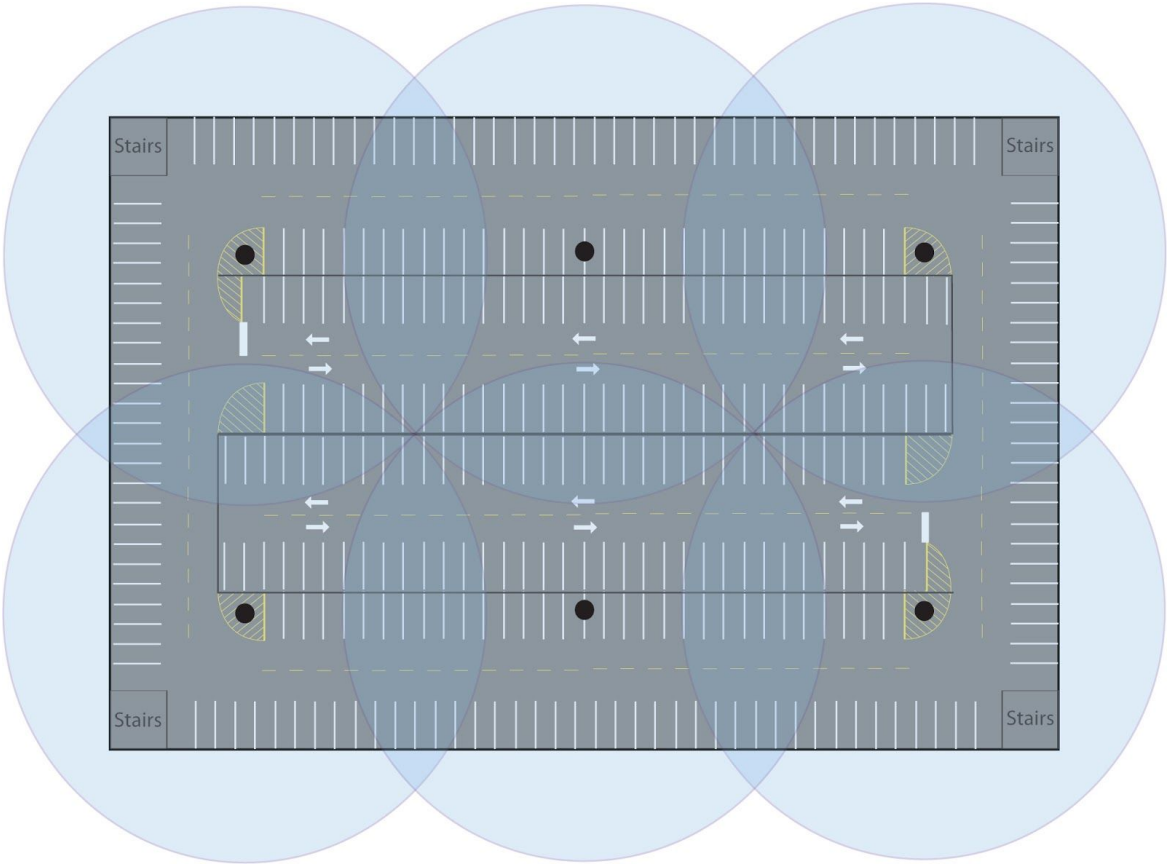


Figure 7.17 - Proposed Locations of the Access Points on one Floor of the Garage

Because 2.4GHz passes through walls (and floors) with relative ease, interference between the access points between the concrete floors is not entirely avoidable. One benefit, however, of the construction of the garages is that even though 2.4GHz WIFI does pass through walls in a house or office building, the ability for the signal to due so in a garage is reduced due to the fact that all of the walls and floors are made of steel and concrete which reduces the permeability of the 2.4 GHz signals.

Connecting the router access points to one another is done with the use of ethernet cables. The routers are out of range of each other as to not interfere. While this would increase the overall system cost due to the added cost of the quality ethernet cables which would be required, the added reliability and robustness of a wired connection would be a worth-while trade off. Because the system is completely reliant on all components in the system functioning

correctly, the reliability of the connections is an important aspect of the network design.

The routers are arranged in a Mesh network configuration. This configuration has an added benefit in that it is able to “self heal”. This means that traffic is routed in the most efficient way through to the exit node of the system, but also that if one of the connections between two nodes fails, the network protocol is able to reroute packets around the broken connection.

In **Figure 7.18** below, the proposed configuration of the routers connected via ethernet cable (red lines) into a mesh network topology is shown. While garages come in all sorts of different sizes and shapes, the garage shown in the figures above and below are modeled after parking garage C at UCF. Bidirectional traffic is possible on both ramps and the garage is very symmetric, which makes designing the network slightly simpler.

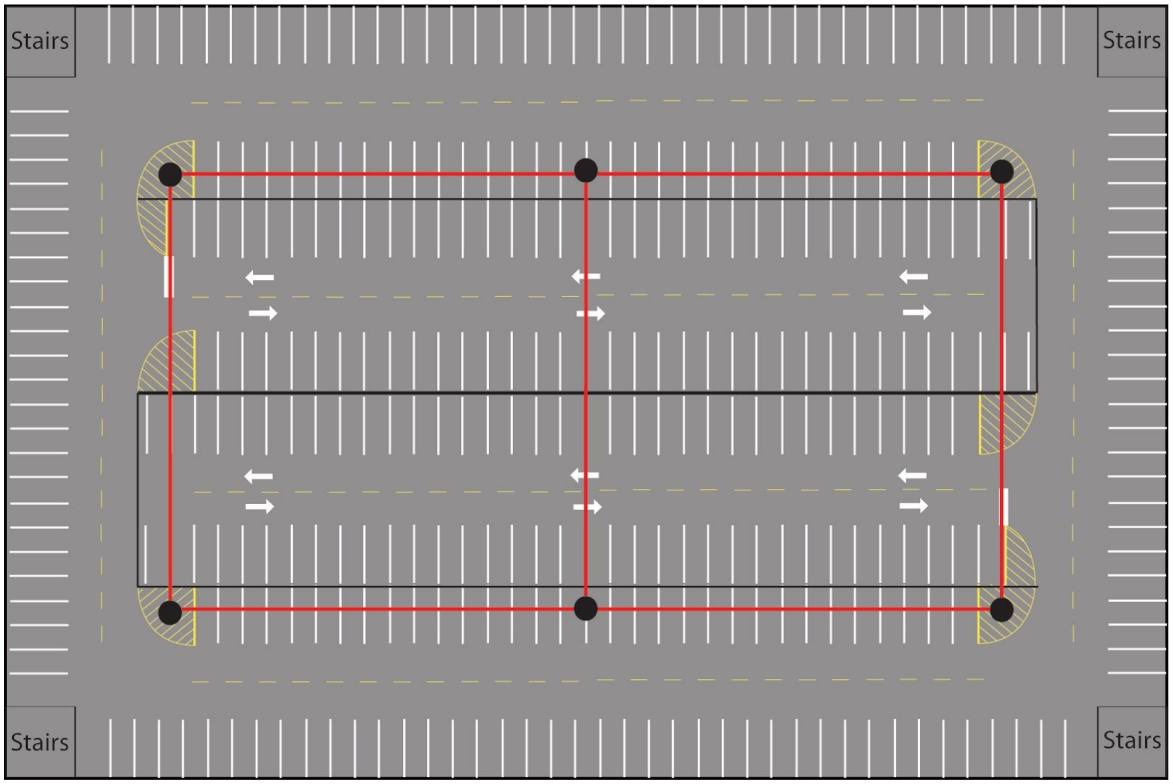


Figure 7.18 - Proposed Connections Between Access Points on one Floor of the Garage

The network described above, built using the wireless Star network topology at the sensor (client) end and a wired Mesh network topology between the access points seems to be the most effective way for the team to create a network that is both reliable and simple. To over-complicate the network could prove to be detrimental to sustainability and could result in system failures. With a simple system, if system failures are to occur, troubleshooting the failures will be much

simpler. After all, a parking management system isn't of much use if the system is plagued with system failures, and requires specially trained technicians to fix the problems.

7. 7. 3 Database Design

After evaluating pros and cons of both MySQL and MS SQL Server, the team opted to use MySQL as the DBMS for the project. Immediately, the design phase of the database began. The project is pretty simple: To obtain the status of a parking spot at any time, record it, and make it available to three different user types: Admin, monitors (operators who routinely checks the system looking for errors), and standard users (people looking for a place to park)-.

In order to obtain parking information, team nine has designed a microcontroller capable of monitoring three parking spots at a time, and transmit the acquired data via Wi-Fi to a router; which in turn will transmit the data to a server. The problem consists in how to organize the microcontrollers' modules in a way that they can be associated to a particular client, in this case UCF, and within the client; where the parking is located.

Additionally, (and for security reasons) access to data must be controlled, and a log file for transactions (for statistical purposes) is required. The resulting schema for the database is presented in **Figure 7.19**.

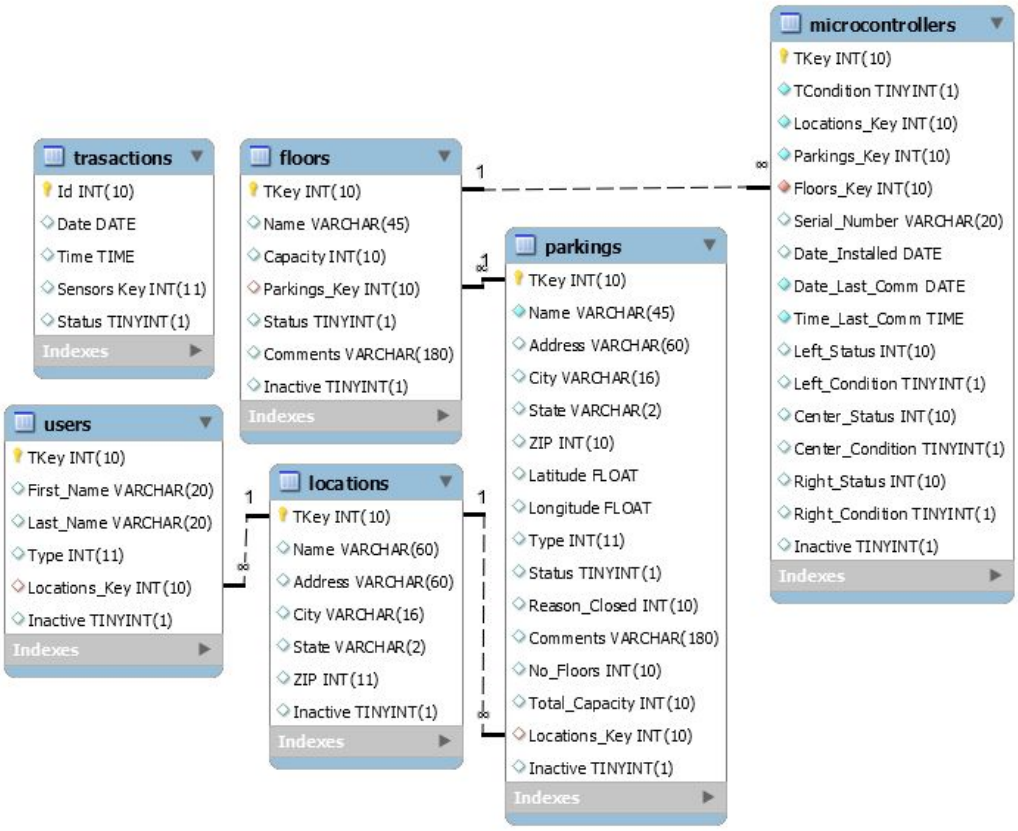


Figure 7.19 - U-Park Database Relational Data Model

Considerations about the Schema:

- Columns and tables are named Cxxx (Where ‘C’ is the first letter of the name capitalized, and xxx and the rest letters non-capitalized). When a column contains in its name more than one word it will be constructed as Cxxx_Cxxx, or Cxxx_Cxxx_Cxxx depending on how many words.
- For attributes for columns, please refer to the schema depicted in **Figure 7.19** above.
- The following consideration do not apply for table *Transactions*.
- Due to the fact that the database is organized in a hierarchical way, and to maintain data integrity, no records are to be physically deleted, so with the exception of the transactions table, all tables contain a column called *Inactive*, which will be false for active rows (records), or true for logically deleted rows.
- In other to establish a reliable way to link tables the following having been established.
- Primary keys *Tkey* are an integer number self-generated (auto-increment) by MySQL, therefore no user or programmer interference is required. The exception is for table *Microcontrollers* which for information purposes the key of the row must be presented to the user before actually inserting it into the table, therefore the auto-increment process is programmatically performed before row insertion.
- A foreign key is established between a table and its ‘parent table’, being *Locations*, the main and first in the hierarchy. Foreign keys are set to:
    - Take no action In case of accidental deletion.
    - Cascade in case parent key is changed.

  (Both cases unlikely to happen using the applications written by team nine).
- The *Transactions* table is considered merely a sequential file where rows are stored as they are inserted. Later, they can be organized –at run time- depending on what information is desired.

7. 7. 3. 1 ER Diagram

The entity relationship diagram for the database (SPOT) used in UPark is extremely simple, and consists of a hierarchical design. On the top of this hierarchy if the table *Locations*, the child tables to the *Locations* table are: *Users* and *Parkings* (For more information on tables refer to 7.7.3.3 My SQL tables). In all cases, except table transactions, the relationship Parent/Child is One-to-Many. Therefore, there was no need for inserting *relationship entities* as in the mandatory case of Many-to-Many.

To start, **Figure 7.20** shows the *entity relationship* between *Locations* and *Users*, where a location may contain 1 or many users. On the other side of the *Locations* table *Locations*, **Figure 7.21** shows how *Parkings* (garage) is related to

Locations. Again this is a One-to-Many relationship (one *Location* that contains one or many parking garages). Next, **Figure 7.22** connects the table *Floors* to *Parkings* (see connector 'A'), and table *Microcontrollers* to *Floors*.

Finally, **Figure 7.23** shows table *Transactions* on its own. Even though the hierarchical schema could be drawn down to this table, it is omitted. Its use is for recording history transactions, which may be later retrieved for statistical purposes, and its relation to table *Microcontrollers* does not require a direct link.

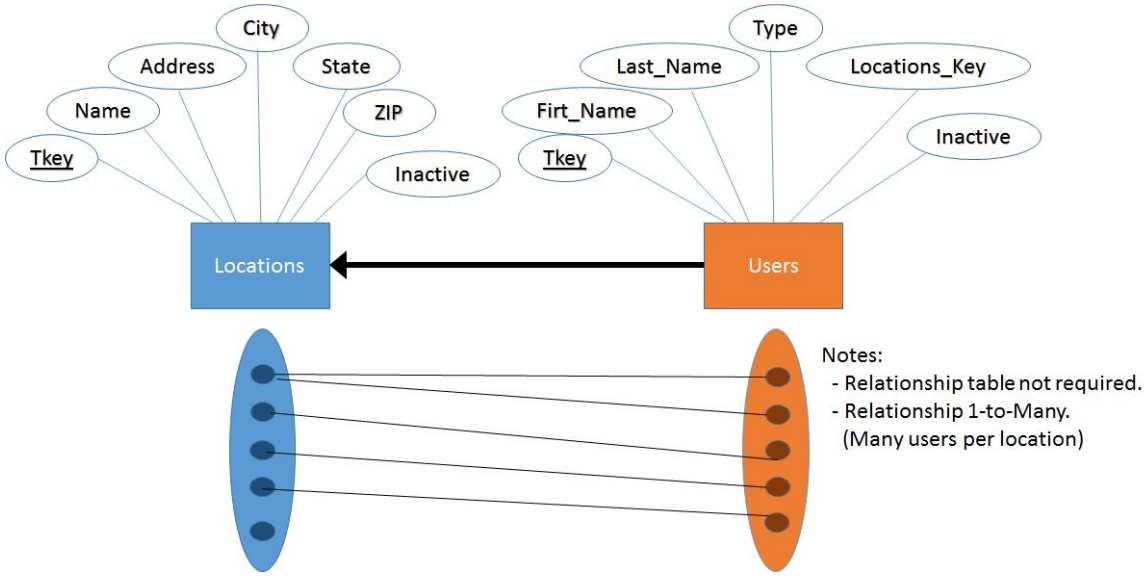


Figure 7.20 - ER Diagram 1 of the U-Park Database

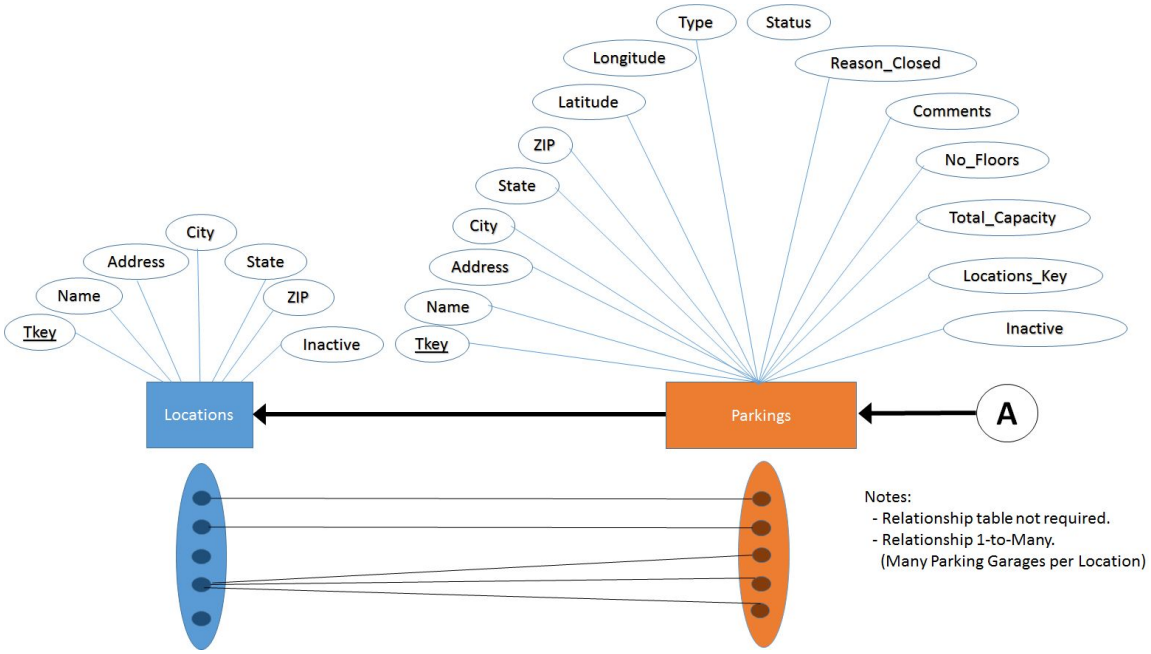


Figure 7.21 - ER Diagram 2 of the U-Park Database

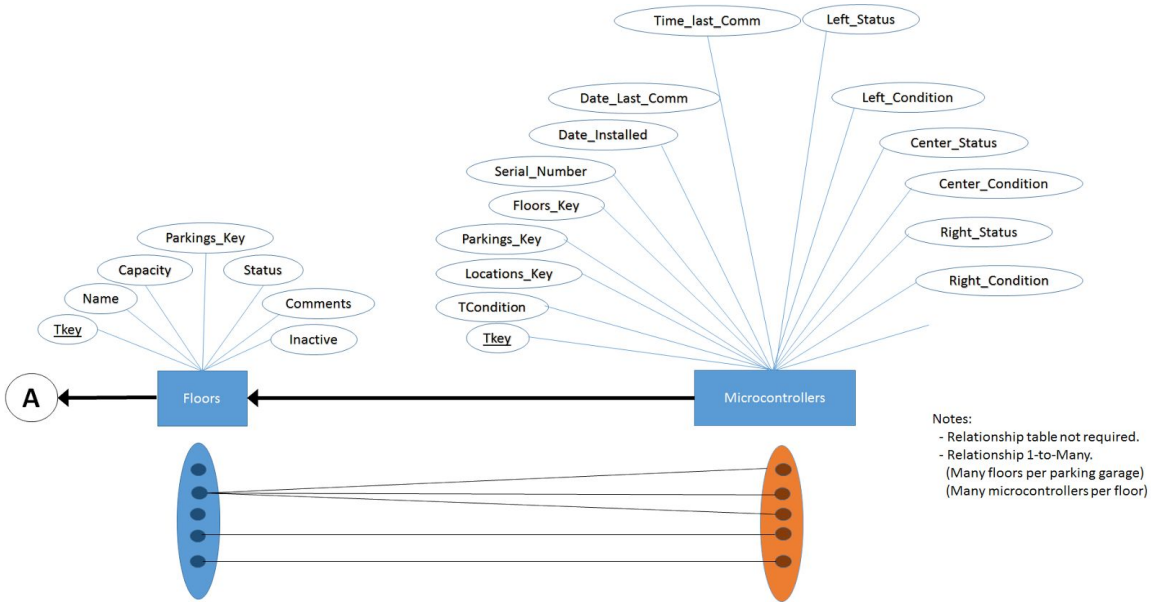


Figure 7.22 - ER Diagram 3 of the U-Park Database

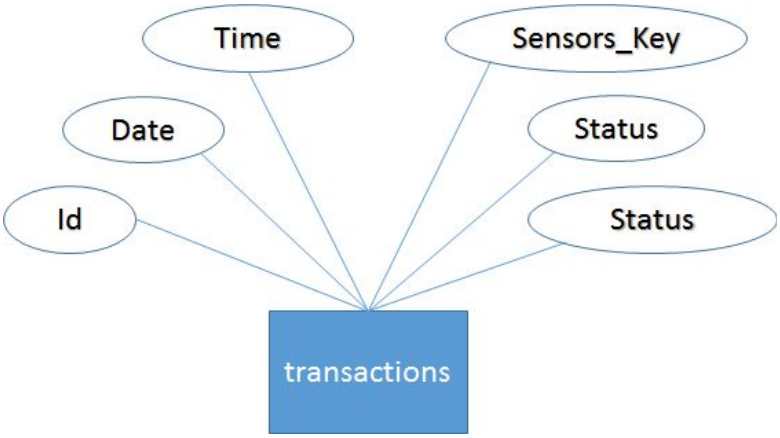


Figure 7.23 - ER Diagram 1 of the U-Park Database for the Transactions Table

7. 7. 3. 2 mySQL Tables

Table 7.5 - Floors: This table holds the data (capacity and status) of a given floor within a parking garage. Even though is not a relationship table, it can be seen as a link between table *Parkings* and *Microcontrollers*.

Column	**Type**	**Use**
TKey	Integer	Primary key of the table. Not known to the user or programmer, but used by MySQL to establish a key with daughter tables.
Name	Varchar(60)	The name of the floor. By floor it's understood one of the areas (or layers) the parking (a basement, or a parking building) has. Example: Floor # 2.
Capacity	Integer	The maximum number of vehicles (excluding motorcycles) that can be parked at a time.
Parkings_Key	Integer	Foreign key pointing to the parent row in *Parkings.*
Status	Integer	This column indicates if the parking is: 0 = Open. 1 = Closed. More and better detailed information may be obtained in the *Status* column for the *Parkings* table.
Comments	Varchar(180)	Any comment the administrator considered valuable.
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)

Table 7.6 - Locations: This table contains the information about the person or institution giving parking access to *Standard Users*.

Column	**Type**	**Use**
TKey	Integer	Primary key of the table. Not known to the user or programmer, but used by MySQL to establish a key with daughter tables.
Name	Varchar(60)	The name of the location. By location it's understood the entity who own the parking area, and offers the information, so clients can locate a parking spot in the shortest time possible. Example: UCF.
Address	Varchar(60)	The address for the location's main offices.
City	Varchar(16)	Complement to *Address.*
State	Varchar(2)	Complement to *Address.*
ZIP	Integer	Complement to *Address.*
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)

Table 7.7 - Microcontroller: This can be considered the main table of the hold applications, for it holds the actual status of the parking spots. Each microcontroller has three sensors; left (1), center (2), and right (3). Every thirty seconds (default) the status of each physical sensors is sent to the database for updating.

Column	**Type**	**Use**
TKey	Integer	Primary key of the table. Known to the user, so he/she can associate to a physical microcontroller, by hardcoding this key into the microcontroller (See C language sketch).
TCondition	Boolean	This column indicates if the microcontroller Is: 0 = Working. 1 = Not Working.
Locations_Key	Integer	Quick reference to find the grand-grand-parent row at table *Locations*.
Parkings_Key	Integer	Quick reference to find the grand-parent row at table *Parkings*.
Floors_Key	Integer	Foreign key pointing to the parent row in table *Floors.*
Serial_Number	Varchar(20)	The serial number of the physical microcontroller whose data is recorded in this row.
Date_Installed	Date	The date the microcontroller was activated, and started working.
Date_Last_Comm	Date	The last time the microcontroller sent data. In the format of DATE.
Time_Last_Comm	Time	The last time the microcontroller sent data. In the format hour, minutes, second.
Left_Status	Integer	The *Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (*)

Table 7.7 (cont'd.)

Left_Condition	Boolean	The *Condition* can be: 1= Working. 2 = Not working (damaged).
Center_Status	Integer	The *Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (*)
Center_Condition	Boolean	The *Condition* can be: 1= Working. 2 = Not working (damaged).
Right_Status	Integer	The *Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (*)
Right Condition	Boolean	The *Condition* can be: 1= Working. 2 = Not working (damaged).
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)

Table 7.8 - Parkings:

Column	**Type**	**Use**
TKey	Integer	Primary key of the table. Not known to the user or programmer, but used by MySQL to establish a key with daughter tables.
Name	Varchar(60)	The name of the Parking. By *Parking* it's understood the physical location a vehicle is to be parked. Example: Parking garage A.
Address	Varchar(60)	The address for the parking.
City	Varchar(16)	Complement to *Parking.*
State	Varchar(2)	Complement to *Parking.*
ZIP	Integer	Complement to *Parking.*
Latitude	Float	Stores the latitude in degrees of the parking's location. Used in combination with *Longitude* to retrieve (via Google API) a map of the parking's surrounding area.
Longitude	Float	Stores the latitude in degrees of the parking's location. Used in combination with *Latitude* to retrieve (via Google API) a map of the parking's surrounding area
Type	Integer	Indicates if the parking is: 0 = In a basement. 1 = In a building. 2 = in a lot.
Status	Integer	Indicates if the parking is: 0 = Open. 1 = Closed.

Table 7.8 (cont'd.)

Reason_Closed	Integer	Indicates the reason why the parking is closed. Reasons can be : 1 = Maintenance. 2 = Security. 3 = Special Event. 4 = Sports. 5 = Others (to be specified on Comments)
Comments	Varchar(180)	Any comment the administrator would like to add regarding the parking. This is of importance when the parking is closed due to *Others*, for monitors, and standard users will know the specific reason on why it's closed.
No_Floors	Integer	In case the parking is a basement or a building, this column will indicate how many floors are allotted for parking purposes.
Total_Capacity	Integer	Indicates the total number of parking spots available in the parking. This is updated by add *Floors*, for each row in the Floors table contain a *Capacity* column. It's main value if for speeding up the process of determining how many empty spaces remain in the parking.
Locations_Key	Integer	Foreign key pointing to the parent row in table *Locations.*
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)

Table 7.9 - Users:

As its name indicates, this table contains all information pertaining to the users who are going to access, monitor, or administer the system.

Column	**Type**	**Use**
TKey	Integer	Primary key of the table. Not known to the user or programmer, but used by MySQL to establish a key with daughter tables.
Login	Varchar(15)	This is the key used by the user to identify him/herself to the system.
Password	Varchar(15)	Validation for the Login. It is case sensitive, not less than six digits, and must contain at least one number and one special character.
First_Name	Varchar(20)	The user's first name.
Last_Name	Varchar(20)	The user's last name.
Type	Integer	The *Type* can be: 0 = Administrator. 1 = Monitor. 2 = Standard user.
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)

(*) As a reading error may occur at any time, many reading errors will be used to consider that the sensor is not working at all, and needs to be replaced. A damaged sensor is set in the next (XX_Condition) column.

Table 7.10 - Transactions: A sequential recorded, accessed table containing a log file of all the updates occurred at the *Microcontrollers* table

Column	Type	Use
Id	Integer	A sequential number generated by MySQL used merely as an identifier of the transaction (row).
TDate	Date	The date when the transaction occurred.
TTime	Time	The hour, minute, and second when the transaction occurred.
Microcontrollers_ Key	Integer	Not a foreign key though, but a reference to the microcontroller affected by this transaction.
Left_Status	Integer	The value for *Left_Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (**)
Center_Status	Integer	The value for *Center_Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (**)
Right_Status	Integer	The value for *Right_Status* can be: 0 = Empty spot. 1 = Occupied spot. 2 = Read Error. (**)
Inactive	Inactive	Boolean to determine is the row is logically deleted (True) or active (False)
(**) Refer to *Read Error (*)* in table *Microcontrollers.*		

8. Prototype

8. 1 Microcontroller

The main purpose of this section is to discuss the process of prototyping the microcontroller. There are two types of testing boards that can be used to prototype the microcontroller: breadboard and through-hole proto-board. Team nine prototyped the microcontroller on the breadboard to make sure that it worked first before all of the components were soldered together on the prototyping board.

The purpose of using the proto-board is to test out a standalone microcontroller in a sleeker way than a breadboard. The components will also not fall off, which is also a benefit.

There are two options for bootloading the ATmega 328. The first method, using Arduino board and an AVR programmer, is quite easy. The second method is bootloading the chip on breadboard and AVR programmer. Both of the options will be discussed.

Using an Arduino board:

- Place the ATmega328 into the Arduino board
- Connect the jumper to the external power supply.
- Attach the 6-pin female plug of your AVR programmer to the 6 male header ICSP pins with the plastic nub of the ribbon cable head facing inward.

Using a breadboard:

First of all, to power the microcontroller, a voltage regulator is needed to make sure that the voltage supply to the microcontroller is stable and is not over the maximum voltage that the microcontroller is rated.

- Add the TL2575 switching regulator and the lines to power the board. Input from the external power supply goes into pin one of the regulator. Connect Cin (100 uF) from pin one to the ground.
- Pin three is the ground pin.
- Pine five is ON/OFF pin so it is connected to the ground.
- Pin two is output pin. Connect a diode from pin two to ground.Connect an inductor from pin two to a capacitor Cout and then ground Cout.
- Pin four is the feedback so it needs to be connected to the same node as the inductor and Cout capacitor.

Second, connect all the components to the microcontroller ATmega328. Before start connecting, it is important to read the ATmega328 pin mapping; this pin mapping provides information about each pin of the ATmega328

- Start by connecting 10KΩ resistor from +5V to pin number 1 (reset pin), the purpose is to prevent the ATmega328 reseting itseft while operating.
- Add a small tackle switch close to the reset pin. Add a wire from the bottom left leg to the reset pin of the chip and a wire from the top left to the ground. This switch will reset the board whenever we like and also prepare the chip for uploading program.
- Add a 16 MHz external clock between pin 9 and 10, and add two 22 pF capacitors running to ground to pin 9 and 10
- Ground pin 8
- Connect pin 7 to Vout in the voltage regulator
- Ground pin 22
- Connect pin 20,21 to +5V

Figure 8.1 below shows the schematic of the microcontroller.

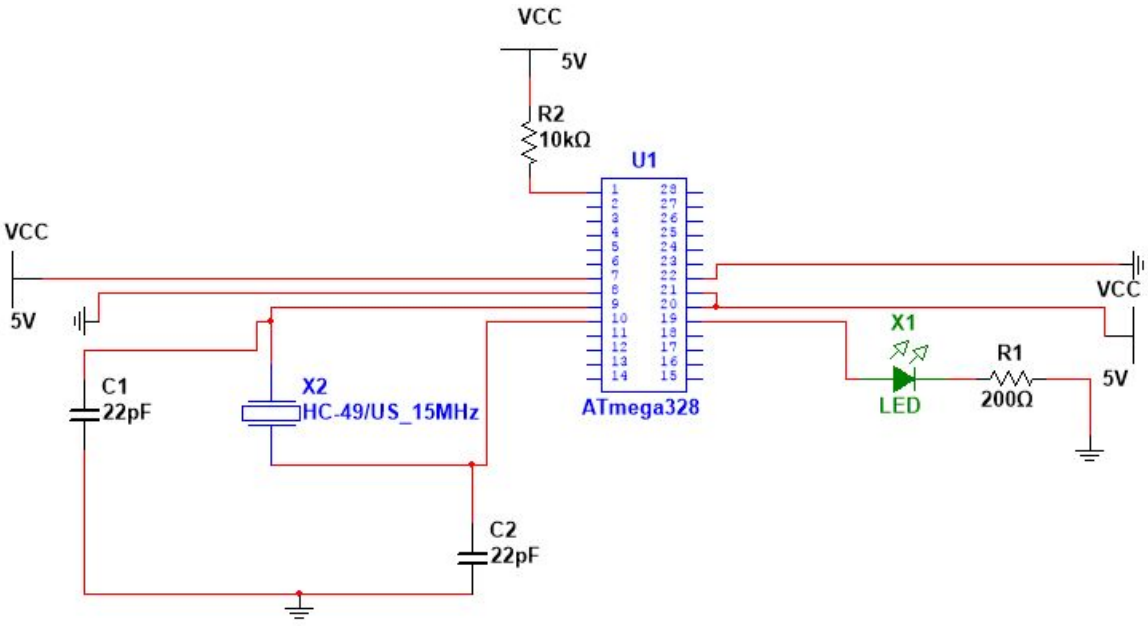


Figure 8.1 - Schematic of ATmega328p-pu

To test if the board is working, team nine used an ATmega328 that is already programed. To demonstrate that the board is working, the blink test code was uploaded to the microcontroller. The longer leg (anode) of the LED is connected to pin 19, and the shorter leg (cathode) is connected to a 200Ω resistor and ground the other leg of the resistor. Now if everything is connected correctly, when supply power for the board, the LED should blink. **Figure 8.2** is the blink-test flowchart.

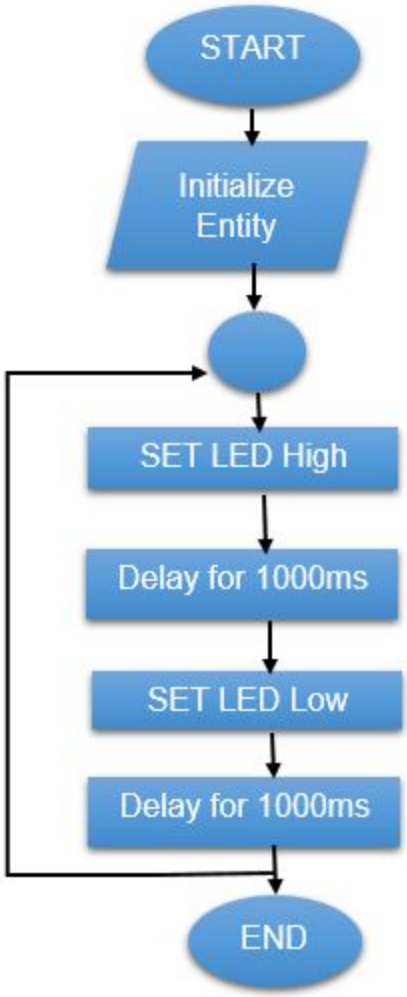


Figure 8.2 - Blink LED Flowchart

Second method:

After all of the connections on the breadboard have been made, the second method to boot-loading the microcontroller, is by using an AVR programming adapter. This adapter breaks out the six pins on the programmer to six inline pins making it easy to connect to the breadboard. **Figure 8.3**, on the top of the following page, shows the AVR programming adapter.

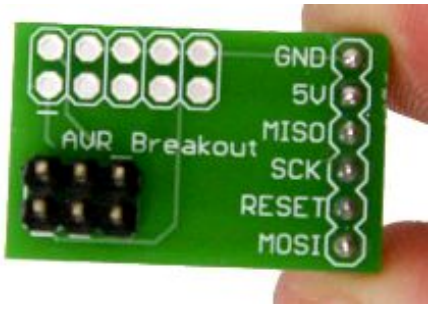


Figure 8.3 - AVR Programming Adapter

- Add USB to Serial breakout board to breadboard.
- Connect the VCCIO of the breakout board to the +5V and Ground to ground of the breadboard.
- Connect the AVR adapter such that the 5V pin is connected to the +5V and the ground is connected to the ground of the board.
- Use wire to connect the MISO pin to pin 18 of the ATmega.
- The SCK will be connected to pin 19.
- The Reset pin goes to pin 1.
- The MOSI pin goes to pin 17.
- Plug USB cable to USB breakout board to get the breadboard connected to the computer.
- Plug the 6-pin plug of your AVR programmer to AVR programming adapter.The black nub of the 6-pin head must be facing upwards towards the Atmega chip

Using Arduino software to burn bootloader:

- Fire up Arduino. Go to Tools and then Boards and then choose the type of board.
- Go to Tools and Burn Bootloader and choose the programmer.
- The AVR programmer will start bootloading the ATmega328.
- When bootloading is finished, a message will appears on status bar say “Done burn bootloader.” The ATmega is now ready to be programmed using Arduino software.

Now that the ATmega is bootloaded, the AVR programmer and the USB serial can be taken out from the breadboard. Since the board is powered by external power supply, the positive power lead is connected to the first leg, and the negative power lead is connected to the ground leg of the voltage regulator.

8. 2 Sensors

The means by which the ultrasonic sensor is connected to the microcontroller will be discussed in this section. As mentioned in the previous section, the ATmega328 pin mapping is really important to know, because it describes the role of each pin. The **Figure 8.4** below is of the HC-SR04 ultrasonic sensor. Which is the team's choice for a distance detection sensor.

Figure 8.4 - Ultrasonic Sensor (HC-SR04)

- Connect Vcc to +5V of the voltage regulator.
- Ground pin to ground.
- The Trigger pin is connected to digital pin and Echo pin is connected to digital pin PWM.

When measuring distance, code is uploaded to the ATmega328, turn on the switch. To test if the ultrasonic is working, just simply place an object in front of the ultrasonic sensor. The console on the computer will display the distance of the object from the sensor. If the object is moved further away, the distance display in the console will change according to the distance of the object. If the object is placed in a range that the sensor can not detect, the console will display "The distance is not within the range."

8. 3 Software

8. 3. 1 Web Application

The U-Park web application prototype will be a full instance of the proposed website application. It will allow for a full demonstration of normal use of the web application. It will be running on a laptop, tablet, and mobile phone to demonstrate the scalability of the application.

team 9 will be able to demonstrate to the users how to create a username and password and login to the system. Once a user has logged in, the garage that will be viewable will be the prototype model used for the demonstration. Users can watch in real time as the application is updated as cars are moved in and out of the parking spaces on the model. The application will also demonstrate the ability of the admin to close and reserve garages. These attributes can be changed on one device, and the other devices will reflect the change.

In addition to the standard website application, there will be a driver built to simulate different parking states. This will be something that Group 9 will be able access and change the states of parking garages. This will allow for the demonstration of the "busyness" indicator and also show the states of a large number of garages. This will demonstrate how the application would run in the real world.

8. 3. 2 Mobile Application

The mobile application prototype will have all the functionality mentioned above in the website application. It will be running on an Android mobile phone to demonstrate the use of the touch screen and immediate log in features of the mobile application.

Group 9 will be able to demonstrate all the same aspects of the web application working on a mobile phone. Users can be registered and logged in on the sample phone. This application will update at the same time as the running website application when the spots are changed on the model garage. It will also be able to access the simulation that will show how the application will be used on a broader scope than can be demonstrated with the model.

8. 4 Database (and Server Application)

As previously stated, the team selected MySQL as the DBMS engine, and Visual Basic 2015 for the development of the server application. Prototyping the database consisted in two steps:

1. Creating a database prototype from the schema are shown in **Figure 8.5** and **Figure 8.6**.
2. Designing, and writing the server application that will be used by administrators and monitors to load the initial data and watch over it.

The first step –Creating the database- consisted in using the MySQL workbench to create a database from the schema created in 7.6.3. There are seven steps in this process.

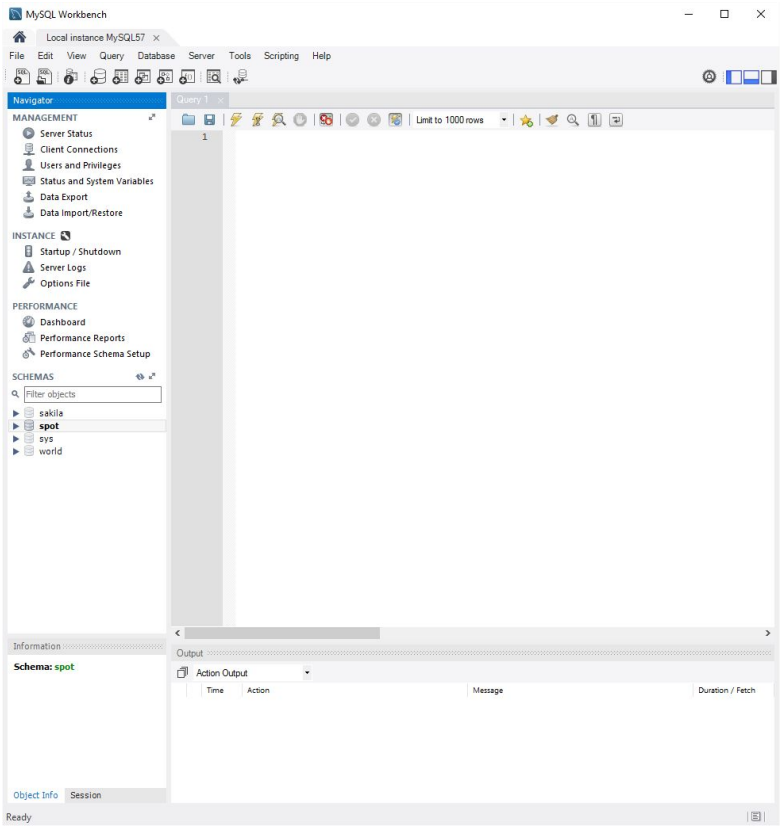


Figure 8.5 - Selecting the Schema

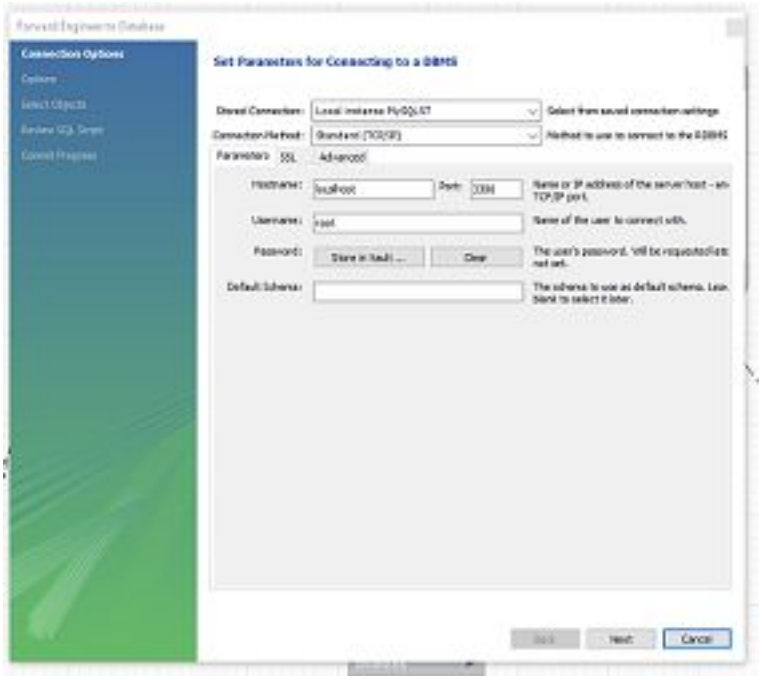


Figure 8.6 - Set Parameters to Connect to the DBMS

The first step is choosing the schema from MySQL workbench. (1), Then clicking on *Database,* in the designer select *Forward Engineer* (2). The next step (3) consists of setting the parameters for connecting, they include the use of specific TCP ports, the users connecting and SSL certification parameters. On step 4, the designer is asked if options for the creation of keys or foreign keys will be included. This is followed by a screen (5) requesting for objects to create. Next on (6), the designer is given the opportunity to revise the auto-generated script that will create the database. Finally, the database is created, and the designer is informed as this is happening (7).

Once the prototype of the database is created, the designer proceeds to enter the basic data needed to perform. To accomplish this task, the server application will be used. As for the server application, it performs two tasks; update/inquire database tables, and monitor the parking availability.

- Update/Inquire of database tables (which are performed by the administrator).
- Once completed the login process, the user is presented with the main screen as shown in **Figure 8.7**.

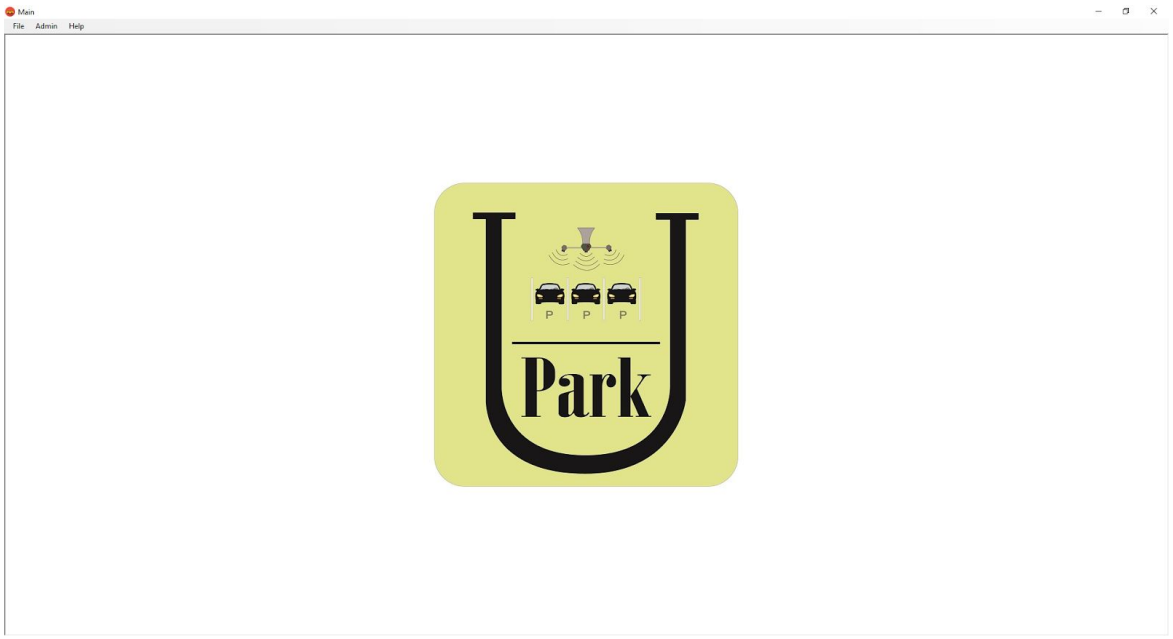


Figure 8.7 - Main Application Screen

The required sequence to follow for data updating is shown below in **Figures 8.8(1)** through **Figure 8.8(6)** :

1. Create location(s).

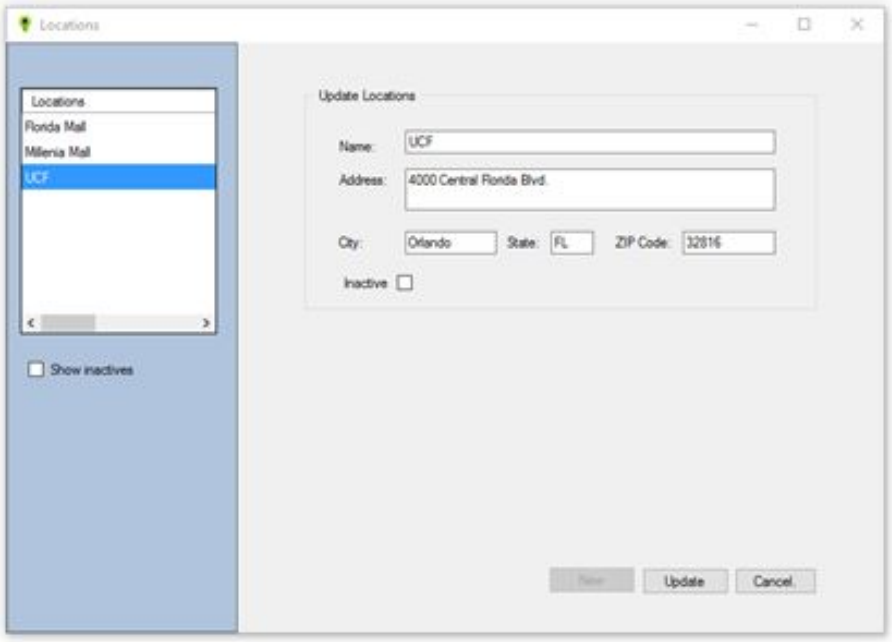


Figure 8.8(1) - Update Locations Screen.

2. Create a ‘Parking unit(s)’ within a given location.

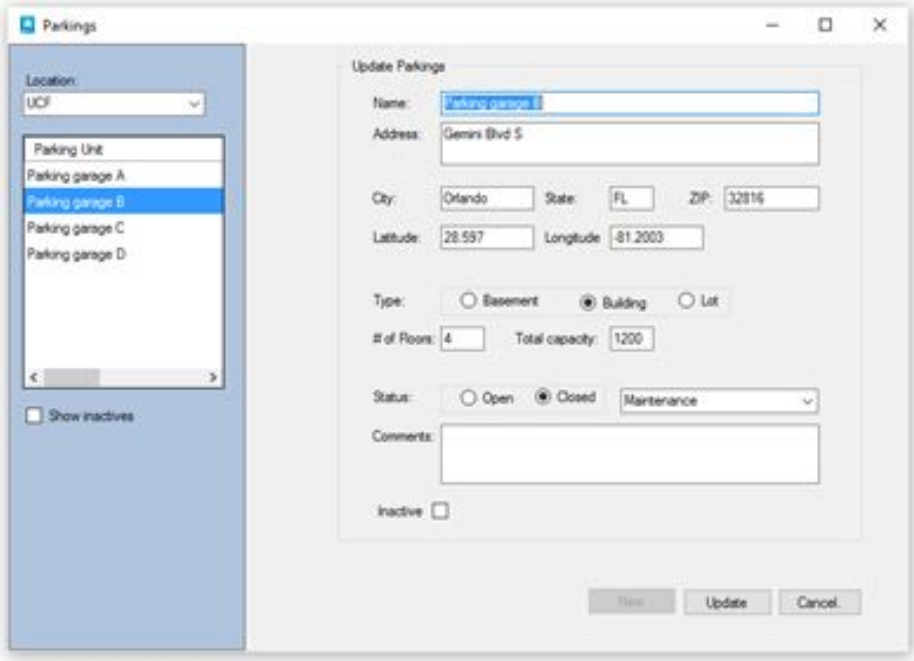


Figure 8.8(2) - Update Parkings Screen.

3. Create floor(s) within a ‘Parking unit’.

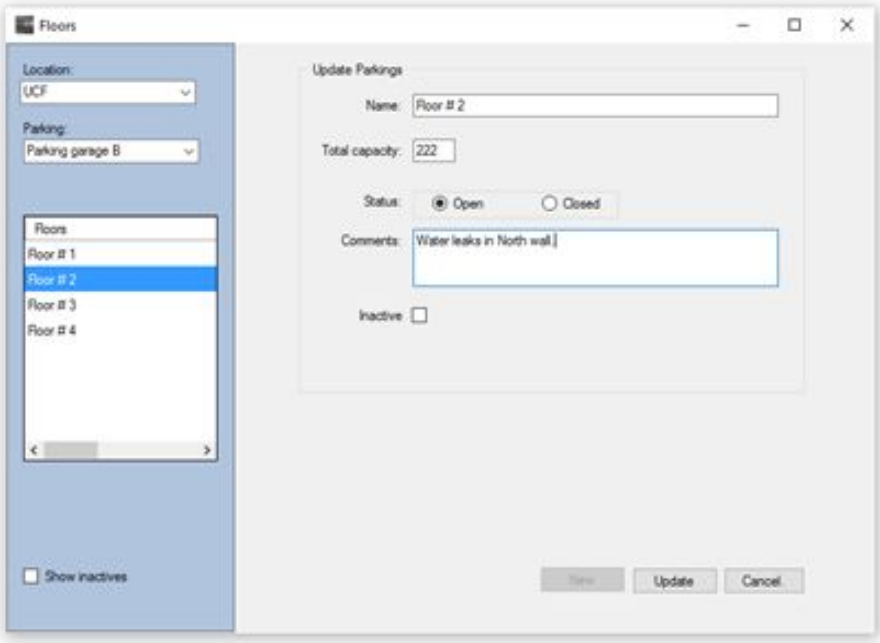


Figure 8.8(3) - Update Floors Screen.

4. Create Microcontroller(s) within a floor.
    - In this step the operator must take note of the Id created by the server application, for this must be hardcoded into the microcontroller using the previously provided sketch (C program), and locate it the designed parking/floor.

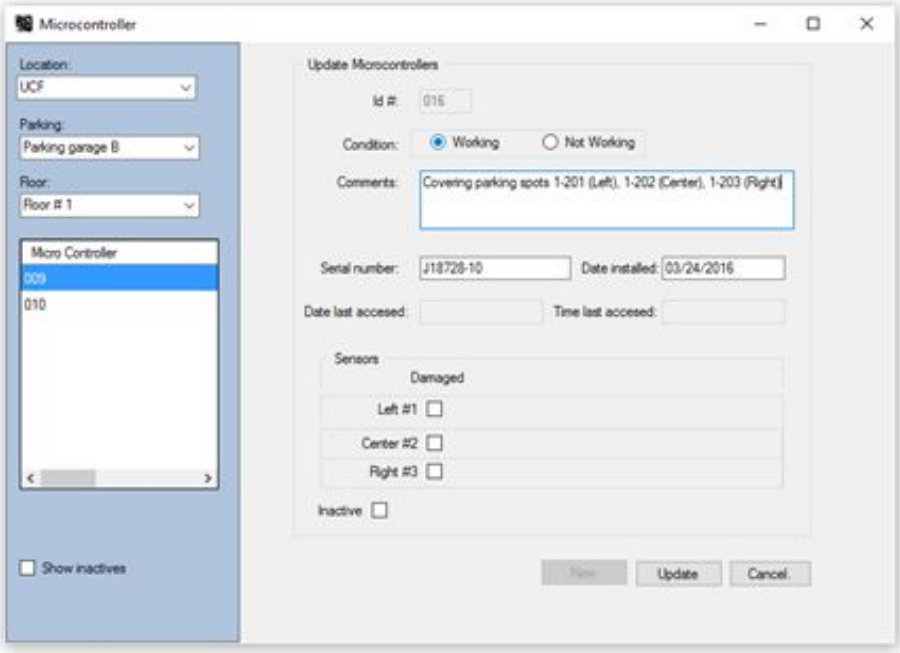


Figure 8.8(4) - Update Microcontrollers Screens.

Monitoring the parking is performed by the monitor user. This task besides informing of parking availability, also informs about problems in sensor, and microcontroller. It also informs of closure of floors, or parking buildings. From the main menu, the operator select *Monitor* from the *File* from the menu (shown below in **Figure 8.8(5)**).

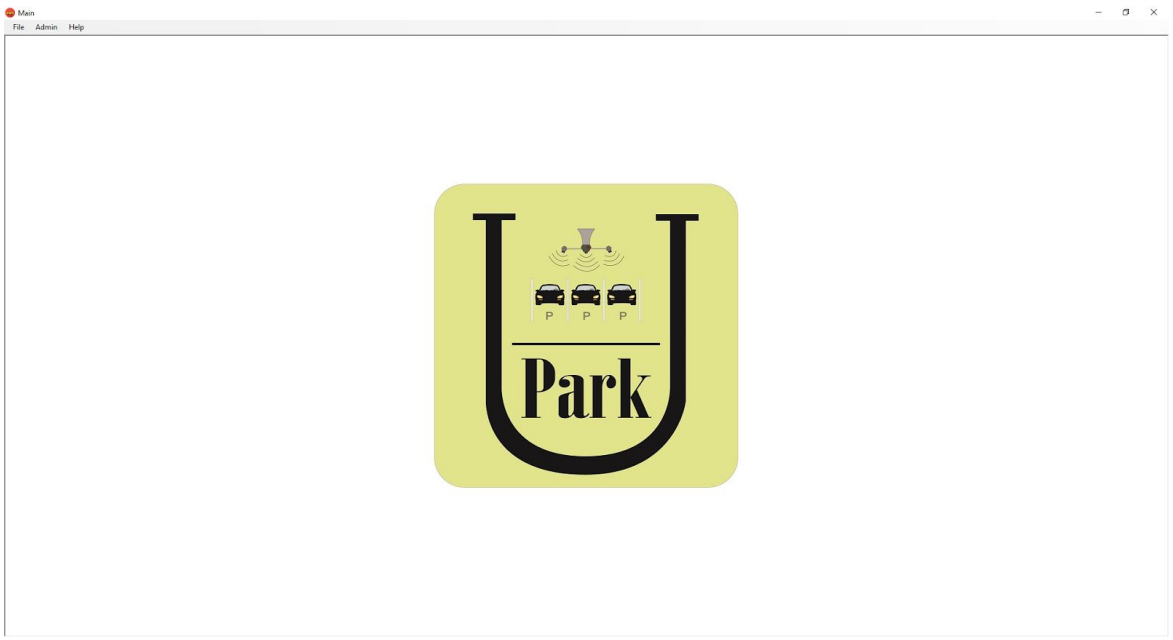


Figure 8.8(5) - Main Screen.

Upon selection, the following screen is displayed:

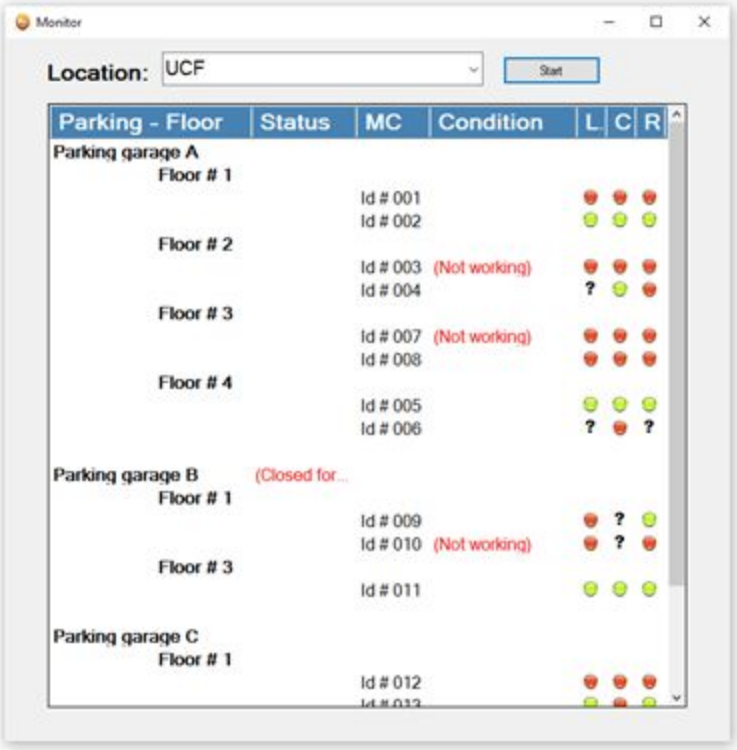


Figure 8.8(6) - Check Status Screen.

This screen presents all parking spots sorted by ‘Parking garage’, then floor, and finally microcontroller. To read this screen is necessary to know the following indications:

- A red flag indicates the parking spot is occupied.
- A green flag indicates the parking spot is empty (available).
- A question mark (?) indicates that the last ‘read’ operation was not successful.
- Additionally, information about the status of parking, floors and microcontroller is also presented. Columns are resizable and movable in order to accommodate all information.
- This screen is updated every 30 seconds.

Communication module: This program runs on the server as a windows service (background process). Its role consists in reading the data produced by the microcontrollers, and update the database with such information. To perform these tasks, the program must contain the following functions:

1. Start: Initiate a Telnet session (as a client).
2. Read: Read each microcontroller's data, and verify its validity.
3. Write: Insert/Update the data just read into the database.
4. Check: for missing microcontrollers, and if found report them. Also check for connections with router, microcontrollers, and database.
5. Sleep: For thirty seconds and loop to number two.

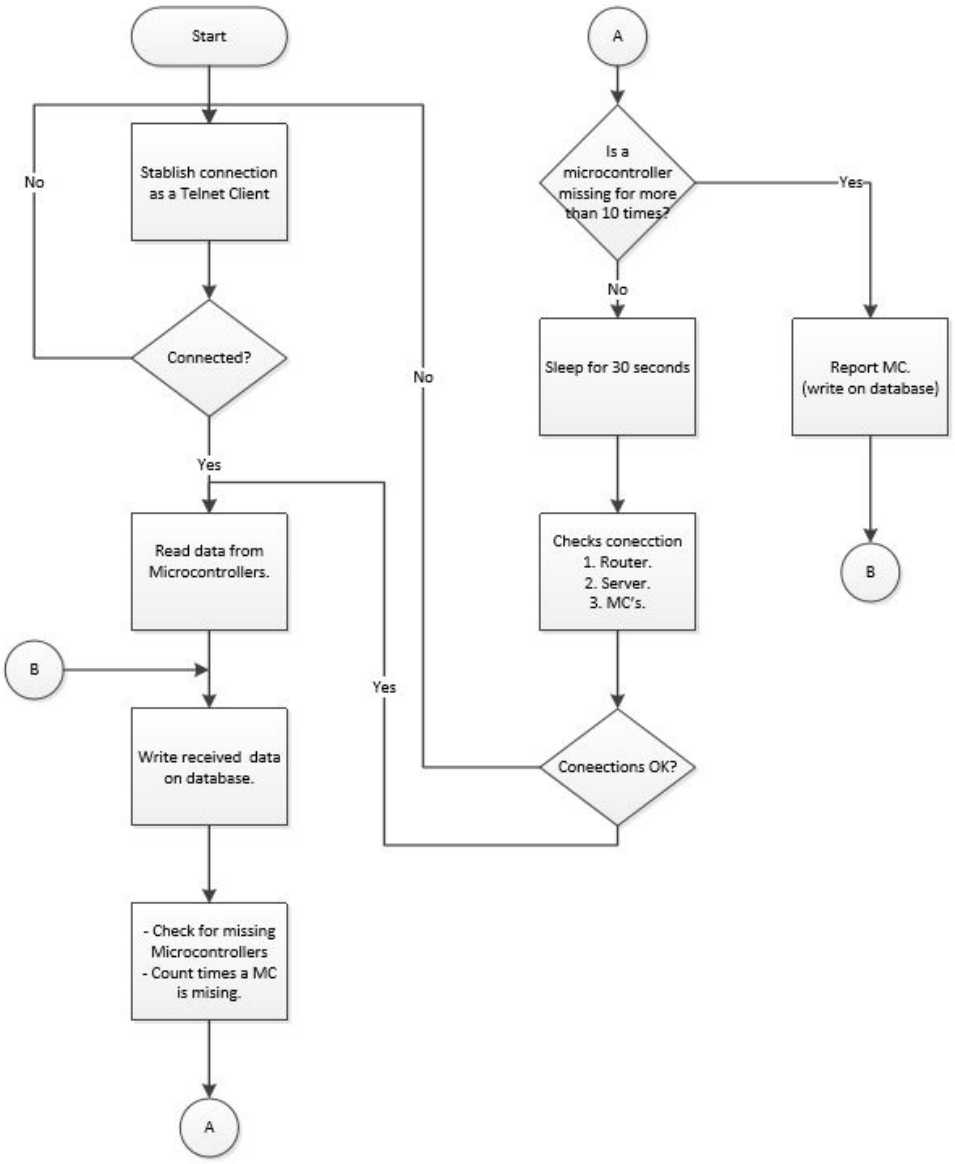


Figure 8.9 - Arduino Sketch Flowchart

8. 5 Construction

In this section the specifics of the construction of the U-Park prototype as a whole will be discussed. This section will cover the facilities which will be used by the team to develop and build the prototype, and the equipment provided by these facilities which the team has made use of. Various machines and materials are provided for use at UCF, paid for under tuition. The team will require some other equipment to build and test the prototype, which will also be discussed.

In the second part of this section, the PCB supplier OSH Park will be explored and the process through which the team will obtain the PCB board from OSH Park will be covered. The parts suppliers and the purchasing will also be discussed, and the reasoning behind why certain suppliers were chosen will be listed.

8. 5. 1 Facilities and Equipment

The U-Park system contains a mix of software, mechanical and electrical hardware components. A variety of facilities and equipment is required to build the final prototype. This sub-section will provide an overview of the facilities and equipment used in the development and production to construct the working final prototype used in the Senior Design two faculty demonstration.

During the initial development, the team member require fewer resources than in the later portions in the design process. Much of the smaller equipment used to prototype such as power tools, soldering equipment, and hardware are able to be supplied by the U-Park team members. Roddey Smith and Carlos Pereda already own much of the hardware required to produce a working prototype. The database and web interface require a small test server and a computer to run and develop these. Luckily, as students who have had to accomplish similar tasks in previous classes, the team members do not have to rely on facilities or hardware provided by UCF to develop these features.

One main exception to this will be when the team needs to design and construct the 3D model of the casing for the sensor modules. For this, the team will need to make use of the CAD software, and the 3D printers provided in the various labs at UCF. Another option available to the U-Park design team to design and build the final sensor module housing is to make use of the facilities at the Space Coast FabLab. This is a privately owned and operated facility, originally developed by MIT, which has opened up locations around the country. The Space Coast FabLab facility offers free 3D printing for students working on school projects, and would therefore be a good option if the labs at UCF become too crowded with other Senior Design groups. The Space Coast FabLab facility is located in Palm Bay Florida, and also offers many other tools which may be required by the U-Park design team when constructing the final prototype.

8. 5. 2 Suppliers

This section contains information about the various suppliers where many of the components used by the U-Park design team during the developments process were purchased. The primary suppliers or the components include UCF, Amazon.com, and OSH Park. In Table 8.1 below, the specific suppliers are shown, along with the components obtained from those suppliers.

Table 8.1 - List of Suppliers for the U-Park System Components:

Supplier:	Components:
UCF	● Resistors ● Capacitors ● LEDs ● 3D Printing ● CAD Software in Labs at UCF
Amazon.com	● ATMega 328p-pu Microcontroller ● HC-SR04 Ultrasonic Sensors ● 120V to 12V Transformer ● Switching Regulator ● 16 MHz Crystal Oscillator ● Fuses ● 1N4007 Diodes ● Wire (misc.)
OSH Park	● Custom PCB Fabrication
Home Depot	● Mounting Hardware ● Angle Aluminum for the Sensor Extension Arm
U-Park Design Team	● Website and Database Development Software ● Power Tools Used During Construction of Final Prototype

While Other suppliers may be required as time progresses, this table gives a preliminary plan for purchasing, and lists the sources where many of the initial prototyping components have been sourced from. Amazon has proved to be an

invaluable resource for components as many of the components can be purchased and arrive via either 1-day or 2-day mail.

9. Testing Plan

"*Testing the system is very different from unit and integration testing. When you unit test your components, you have complete control over the testing process. You create your own test data, design your own test cases, and run the test yourself. When you integrate components, you sometimes work by yourself, but often collaborate with some of the test or development team. However, when you test a system, you work with the entire development team, coordinating what you do and being directed by the test team leader*." (3)

9. 1 Sensor

The first test needed to be performed by team nine, is the depth sensor reading test. This test is done on each ultrasonic sensor to ensure the distance reading is accurate. Sending consistent ultrasonic pulses out from an individual sensor, use a tape measure to read the current distance and look at the values received from the ultrasonic sensors. If they are within +/- an inch of each other, the sensor should be functioning properly. After that, place different objects (blockages, cars, soft objects, etc.) in front of the sensor to observe how the reading changes.

The second test is the connection between sensors and microcontroller. Since there will be at least two sensors connected to one microcontroller, make sure that all of the sensors communicate with the microcontroller.

One microcontroller will be installed per two or three parking spots, so two of the sensors will be at an angle. This angle must not be so steep the ultrasonic signal has trouble reflecting back. The angle of the Ultrasonic sensor should be positioned at around 15 degrees from the vertical axis to meet this requirement. .
The Ultrasonic sensor will not be tested in a full-scale implementation in a real parking garage due to the garage being owned by UCF. Instead, Team nine will build a scaled-down parking garage and install microcontrollers and sensors in that parking garage model as a proof of concept. While there are some obvious aspects that will be different from a "real" parking garage, such as temperature, target, etc., the basic idea functionality is exactly the same.
Another test of the ultrasonic sensor, is to see how long it takes for the sound wave reflect back to the receiver (ping). The sensor needs to provide the information to the microcontroller as fast as possible so that the information can be uploaded to the database and updated on the application interface.

A full system test will need to be evaluated as soon as possible to ensure that all sensors are stability tested with the full readout chain. This will help to ensure that unforeseen electronics-related issues, including noise, are addressed as early as possible.

9.2 Microcontroller

To ensure that the whole system is functioning properly, the most important test of the hardware side is microcontroller testing. The microcontroller is the core of the PCB board, so if something is wrong with the microcontroller then the entire system will not work. Since to test the microcontroller unit, the sensors need to be connected in order to have visual input and output, this unit testing has to be done after sensor testing.

First of all, make sure all the components that are connected to the microcontroller are in the correct pin base on the ATmega328 pin mapping. The values of these components have to be exactly the same as the design. Especially, the two capacitors that are connected to the crystal oscillator.

Once the connection is done, power the microcontroller. Use the multimeter to measure the voltage at all points on the board. Make sure that the measured voltages are correct. After the code is uploaded to the microcontroller, connect sensor to the microcontroller. If there is a target in front of the sensor, the computer screen will appear the result. If the reading result is correct, then the microcontroller is functioning correctly. If there is no response, then there is something wrong with the microcontroller since the sensors are already been tested. If so, the connection of the microcontroller need to be checked again.

9. 3 Network

The U-Park Network is constructed on a system of WIFI transceivers which all communicate with a central router to the internet. The network management system will be similar to that used in auditorium-sized classrooms. Because there is a bandwidth limit on routers, and because routers (aka “access points”) can only support a certain number of concurrent devices, the actual real-life implementation of a full-scale system in the prototype for this stage of the senior design process is impossible without further funding. The professional-grade access points which would be required to support the total number of sensor nodes is well into the thousands of dollars for the access point alone. Most consumer grade router hardware can only support up to about 255 concurrent

users, which in reality is not possible due to interference issues between network channels. This would limit the size of the garage to a maximum of 765 spaces (three sensors per connected modules) connected to a single router. This would also be a huge problem since the system would be operating at the quoted limits of the router and is not desirable for liability purposes. For this reason, industrial-grade network hardware infrastructure would be required and would need to be developed in such a way that would allow for the system to grow.

In an entire parking garage, physically reaching all of the individual sensor nodes, where the number of total parking spaces can be between 1000- 2000 proves to be a challenge. To ensure that the system is robust and reliable enough to be expandable to other parking applications in other locations, the routers used in the final project would be responsible for a certain number of sensor modules in a certain physical area. Multiple routers would need to be placed throughout the garage in strategic locations. The multiple routers would need to be connected in a mesh network, and would communicate with each other and work together to be able to support a growing number of sensor nodes in larger and larger parking garages.

The U-Park system final project deliverable shown at the final demonstration will contain only one router which will connect to a total of less than three users (sensor modules). The displayed system will be developed as a system of building blocks, where once a router is connected to a set of sensor modules, that router can be connected to the U-Park mesh network inside of the garage to monitor different physical areas in the garage. This allows for the use of standard network control protocols, and would not require the development of new and untested systems, which would be a risk to the reliability specifications of the U-Park system.

In summary, the U-Park network will consist of the individual sensor modules connected in groups of 10 or 15 by physical locations in the garage connected through 2.4GHz Wi-Fi to a router/ access point. The total number of access points required is dependent on the size of the garage in question, but would communicate between each other in the same way in the mesh network of the garage regardless of the size of the garage or the number of nodes required by even a large garage. This network of routers accesses the internet through a gateway and communicates back and forth between the database to populate the data on the web interface for the parking availability in the garages.

9. 4 Database

For testing the database and the server application, team nine applies the classical *System Testing Process*, the tests are:

- Function testing -Each function in each program of the server application is checked for correctness. Example a routine that insert row into a table is checked that it performs that action with accuracy, and does nothing else. In practice, table update programs consists in three basic functions:
    - Insert rows.
    - Update rows.
- For the database portion, it's checked that:
    - All tables with foreign keys relate in the proper way to their parent table.
    - Key increments (both self-generated by MySQL and programmed are accurate.
- Web Server and Mobile App:
    - Basically put the applications to run and compare the data shown with the data stored at the database.
    - Performance testing. See *Proposed tests*
    - Acceptance testing.
        - Once performance tests are completed, the system will be given to the final user (simulated), for tests and acceptance.
    - Installation testing.
    - After final installation of the system, another set of performance tests will be conducted.

Test teams: In order to accomplish all tests required for this project, team nine has divided the work into four areas (listed below):

1. Function testing:
   Performed by the developers themselves.
   Note: Since this project involves developments both in hardware (microcontrollers), and software (Microcontrollers sketches, communication software, server application, web site, and mobile applications), the rest of tests will be conducted with both hardware and software operational.
2. Performance Testing: the software team swaps with the hardware teams for conducting tests. That's is, software developers will test hardware (AC/DC converters, Microcontrollers, Wi-Fi portion, etc.), while hardware developers will test the software applications.
3. Acceptance testing: (Simulated)
   The whole team will act as a customer receiving an unknown solution and will test it for functionality.
4. Installation testing:
   The whole team will test again prior to presentation.

Proposed tests:

- Functional Tests: Functional tests have to be conducted on the following software components:

    - Microcontrollers Sketch: Read sensors data using another microcontroller and sketch to compare them with the ones proposed in the project.
    - Communication software (Liaison) compare data read with data sent by microcontroller's data sent. Check for connectivity (refer to flowchart 8.4(9).
    - Server Application (SPOT): Verify functions:
        - Insert_Row: Make sure all data selected, "points" to the right columns.
        - Update_Row: Make sure all data selected, "points" to the right columns.
        - Verify_Data: Validate all 'validated data' from this function is accurate; Dates are in the right format, values are not out of range, etc.

- Performance Tests: This tests (under development) consists in putting the application under pressure. To achieve this goal, the following tests will be conducted.
    - Put the system to work, that is start capturing data (24/7 every 30 seconds), from every microcontroller, and store data into database. From this test, an evaluation of response time will be performed.
    - Querying data (Users looking for a parking spot). Volunteers will be required, for the test would be conducted with 20, 50, and 100 users simultaneously. From this test, an evaluation of response time will be performed.

- Acceptance Tests: The (simulated end user) will operate the application (subject of this project) as if it was a daily routine.
- Installation Tests: Finally the project will be put to work before the presentation datta.

9. 5 Application

9. 5. 1 Web

1. Test Create User

    a. Create user
    b. Try all cases outside of the scope of create user
    c. Make sure all cases are stopped
2. Test Log In
    a. Create username and login
    b. Login and logout multiple times
    c. Make sure system keeps track of user info
    d. Continue to monitor over time
3. Test Main Page
    a. View garage information on main page
    b. Update data in the database
    c. Make sure the changes are shown on main page
4. Test Admin
    a. Create Admin Account
    b. Use each feature of admin account
    c. Watch database to ensure correct changes are made
    d. Log in as user to check that changes are being shown on main page
5. Test Add Garage
    a. Login as user
    b. Add and delete garages multiple times
    c. Ensure that main page updates correctly.

9. 5. 2 Mobile

1. Test Create User
    a. Create user
    b. Try all cases outside of the scope of create user
    c. Make sure all cases are stopped
2. Test Log In
    a. Create username and login
    b. Login and logout multiple times
    c. Make sure system keeps track of user info
    d. Continue to monitor over time
3. Test Main Page
    a. View garage information on main page
    b. Update data in the database
    c. Make sure the changes are shown on main page

4. Test Admin
    a. Create Admin Account
    b. Use each feature of admin account
    c. Watch database to ensure correct changes are made

    d. Log in as user to check that changes are being shown on main page
5. Test Add Garage
    a. Login as user
    b. Add and delete garages multiple times
    c. Ensure that main page updates correctly

10. Project Operation

*Note: (This part refers to the daily operation, and it's not intended to be an installation manual)

On the microcontroller:

- Once a microcontroller is installed and powered, it starts a sketch (AVR-C program) written in EEPROM.
- Logic of the sketch (as described in the flowchart shown in figure 8.4.9) starts by declaring itself as a Telnet server, next the microcontroller on a 30 seconds (default) loop queries the three ultrasonic sensors attached to it and creates a record (as shown if figure 7.4.1) with the microcontroller ID, and the status for each sensor. In turn this information is propagated (sent via routers to the server). The microcontroller sleeps for another 30 seconds.

On the server:

- A Windows service "Liaison" is permanently waiting for microcontrollers to send data.
- After a record is received from a microcontroller, the information on the DBMS database (spot) is updated, and a history transaction is created if the table "transactions".
- At this point data is available for both "clients" (users looking for an empty parking spot) and "administrators" (operators checking for space availability and microcontroller malfunction).

On the "Client"-Side:

- Anyone looking to park at the location (In this case UCF), accesses the website designed for this application.

- Once in the website, the application determines geographically the location of the “Client”, and will inform of the nearest parking garage, and its availability giving in garage’ floors.
- Once the target gets to the parking spot, and his/her vehicle occupies an empty spot, the microcontroller will update the data.

11. Conclusion

11. 1 Reflections

11. 1. 1 Features Left Out

The U-Park system is the type of systems which performs one basic functionality, which means there are not too many extraneous features which are not included. The system requires a basic set of functionality to work, and adding too many extra features could actually get in the way of users using the parking management interface efficiently. However, there is one aspect of the system which is currently being overlooked due to time constraints. Currently the design team has decided to leave out the feature to be able to see how much traffic is driving through a particular garage. This information would help users not only know how many spots are available, but also if those spots are likely to be available in the near future, or if there already are a significant number of drivers in the garage that by the time the user would arrive the spots would be gone.

The U-Park system aims to be a product similar in usefulness to a fork. For instance a fork does a very helpful daily task, and no-one would want to live without a fork. But, adding too much functionality to the fork could detract from the actual usefulness and optimization for the specific task of eating. In the same way, if the design team tries to add too much functionality to the U-Park system, the added features could actually detract from the system’s primary functionality of helping users park faster and avoid driving around aimlessly.

11. 1. 2 Future Improvements

There are two main improvements that the U-Park system could benefit from. The first, already mentioned in the above paragraph is the addition of sensors to detect users driving through the garage. This would likely require sensors at all of the entrances and exits of the garage that would be able to keep a rough estimate of how many cars are moving in and out of the garage. This would be extremely inaccurate, however, since the sensors would not be able to tell the difference between a person and a car and would therefore falsely estimate the number of cars driving through the garage. The fix to this problem would be to have cameras set up that using image processing software, would be able to detect the difference between cars, motorcycles and humans, which would allow it to keep a better count on the number of vehicles inside the garage.

The other improvement which would be of use to the U-Park system would be the inclusion of user studies. These user studies would examine the users opinions about the web application interface, and would ask users for input on what features, if any, to add the the U-Park system which would help it better accomplish its goal of making the search for parking quicker and less cumbersome.

The primary reason these improvements are left to the future is due to the strict timeline required in the senior design process. If these improvements were tried to be accomplished in the time for senior design two, less time would be able to be spent perfecting the primary features of the U-Park system. Also, as the project develops in maturity, other important missing features may arise that could take precedent of the aforementioned ones.

12. Appendices

A. Works Cited

1. *What is Java Technology and why do I need it?.* java.com. Oracle Corporation. n.d. 16 March 2016. https://java.com/en/download/faq/whatis_java.xml
2. *My SQL Community Edition*. Mysql.com. Oracle Corporation. N.d. 25 Match 2016. https://www.mysql.com/products/community/
3. Pfleeger, Shari, Atlee Joanne. (2013). *Software engineering, Theory and Practice.* New Delhi: Pearson.
4. "Sharp Infrared Ranger Comparison." Acroname. Accessed April 26, 2016. https://acroname.com/articles/sharp-infrared-ranger-comparison.
5. "BU-106: Advantages of Primary Batteries." Primary (non-rechargeable) Batteries – Battery University. Accessed April 26, 2016. http://batteryuniversity.com/learn/article/primary_batteries.
6. "ATMEL 8-Bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash." Atmel. Accessed April 26, 2016. http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf.
7. "Full Wave Rectifier and Bridge Rectifier Theory." Basic Electronics Tutorials Full Wave Rectifier Comments. 2013. Accessed April 26, 2016. http://www.electronics-tutorials.ws/diode/diode_6.html.
8. "HC-SR04 User Guide." ElecFreaks. Accessed April 26, 2016. http://www.elecfreaks.com/store/download/product/Sensor/HC-SR04/HC-SR04_Ultrasonic_Module_User_Guide.pdf.
9. "BlueTooth HC05 Modules - How To"Arduino-info. Accessed March 1, 2016. https://arduino-info.wikispaces.com/page/history/BlueTooth-HC05-HC06-Modules-How-To
10. "Prescaler utility library" Arduino Playground. Accessed March 1, 2016. http://playground.arduino.cc/Code/Prescaler

A.

B. Permissions

BatteryU <BatteryU@cadex.com> Mar 29 at 4:58 PM
To themle21@yahoo.com

Hi Them,

Yes, you may use the material as requested. Please cites sources where appropriate.

Regards,

John Bradshaw - Marketing Communications Manager
Cadex Electronics Inc. | www.cadex.com
Vancouver | Minneapolis | Frankfurt
Tel: +1 604 231-7777 x319 | Toll Free: 1-800 565-5228

Permission email #: Battery University

B.