University of Central Florida
Department of Electrical Engineering and Computer Science

Senior Design II

Final Paper

LookSee

Group 6

Group Member	*Major*
Stavros Avdella	Computer Engineering (BSCpE)
Austin Pena	Computer Engineering (BSCpE)
Tyler Wallace	Computer Engineering (BSCpE)
Jarvis Zsiros	Computer Engineering (BSCpE)

UCF
UNIVERSITY OF CENTRAL FLORIDA

Contents

1 Executive Summary

In robotics, it is common to say that a robot can or should replace a human worker in situations represented by one of the three “D’s”: Dull, Dirty, or Dangerous. Robot-composed symphonies and robot-designed paintings are rarely able to rival a shadow of the heart a human artist is capable of evoking, but for toxic, boring, and high-risk jobs, getting the fragile people, prone to errors after long shifts and irreplaceable if damaged, out of harm’s way is the priority of the field of robotics.

As robotics science progresses, more and more of the population will have access to robotic solutions, and more people will want to own a piece of this emerging technology. As high-cost, high-performance robotic technologies for surgical and military use progress, it is the belief of the members of this project that common home and business owners should be able to benefit directly from advances in this field as well.

LookSee is a low-cost, consumer-level surveillance robot intended to patrol an indoor area for the purposes of a small business. Many small grocers and offices would like the security of knowing that their building is safe during hours when regular employees are not there, but cannot afford to hire a night guard or install an expensive camera system. LookSee will be able to be purchased, removed from the box, and used with minimal setup time for less than 1000 dollars.

2 Project Description

LookSee is a low-cost surveillance robot that will autonomously patrol indoor facilities when no employees are present and detect human intruders. If an intruder is detected, LookSee will begin a video call with an emergency contact, who will attempt to communicate with the intruder. LookSee will also record video of the detected intruder for later review. A desktop interface will be available for use by the emergency contact.

2.1 Motivation

This project had several motivations that contributed to the final project concept. The first motivation was to design a robot. The majority of the project members have an interest in robotics as a future career and are looking for a project to talk about in job interviews. The second motivation was to design a project whose work would be extremely modular. As the current year’s senior design program is online, and the team members have extremely limited ability to meet and work on the project together in person, the most important requirement of the project design from a logistical perspective was that the vast majority of the work could be completed independently, with the expectation that the work on individual robot components could not be integrated for weeks or even months at a time. The final major motivation for the project was to find a use case we could all be excited about. As this is a project we would be working on extensively for two semesters, it was of the utmost importance for team morale that the robot we designed contained elements that all four members would want to see to completion.

Once we had determined those constraints, we settled on the idea for a mobile autonomous robot. From there, we narrowed down our concept further by discussing individual elements we wanted to incorporate for various logistical or personal reasons. We wanted to use computer vision, so we knew that the robot would need to search for a visual cue while driving. We wanted to use sensor fusion, so we decided to combine the thermal sensor with the camera to search for and verify the existence of a human. We wanted to display competency with LiDAR navigation, so we decided that our robot would patrol an area. We wanted an element of human-robot interaction, and so we settled on the emergency contact interface that would allow a remote user to take over control of the robot and converse verbally with an intruder. By the time we had highlighted all of the areas that our team members wanted to focus on, we had an idea that the elements outlined could come together to make a strong surveillance robot.

2.2 Objectives and Goals

The goal of this project is to create a robot that can autonomously patrol an indoor area when no humans are expected to be present and respond if a human intruder is detected. The robot will be able to navigate a space, identify the presence of a human, and contact a designated emergency contact if an intruder is detected. A major design focus is ease of use. The robot should require extremely minimal setup time on the part of the user, and in the case of an emergency, the user should be able to access and use the user interface with greatest possible ease.

This robot has several objectives. The first objective is to patrol an indoor area and to be able to sense in all, if not nearly all, of the desired area to be patrolled. The second objective is to detect the presence of a person in the patrolled area. If a person is detected, the robot must begin recording video and attempt to contact a designated emergency user, who will be using a desktop computer.

The emergency user, if successfully contacted, must be able to see streamed video of the intruder and control the driving of the robot. The emergency user must also be able to converse with the intruder.

2.3 Core, Stretch, and Advanced Functionality

We have come up with a design for core functionality that we believe we can achieve with leeway in the next semester. However, if we finish early, we do want to be able to add functionality to improve our product.

Core functions that the robot must have by the end of Senior Design 2:

- The robot must be able to navigate its environment without live control from a human operator.

- The robot will drive behaviorally using "find-the-gap" navigation or another algorithm of comparable complexity.
- The robot must be able to detect when a person walking normally passes in front of it in clear view.
- If an intruder is detected, it must attempt to contact the human operator.
- The human operator will have a desktop or browser application that allows it to see the robot's camera footage in the case of an intruder.
- The robot will have functionality that will allow the human operator to speak through the robot's speakers and listen through the robot's microphone.
- The battery should be able to last at least six hours while idle.

Advanced functions that the robot may have if we wish to add additional functionality:

- If an intruder is detected, the robot will begin recording video footage for later retrieval
- The human operator will have the option to record a path for the robot to traverse
- The sound of the robot's regular driving should be reasonably quiet.
- The battery should be able to power the robot for nine hours while the robot is idle.
- The robot should be able to start its routine automatically on a schedule.
- The robot should contain a gyroscopic sensor and alert the human operator if any unwanted motion is being inflicted on the robot, such as if a person were moving it by hand.
- The robot should listen for loud noises and send a transcript of any spoken words to the emergency contact over text message.

Stretch functionality that we may add to the robot if we are finished with all core functionality very early:

- If an intruder is detected, the robot will begin recording video footage and back the footage up to an external location over wireless connection.
- If the robot loses connection with the human operator's station for more than one minute, an error is logged and the robot attempts to contact the human

operator to inform them of possible tampering.

- The battery should be able to power the robot for twelve hours while the robot is idle.
- The robot will be able to navigate using SLAM.
- The robot's navigation will be assisted by sensor fusion of data from both the camera and the LiDAR unit.

2.4 Requirements and Specifications

This section will detail the base requirements and specifications for our project. These specifications will serve as a guide to make sure we are meeting the minimum requirements needed for the overall project to be completed successfully. The requirements and specifications are detailed in Table 1 and requirement falls into a category based on the requirement ID listed.

- NV - Navigation
- PR - Processing
- SE - Sensors
- CHA - Chassis
- PWR - Power
- CAM - Camera
- LID - LIDAR
- CO - Control

2.4.1 Requirements List

ID	Requirement Description
NV1	Must be able to drive autonomously in a loop
NV2	Must be able to autonomously navigate a reasonably uncluttered room for at least 10 minutes
CAM1	The camera we select should be able to connect over USB
CAM2	The camera we select should produce at least 256-color images
CAM3	The camera we select should weigh 4 ounces or fewer
MIC1	The microphone we select should be able to pick up the sound of a person speaking at normal volume from 10 feet away or farther.
LID1	The LiDAR unit we select should have a footprint less than 3" by 3"
LID2	The LiDAR unit we select should return at least 360 points per scan.
LID3	The LiDAR unit we select should be able to scan at least 200 degrees
LID4	The LiDAR unit we select should be able to scan at at least 10 Hz
CHA1	The chassis we select should be prebuilt with wheels, motors, a battery, and a motor controller.
CHA2	The chassis we select should be a 1:16 scale RC car or larger.
CHA3	Any plastic decorations on the car should be easily removable so that the sensors and electronics can be easily added to the vehicle.
WH1	The wheels we select should be large, all-terrain rubber wheels. Nearly all models of RC car in the size we are looking for use turn steering, and as such, we cannot use mecanum or omni wheel drive systems.
ST1	The chassis we select should use turn steering.
CO1	The chassis we select should have a motor controller that can be accessed by the Teensy 3.2 with minimal modification.
SE1	Use a LiDAR for navigation
SE2	Use a camera for intruder detection
SE3	Use a thermal sensor to facilitate intruder detection
SE4	Have a microphone and speaker to conduct voice calls
SE5	The robot may use a gyroscope for error detection
PR1	The processor must fit within the form factor of the chassis
PR2	The processor must be able to install and run Ubuntu 16.04
PR3	The processor must be able to support connections to all inputs and outputs
PR4	The processor must be able to run ROS and all team code written for ROS
PR5	The robot must be able to detect an unmasked, adult person introduced to a reasonably uncluttered building located 10-20 feet in front of the robot in at least 70 percent of tests
PWR1	The robot must be able to be powered on for at least 4 hours
PWR2	The robot's batteries must be able to be recharged
PWR3	The robot must use Lithium Ion batteries
PWR4	The batteries should be easily accessible and removable

Table 1: Robot Requirements

2.5 House of Quality

This section shows the House of Quality (HOC) which simply charts the correlations between customer requirements and engineering requirements. This diagram shown in Figure 2.5.1 details a list of customer requirements on the left, their perceived importance on a scale from one to five, and the percentage of perceived importance. On the top are a list of some basic engineering requirements. The intersections of each of the customer requirements and engineering requirements are then given a relationship symbol to show how strongly each affects the other. A key for what each symbol means is given to the right of the HOQ. Just below the decision matrix the importance of each engineering requirement is calculated based on the relationship matrix. These numbers will give us important insight as to which engineering requirements we should focus on. Just below the importance ratings are the target for each engineering requirement. Just above the engineering requirements is a row of arrows pointing either up or down. This row indicates whether each engineering requirement is better higher (up arrow) or lower (down arrow). Finally just above that is the correlation pyramid showing how each engineering requirement correlates with one another. Again the key for the correlation symbols is to the right of the diagram.

The House of Quality lays the foundation for what needs to be focused on based on the customers needs while defining clear guidelines. This chart assists engineers by directing their focus on what needs to be focused on and why by relating it to the customers needs. The customer requirements we selected are:

- Size: This was selected because we don't want our solution to be big and intrusive when not in use. It should have a size that allows businesses of various floor plans and layouts to adopt a system such as ours.

- Lightweight: The customers would want this device to be lightweight in the event that they would need to transport it to another location or a different floor. It should be able to be carried by a normal person without much effort.

- Easy to Use: This encompasses a few things, first the user interface should be easy to use. When the user needs to take over manual control it should be second nature without any real learning curve. Also the autonomous mode should just work with barely any setup from the end user.

- Reliable: The solution should be reliable both in construction and operation. The chassis and overall design should be sturdy and not break down over time or after considerable use. The device should also be reliable in detection, operation, and performance.

- Cheap: Price points are always a factor for the end user. Specifically this device is looking to be either a placeholder for an area who does not already have a security system in place or cannot afford one. Therefore the price point would be a deciding factor for many end users.

- Accurate Detection: Users want the solution to be able to accurately detect a human presence as to not be bothered with false positive alerts. The solution must also be good at human detection when an actual human is present otherwise the solution is not really a solution.

- Long Lasting Battery: The battery life is important for many customers as it will need to be able patrol a given area while the business is closed either continuously or in intervals. The battery needs to be able to sustain the system long enough to complete its entire route or its entire shift.

- High Quality Camera: Whether the user is operating the device manually or it is autonomous mode, the image quality of what is captured is important. If the device decides to record an image and send it to an end user, the quality of the image must be good in case there is a false positive. The end users are the final check to make sure an intruder has been detected and therefore they need to be able to make determinations based on the images provided. Also the detection algorithm will be more accurate if the picture it is able to analyze is as clear as possible.

The engineering requirements are general and allow us to determine the importance of major aspects of our device. The engineering requirements we selected and how they relate to the customer requirements are:

- Weight: The weight of the device needs to be determined since there are many components that will compose the final product. If it is determined that the weight has a high customer importance then we need to be very conscious of the types and weights of the other components we select. 'Size' and 'lightweight' have obvious strong relationships. Since components cost and their size are closely related there is a weak relationship between the weight of the solution and the cheapness of it. This relationship is weak because most parts we will be selecting come standard in a rather small form factor. A long lasting battery means a larger battery which will also increase the overall weight of the solution.

- Cost of Production: The engineer's goal is always to keep the cost of production as low as possible. This is included so we can rank the importance of it to everything else. In some cases it may be acceptable to push the cost of production up if cost isn't important to the customers. The cost of production is directly related to the following customer requirements; size, reliable, cheap, and high quality camera. All of these have a strong relationship with the overall cost it will take to manufacture. A long lasting battery also matters for the overall cost, there are many different acceptable batteries and this should not change the cost as much as the others.

- Detection Algorithm: There are several methods to implement computer vision detection and classification. Each of these different methods can be compared and contrasted and a method can be selected based on the overall importance. This is important because customers want accurate detection and they want it to be reliable. Both of these are strongly related to the detection algorithm we select.

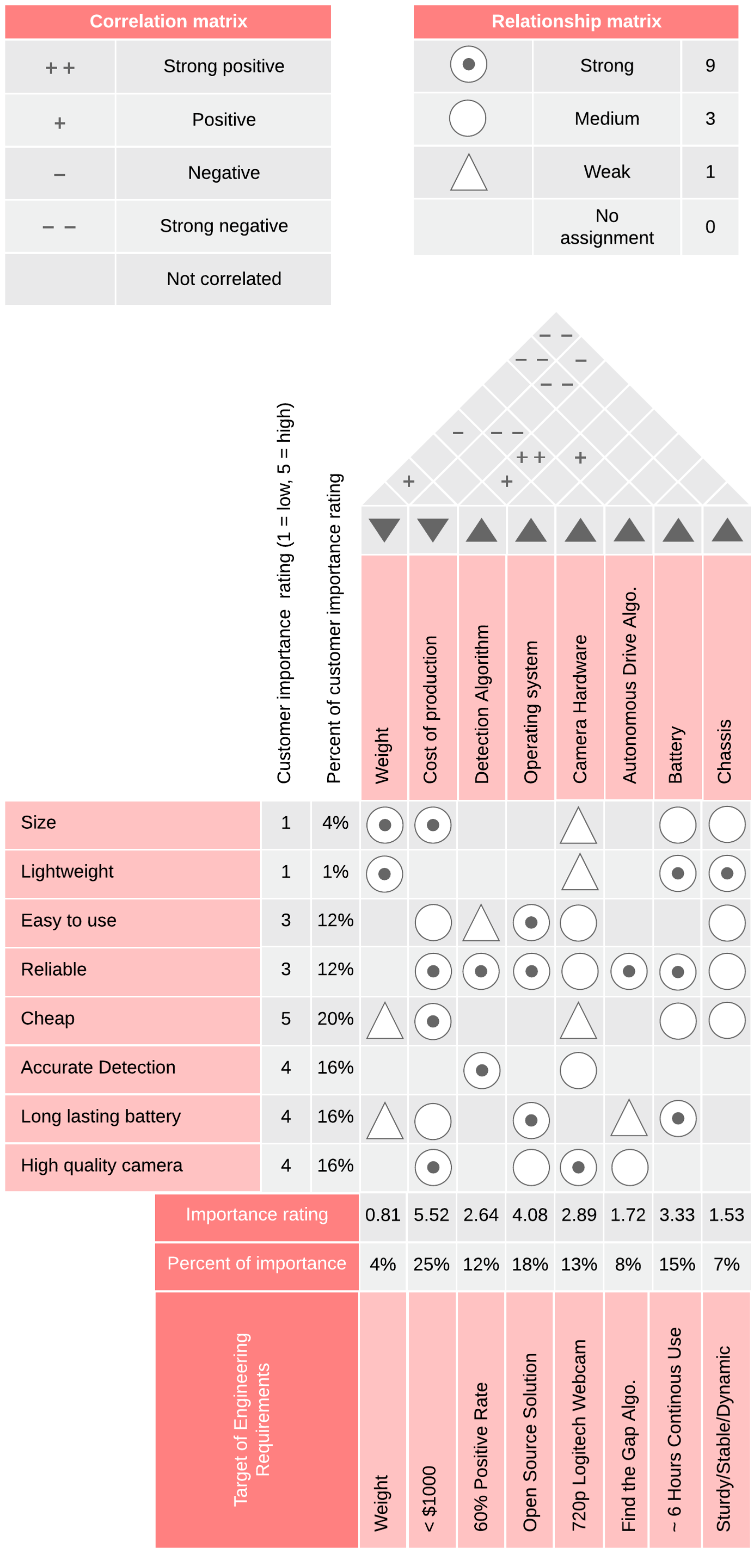
- Operating System: Depending on the hardware we will be using there are several options for which type of operating system we want running on the on board computer. The operating system strongly impacts the customer requirements of its ease of use, reliability, and long battery life, since the operating system manages a lot of these aspects. The operating system must also be able to manage taking in the image data and relaying it to the front end thus impacting the high image quality wanted from customers.

- Camera Hardware: The camera selection is an important aspect and must fit into the overall design as well as provide the customer with quality images. The camera selection has a relationship with every part of the customer requirements except for the battery life since almost all cameras will use a small amount of power.

- Autonomous Drive Algorithm: There are several methods to implement a self driving vehicle. Each of these different methods can be compared and contrasted and a method can be selected based on the overall importance. This greatly affects the reliability of the system and the battery life. A bad drive algorithm means we could waste time bumping into walls or getting stuck rather than patrolling.

- Battery: The battery selected will factor into the overall weight and reliability of the overall system which is why we chose a strong relationship for these. The battery will have a more moderate impact on the overall size and cost of the system.

- Chassis: The chassis will greatly impact the overall weight of the design since it will be the heaviest part of the entire build. Other customer requirements that will be affected by the chassis chosen will be the size, reliability, price point, and ease of use.



Correlation matrix	
+ +	Strong positive
+	Positive
–	Negative
– –	Strong negative
	Not correlated

Relationship matrix		
⊙	Strong	9
○	Medium	3
△	Weak	1
	No assignment	0

	Weight	Cost of production	Detection Algorithm	Operating system	Camera Hardware	Autonomous Drive Algo.	Battery	Chassis
Weight		+		–			– –	– –
Cost of production					– –		– –	–
Detection Algorithm				+	+ +			
Operating system						+		
Camera Hardware								
Autonomous Drive Algo.								
Battery								
Chassis								

	Customer importance rating (1 = low, 5 = high)	Percent of customer importance rating	Weight	Cost of production	Detection Algorithm	Operating system	Camera Hardware	Autonomous Drive Algo.	Battery	Chassis
			▼	▼	▲	▲	▲	▲	▲	▲
Size	1	4%	⊙	⊙			△		○	○
Lightweight	1	1%	⊙				△		⊙	⊙
Easy to use	3	12%		○	△	⊙	○			○
Reliable	3	12%		⊙	⊙	⊙	○	⊙	⊙	○
Cheap	5	20%	△	⊙			△		○	○
Accurate Detection	4	16%			⊙		○			
Long lasting battery	4	16%	△	○		⊙		△	⊙	
High quality camera	4	16%		⊙		○	⊙	○		
Importance rating			0.81	5.52	2.64	4.08	2.89	1.72	3.33	1.53
Percent of importance			4%	25%	12%	18%	13%	8%	15%	7%
Target of Engineering Requirements			Weight	< $1000	60% Positive Rate	Open Source Solution	720p Logitech Webcam	Find the Gap Algo.	~ 6 Hours Continous Use	Sturdy/Stable/Dynamic

Figure 2.5.1: House of Quality Diagram

3 Research

This project requires a vast amount of interdisciplinary knowledge. This section outlines different types of research that was compiled for this project. This section will serve as a way for all members of the project to get information regarding each aspect of the project and why certain decisions were made over others.

3.1 Similar Projects

Using the UCF ECE Senior Design website [1] we found past projects that are similar to ours, or share a similar concept that we will use as insight to help us with our own project. We also researched the internet for products that are already out on the market, as well as projects we individually have been apart of that may serve to be beneficial to our project.

3.1.1 A.S.R. The Autonomous Sentry Robot

A.S.R is a past Senior Design project from 2015 sponsored by Boeing, shown in Figure 3.1.1. The project was from the work of Brian Dodge, Nicholas Musco, and Trevor Roman [2]. Their project is the very similar to our own as their Robot is designed for security. Their robot consist of VEX Robotics drive train, sensors, a Microsoft Kinect, and a laptop. They use the Kinect and range senors to map the robots surrounding and be able to move around autonomously. It uses the map to to find the ideal path to patrol and use its sensors to detect any environment changes or motion. If the robot detects something, it will alert the user and provide them with a live video feed as well as the ability to take control of the robot.

Although their project was a success, the Microsoft Kinect is almost ten years old and we will not be considering this as a option for our robot's camera. There is much newer and better technology out there that we can use to improve on this concept.

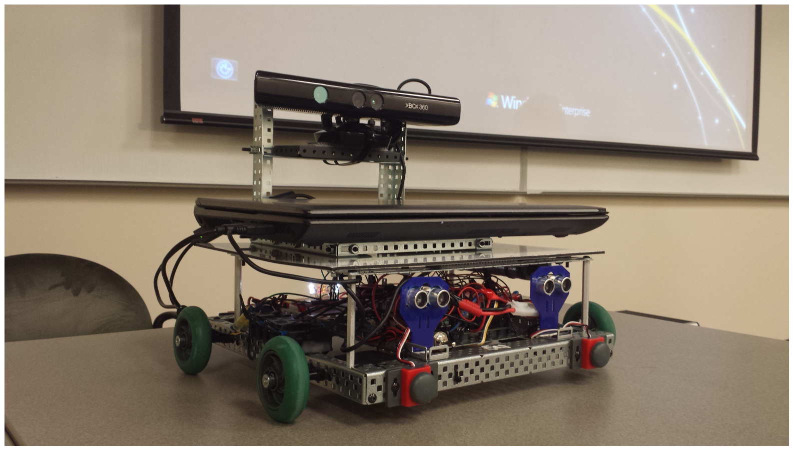


Figure 3.1.1: A.S.R
(Reprinted with permission from Trevor Roman)

3.1.2 T-100 Watch Dog

T-100 Watch Dog is a past Senior Design project from 2014 sponsored by Boeing and Texas Instruments, shown in Figure 3.1.2. The project was from the work of Ismael Rivera, Warayut Techarut, Chris Carmichael, and Journey Sumlar [3]. Their project objective was create an autonomous robot that alerts the home owner when an intruder is detected. They use a combination of object detection, tracking, and avoidance as well as autonomous control, remote control,and wireless communication to achieve this.

Their robot chassis consist of as VEX Robotics base with Mecanum wheels [4]. These wheels give them option to turn while stationary, which is a huge benefit for the purpose of surveillance and we will be considering this as an option for our own project. They use a webcam as their main source of detection using OpenCV algorithms for target detection. The camera they use a Logitech C300 Webcam and a Tamarisk 320 as the thermal camera. Depending on the environments lighting the camera will toggle from the Logitech for normal to bright lighting, to the Tamarisk for dim to dark lighting.

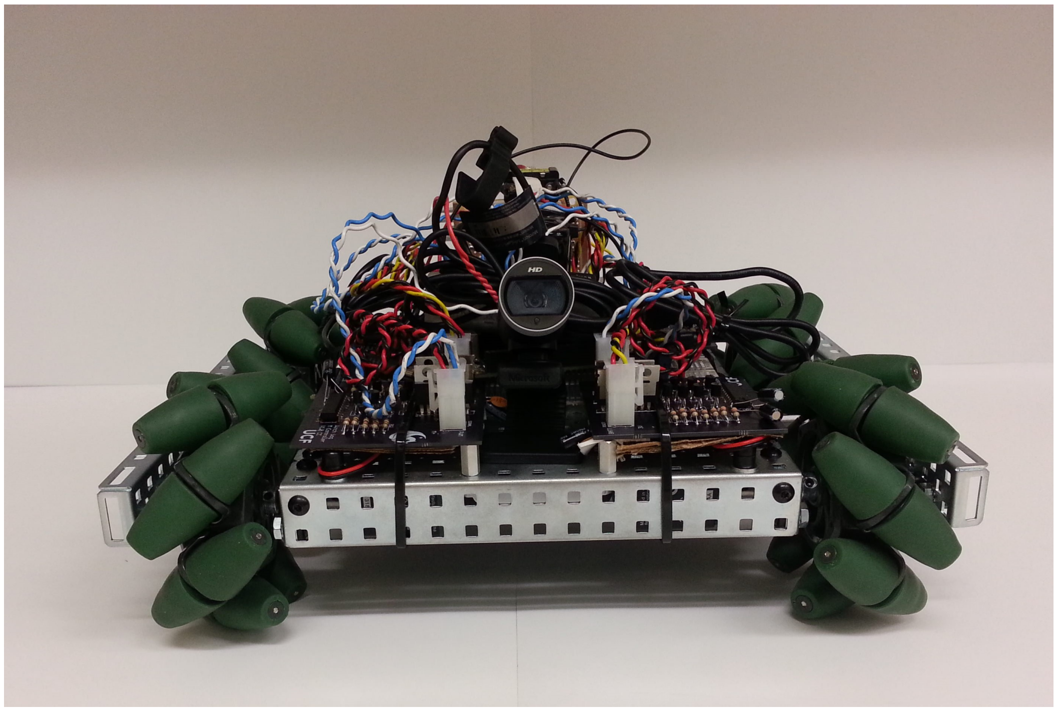


Figure 3.1.2: T-100 Watch Dog
(Reprinted with permission from Christopher Carmichael)

3.1.3 Knightscope

Knightscope is a commercial autonomous mobile robot intended to replace human guards in outdoor situations such as malls, warehouses, offices, hospitals, and even airports [5]. Knightscope uses multiple 360 degree cameras and microphones build all around it's body. The combined cost of renting the unit and its monthly online service makes using it at least as expensive as hiring a human guard. It's capabilities as listed on their website include [5]:

- Force Multiplying Physical Deterrence
- Eye-Level 360 Degree HD Video Streaming and Recording
- People Detection During Certain Restricted Hours
- Thermal Anomaly Detection
- Automatic Signal Detection (whitelist, blacklist, excluded and unknown MAC addresses)
- Live Audio Broadcast (similar to public address system)
- Two-Way Intercom (human-to-human voice interaction)
- Pre-Recorded Messages (may be played manually, randomly, by time or by detection)
- Remote monitoring from anywhere utilizing our state-of-the-art KSOC UI

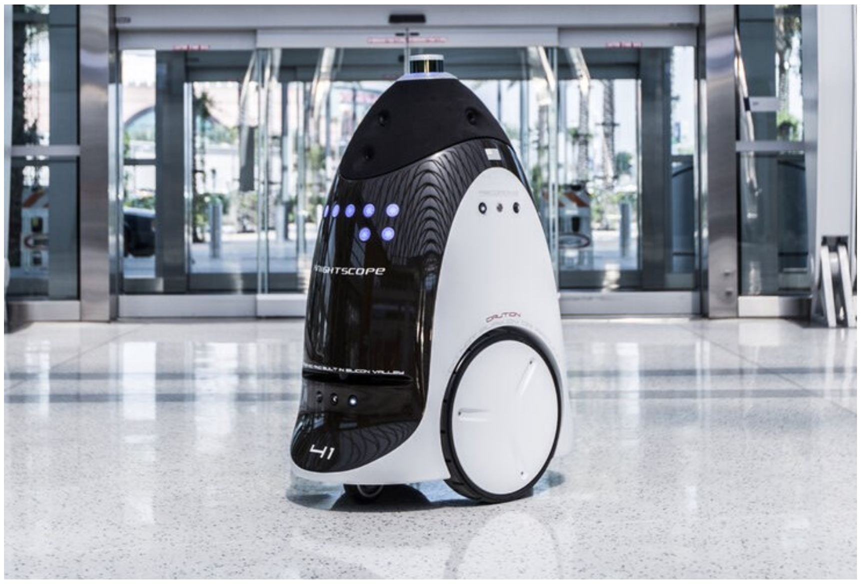


Figure 3.1.3: Knightscope K3 ASR
(Reprinted with permission from Stacy Stephens)

3.1.4 Knight Rider

Knight Rider was a an autonomous 1/10 scale race car built by the UCF Real Time Operating Systems laboratory for participation in the F1/10 research racing competition. The F1/10 competition asked graduate students to develop algorithms to traverse a race track as quickly as possible using remote-controlled cars at 1/10 scale.

Knight Rider was built using a 1/10 scale model of a Ford Fiesta. It had a custom PCB for managing power and for switching between teleoperated and autonomous modes. For autonomous driving, it used an Nvidia Jetson T100 to run ROS, which ran the find-the-gap algorithm to seek the best direction to drive in constantly. It did not use mapping or any planning algorithms. It was a behavioral robot. The Jetson sent commands to a Teensy 3.2, which forwarded those commands to the motor controllers.

The RC car was modified to fit the purpose of the robotics competition by

- Removal of the plastic cover for the vehicle
- Construction of an additional layer over the chassis for the Jetson and other electrical components to reside
- Construction of a bracket for the front of the robot to secure the LiDAR
- Installation of ROS on the Jetson
- Creation of a PCB to manage power from a battery of larger capacity than the RC car was designed for
- Creation of a PCB with a physical switch to turn on and off the throttle motor, as well as a physical switch to go between autonomous and remote teleop control
- Setup of a dedicated router to facilitate communication between the robot and the user station

One of the members of this team, Jade Zsiros, worked on the design and construction of this robot. Much of the inspiration for how to tackle this project comes from this robot, but most of the hardware will be different and the robot will complete different objectives.

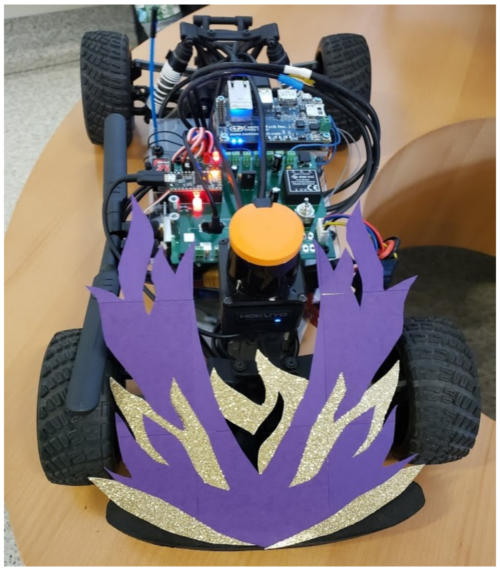


Figure 3.1.4: Knight Rider autonomous race car

3.2 Autonomous Vehicles

A requirement of our project is to be able to traverse an environment autonomously. This means the robot can move around its environment without receiving input from a user. This section will talk about the different methods used to create autonomous robots and which one we will use for our project and why.

3.2.1 Automatic Guided Vehicle

Automatic Guided Vehicle, henceforth AGV are portable robots which follow designated markings (wired, guide tape, laser target navigation) to navigate their environment. With common applications in the industrial sector these robots are often found in warehouses transporting heavy goods from one location to the other wherein their navigation simply requires them to trot along a directed path denoted by the markings.

Wired For this method of navigation within AGVs, a wire with a radio signal running throughout it is placed into a cut slit in the ground. The wire transmits a radio signal to the bottom of the AGV, which transmitter detects the relative position of the AGV from the radio signal. The steering decisions of the AGV are made to follow the signal and ensure that it gets from start location to the desired final location by running along the wire in the ground.

Guide Tape With two variations available, magnetic, or colored tape. The AGV is configured in such way that allows for it to have appropriate guidance following the path of the tape, a major advantage of tape when compared to a physical wire is it's ability to be removed and adjusted as the path or course of the vehicle needs to be adjusted without cutting into the ground. While magnetic tape is more costly, it has it's advantages lies in that it can dirty, or smudge without loss of ability and does not require power like the wire.

Laser navigation For this method of navigation, reflective sheets are placed upon obstacles and the AGV carries with it a rotating laser transmitter and receiver (similar to Lidar). The AGV is then able to determine the angle and distance of an object relative to the AGV, wherein it will compare currently transmitted information of it's surroundings to a prerecorded map and path. It will then adjust to assure that it follows the desired path due to the constantly updating information of it's current position relative to the environment.

3.3 Reactive/Behavioral Navigation Algorithms

These algorithms, as opposed to being controlled by a navigator, or following some set path overlayed on top of an existing mapping of an environment, seek to react to the environment and navigate through a series of conditions that determine the behavior of traversal entirely autonomously. These are generally considered more difficult to implement as you must consider the wide range of conditions your autonomous vehicle may encounter and create a set of cases in which wholly represent the surrounding world.

3.3.1 Follow the Wall

Follow the wall algorithms, are as the name implies, dependant on the existence of wall, somewhere within view of the robots sensor (typically a laser scanner, in our particular case, likely a 2D, 360 degree Lidar). The wall following algorithm desires not to model every possible occurrence of every type of wall, but rather to have a reactive algorithm that contours the curvature of the wall and maintains a set distance from that wall. For the purposes of explanation, and simplicity we'll consider the case where only one wall is adjacent to the robot (LookSee). Ideally, LookSee will maintain the wall within it's viewing angle, whilst maintaining a set distance from that wall. This will be accomplished through a wall model being computed in through the information obtained from a 2 dimensional data line scan. From that data, LookSee will attempt to maintain viewing angle of the wall, and traverse a line perpendicular to the wall. In the event, we encounter a corner LookSee would ascertain through data leading readouts (shown M2 in the diagram) that the current path (the dotted green line), running perpendicular to the wall will lead us into another wall (denoted by the proceeding red M2, and red dotted line showing a collision course), and upon the event where the oncoming wall is determined to be too close, LookSee would alter the viewing angle that maintains a set distance from the observed (presently adjacent) wall, to the oncoming (opposite) wall. Effectively turning, and following the new wall. In theory this would continuously traverse along a series of walls crawling along the wall until the desired destination is reached, or in our case, continuously patrol in a designated environment.

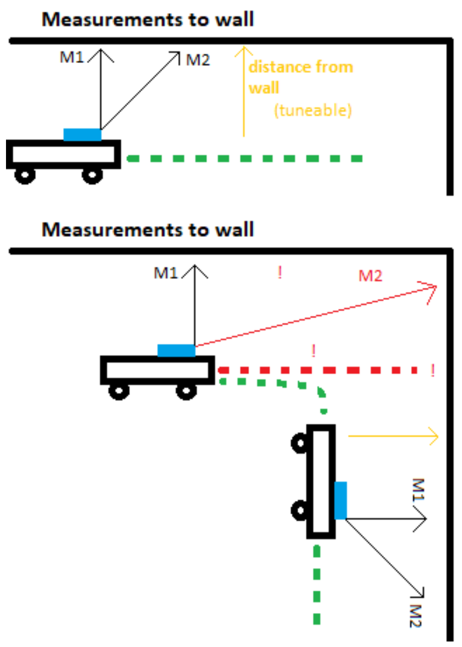


Figure 3.3.1: Finding the gap
(Made by Austin Pena)

3.3.2 Find the Gap

This is our modified algorithm which Jade is most knowledgeable about, though Austin has a strong desire to learn about autonomous navigation (A majority of the research seems to be gated behind paywalls for research papers, but conveniently Jade has worked with a similar algorithm in her research lab; described in detail below.)

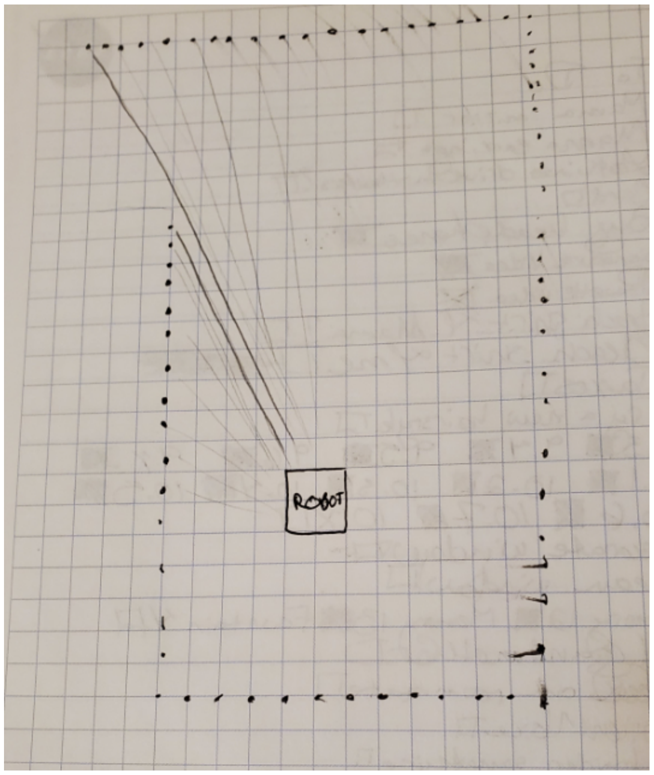


Figure 3.3.2: Finding the gap

Please refer to the above figure, detailed within it is LookSee placed in a square room with only one outlet, (represented by a solid square titled [ROBOT]) whose Lidar beams can be seen scanning the room at a particular instant in time, please note the longest beams traversal path relative to the beam prior to it, which pings off the wall of the corridor entering the room, and the wall of the room respectively. Herein, LookSee would determine (through knowledge of the common distance between two consecutive points) that an anomaly has occurred and there exists some point where there exists a gap between the walls. LookSee would then traverse towards this gap, and as it progressed towards the gap, would ascertain more information about the path and adjust accordingly. (Please do not interpret the path of LookSee's sensors to mean the LookSee's present velocity vector.)

As we've progressed in our understanding we've made several improvements to our initially conceptualized algorithm described above. Firstly, the naive approach of simply approaching the largest gap didn't take into account the angle of approach and issues that lie with a shallow approach angle (hitting the corners of walls). It was initially thought that we could solve this issue by extending the wall of the nearest point by a few degrees (roughly equivalent to half the width of the robot), but this turned out to be a mere patch and not a true solution. With this idea in mind we decided that a superior solution would be to inflate all walls (identified through discrepancy readouts) by this aforementioned radius. Careful to only extend the wall of a shorter distance of the wall of a further distance as not to muddle the readouts and truly extend our desired walls. This effectively translates LookSee into a point object within the Lidar space, and if a gap existed within the the inflated readouts LookSee would definitely be able to traverse the gap. With this in mind we set our destination towards the furthest readout and appropriately dodged all walls with ease.

3.4 Deliberate and Hybrid Navigation Algorithms

Deliberate robot architecture is a school of design for robotics where a robot attempts to create and understand a model of the environment, or ”world”, before completing its required tasks. The world model may or may not be updated during the course of the robot's mission. An example of a robot with deliberate architecture may be one that has a map uploaded by the programmer, which it localizes itself to during its tasks. Another example could be a robot that first roams a building to build a map before it begins to execute its mission. A key component of deliberate architecture is planning moves in advance, with minimal adjustment on the fly. Planning algorithms such as A* are typically used to decide the route a robot will take in advance of beginning a journey.

The hybrid robot architecture blends the deliberate and reactive robot architectures, and in practice the majority of robots who have any deliberate traits use the hybrid architecture. SLAM, where a robot creates a map as it is traversing an environment for the first time, is one such example. The robot is trying to complete its mission while it creates the plan, and the world map is updated and a new version of the plan created every frame.

3.4.1 SLAM

Please be conscious that this is a very far stretch goal, but it is an incredibly fascinating topic with a vast number of real world applications that could likely transition into a career, thus is worth researching at the very least. Our team, did not have the time to execute this on LookSee, but had chosen to research it and at least dabble in its use for our own personal development.

SLAM In Our Use Case SLAM (Simultaneous Localization and Mapping) is a far stretch goal, but an important consideration for our project if we have an abundance of time. SLAM algorithms have common applications in navigation, robotic mapping, with real world examples being self driving cars, or pilot-less aerial vehicles (drones). SLAM, as it's name implies is the problem of constructing or updating a map of a previously unknown environment whilst simultaneously keeping track of a users position within it.

In our case our optical sensor is a two dimensional sweeping Lidar (most likely due to cost restraints) though SLAM implementations exist for purely visual sensing (cameras) as well.

Accordingly it follows that, the construction of the environment is a function of the data available, and different algorithms are used for the various forms of sensing ranging from visual sensors (like Lidar, or camera input) to tactile sensors which contain information points in the immediate vicinity of the agent. We as a group for the purposes of our project do not have the: required post-graduate knowledge, or an extreme abundance (multiple years) to invent a SLAM algorithm. We opted not to include SLAM in our final iteration of the project, though there exists a wide repository of information regarding the SLAM problem in the robotics community with open source solutions widely available. I've selected a few sample algorithms and the criteria which make them valid to be detailed below to further expound upon potential solutions.

SLAM Algorithms Considering our limited time, and the fact that this is a stretch goal and not a major portion of our project, merely an aspect that could potentially be implemented. It's chiefly important for our library to be accessible: implemented in ROS, in a familiar programming language; and applicable to the equipment available: (a budget Lidar). Ideally, no odometric data is required to be fed in and is calculated.

BreezySLAM BreezySLAM is an excellent choice, described to be: "simple, efficient, multiplatform, open-source Python library" for SLAM. The way it works is by wrapping existing implementations of SLAM (specifically TinySLAM, also known as CoreSLAM) algorithms, and providing a Python API that runs "nearly as fast as the original C code." The code works off of the Monte Carlo localization algorithm formally, choosing to wrap it's C implementation CoreSLAM for it's low memory usage. BreezySLAM conviently provides three pure python classes detailed as follows: Robot (robot odometry to velocity method), Laser (laser parameters), and Odometry (calculated at each instant.) These will be the classes modified by users to input information. The remainder of the code is wrapped C instructions.

HectorSLAM Described as another solution that doesn't require odometry like Gmapping, though it can be used with odometry data. It instead leverages the power of of modern Lidar systems which have a high refresh rate. HectorSLAM produces map at a low computational cost as well and is still maintained and updated on it's ROS page working off a publisher/subscriber model, the package itself is coded in C++.

3.5 Computer Vision

Computer vision can be used for a variety of different applications from object detection to autonomous navigation. In our case we will be using it for object detection, specifically human detection. We need to be able to detect when a human is fully or partially in view. Once we identify a person we can use our thermal sensor to be sure it is indeed a person with a heat signature. There are several different methods for human detection, we will analyze each method to determine the best possible choice for our application.

3.5.1 Viola and Jones

The Viola and Jones method for single object detection uses adaboost in cascade over Haar features (wavelets). Haar features are blocks containing two, three, or four rectangles which are used to calculate the pixel value difference within these rectangles. These features are trained by using a dataset of positive and negative reference images, where only the features that are best able to distinguish from the positive and negative images are selected. These features can then be used for detection by placing them over the image where you want to detect and make the calculations based on the chosen Haar features. If that part of the image passes it goes to the next Haar feature to check if that same part of the image also passes the next check. This happens until it fails. If it fails, that portion of the image is ruled out as not being the object you are looking to detect. This is done over the entirety of the image until the whole image has been looked at. OpenCV has a library with pre-trained models for full body, upper body, and lower body detection. This method requires little computation overhead and is quick. Most cameras and smartphones that have face detection use this method to detect faces and place a bounding box around them. In practice this is not as accurate as the other methods and produces some false positives and false negatives.

3.5.2 Histogram of Oriented Gradients (HOG) and Support Vector Machines (SVM)

This method takes an image and applies some preprocessing to it in order to normalize the image. This preprocessing includes color normalization (color to black and white) and resizing. Then the gradient vector, magnitude, and direction are calculated in batches of pixels. These are combined to create feature vectors. These feature vectors are then normalized and end up creating a Histogram of Oriented Gradients. This can then be convolved with a model HOG of an object you would like to detect. This convolved image can then be fed into a classified like a SVM in order to classify if it is indeed the object you are looking for or not. Again OpenCV has libraries to support HOG based human detection using pre-trained models. While this method performs better than the Viola and Jones method but still has some of the same drawbacks including flickering detection and incorrect boundary boxes. There are also issues when trying to detect human bodies from an angle that isn't straight on or if they are in an odd pose.

3.5.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks utilize deep learning to solve a variety of computer vision problems including image classification and object detection. An advantage to convolutional neural networks is that they can also do multiple object classification. This method is by far the most accurate and robust however it comes at a cost of needing increased computational power. There are many open source solutions for creating a CNN for human detection. The most common CNN architectures are:

- LeNET
- AlexNET
- ZF-NET
- GoogLeNET
- VGG-NET
- RESNET

3.6 Thermal Detection

The thermal data received from the Melexis MLX90640 is provided in the form of an IR array containing 768 pixels worth of data which then can be interpolated to a larger image using open source libraries on interpolation to produce a better image. In addition to being a low resolution thermal camera, it has accessible temperature information provided by the Melexis MLX90640 can be accessed and then compared to the known temperature values of a human being. Unfortunately, due to imcompatibilities with the Jetson Nano this portion of the project was discarded in favor of better traditional image recognition.

3.7 Sensor Fusion

Sensor fusion is the practice of using data from multiple sensors to glean information that couldn't be found otherwise or that the system would be less confident in. With the goal of sensor fusion, we want to integrate the many subsystems of sensors and create a more holistic view of LookSees environment. We can produce a more comprehensive visual representation by combining the LiDAR, Camera, and IR Array, Microphone will be first conceptually discussed in the subsections below and further simulated in Gazebo (3.11.9 Gazebo). After physically testing each subsystems components for accuracy, the subsystems will be tested in synergy, and modeled in simulation then physically tested in the real world. For a robot such as LookSee, we are exploring the following common styles of sensor fusion.

3.7.1 Camera and LiDAR

Both of these sensors are related to traversal of the environment with Lidar being the predominate case for traversal, though several algorithms for autnomous traversal exist for the camera, potentially if the camera isn't busy detecting any humans it could provide information to the Lidar for the navigational algorithms. Similarly, in the case of human detection perhaps the Lidar would be able to provide telemetry of the scanned human in addition to the visual distance seen by the camera.

3.7.2 Camera and Microphone

Effectively further with Camera and Microphone interaction to have dialogue to determine the cause or reasoning for this human being intrusion (after identification via Camera and Thermal Sensor and operator of LookSee is informed location within some area designated to be patrolled by LookSee and cause could effectively be determined for the intrusion.

3.8 Sensors

In order for our project to take in information from its environment we will require several different sensors. Objectives like autonomous control, obstacle avoidance, collision detection, people detection, etc all require various sensors that work together in order to provide the proper information so the robot can act accordingly. This section explores the different kinds of sensors that were considered along with the sensors we chose and why.

3.8.1 Camera

One of the key features of our robot is the camera to give us the ability to see the robot's surrounding. Our project mainly depends on computer vision and being able to use the camera feed as data for our code. We are looking for something reliable that will be easy to implement in our design and code. Our main requirements are video quality, frames per second, field of view, and product availability. Having a high resolution camera will provide us with the video quality we need to insure our code will be able to read what it is seeing properly. This also goes for frames per second. If the camera has a low frame rate, then it would cause difficulties to process movements and objects. An example of this would be someone running in front of the robot. With a low frame rate the camera alone would not be able to detect a persons face. Field of view is a essential requirement for us because the more we are able to see the more area we can detect for intruders.Usually product availability is not an issue, however since the Corona Virus 2019 pandemic, webcams have been out of stock due to schools moving over to remote instruction, requiring students to use a webcam during lectures.

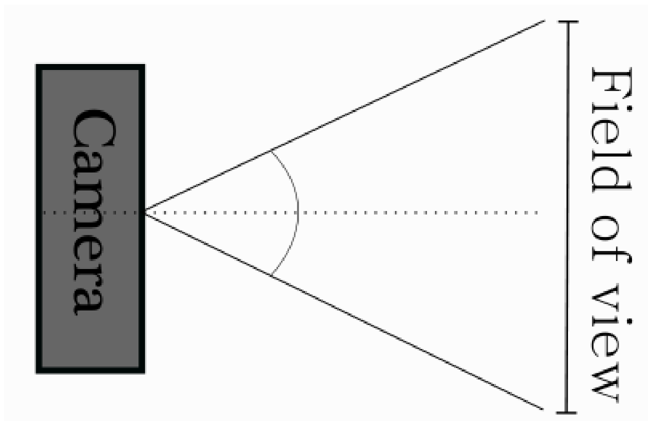


Figure 3.8.1: Camera Field of View Diagram

Our options that we are considering are a OpenMV H7 [6], a Logitech C992 [7], a Logitech StreamCam [8], and a Canon EOS Rebel T8i [9]. Our first three choices are very easy to implement in our design and would work just fine a prototype, maybe even as the finished product. However, other than for the price a DSLR camera would give us the best quality and best accurate object detection. This might be an option for us due to the supply of webcams being sold. We wouldn't have to buy it since one of our group members is willing to lend it for the purpose of this project.

Camera	OpenMV H7	LOGITECH C922	LOGITECH STREAMCAM	Canon Rebel T8i
Megapixels	2.0	2.0	2.1	24.1
Resolution	480p	1080p	1080p	1080p
FPS	60	30	60	59.954
Field of View	78°	78°	78°	74°
Weight	30g	162g	150g	515g
Connector	USB - A	USB - A	USB - C	USB - A
Price	\$65.00	\$99.99	\$169.99	\$899.99

Table 2: Camera Specifications

3.8.2 Lidar

Lidar, etymologically comes from light and radar and later adopted an appropriate acronym: "Light detection and ranging". We're considering this type of sensor as it's data is valuable to autonomous vehicle operation which is crucial to LookSee. Lidar's sensing technology is as it's name applies composed of a laser which is able to ascertain position or motion using the principles of radar with the exception of laser radiation instead of microwaves. Typical Lidar scanners consist of a laser (typically ultraviolet, visible, or near infrared light to image an object), and a sensor which detects the reflected laser beam and a motor to move the laser and receiver. Lower cost sensors will typically return a two dimensional mapping which consists of an array of discrete data points corresponding to the motion trail from the laser. More expensive sensors are capable of creating a point cloud, as opposed to a point line, mapping in three dimensions.

Applications extend far beyond robotics and autonomous vehicle operation, including applications in agriculture, archaeology, biology and conservation, atmospheric, astronomy and physics, spaceflight, law enforcement and military. In fact it's first applications in meteorology where it was used to measure clouds and pollution by the US National Center for Atmospheric Research whose research and early applications helped progress the technology.

For our particular use case, we are interested in autonomous vehicle navigation, and the data to be gleamed from a two dimensional line mapping (as opposed to a point cloud). Though Lidar is incredible in terms of the number of data point gleamed, and the accuracy of the data obtained, the limiting factor and chiefly important to a project of this scope will be the cost of the Lidar sensors.

RPLIDAR A1M8 RPLidar A1M8, hence forth referred to as the RPLidar is a low cost, 2D laser scanner with a 360 degree scan capable of detecting within a 6 meter range. The produced 2 dimmensional point line data is scanned with a variable frequency of 5.5Hz, with a configurable maximum 10Hz maximum, and a 2Hz minimum. The A series in the RPLidar uses the principles of triangulation where objects reflect back and dependent on the angle of reflection to the known separated distance of the receiver distance can be derived. All for the low price of around $100.

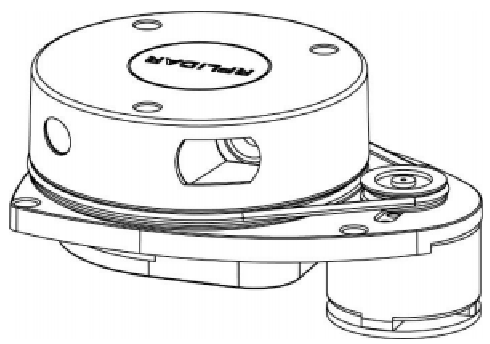


Figure 3.8.2: RPLidar A1M8
(Permission Pending)

Hokuyo UST-10LX The Hokuyo UST-10LX Scanning 2D Laser Rangefinder is a small, accurate, high-speed device. Touting a wide 270 degree field of view with a 10 meter range, with millimeter precision. With a low power consumption it seems like an ideal pick. As opposed to the RPLidars, 360 sample points, and 6Hz frequency. Hokuyo instead offers the following figures: 1081 measurement steps, and a scan speed of 25ms. Roughly 40Hz frequency. (I'm assuming this means what RPLidar calls sample point, by knowledge ascertained of the information available from the documentation.) This sensor comes in a modest, and reasonable $1,600.

Figure 3.8.3: Hokuyo UST-10LX
(Permission Pending)

Velodyne Puck VLP16 The Velodyne puck is an incredibly small, and accurate sensor with a real time, 360 degree field of view with a 30 degree field of view. The sensor boasts and impressive 16 channels, with a measurement range of up to 100m with a range accuracy within 3 centimeters. With a variable 5-20Hz rotational rate and and an incredibly tight angular resoltion this is a superior sensor. Offering a full 3 dimensional data point cloud at a rate of 300,000 points per second in single return mode, and a 600,000 point per second rate in dual return mode. This sensor is top of the line and can be seen on cutting edge applications like Boston dynamics, consumer available SPOT robot as the Lidar attachment of choice. This Lidar demands an entirely justifiable contribution of $8,800.

Figure 3.8.4: Velodyne Puck
(Permission Pending)

Conclusion Unfortunately the technology for Lidar sensors isn't widespread in application enough to drive down the cost of burgeoning technology. Due to it's bleeding edge capability the entire field is striving for a higher quality, more improved laser and not necessarily a cost effective, lower quality laser. With this in mind, the majority of Lidar sensors on the market today are in the thousands of dollar range, which is far beyond our budget as self funded students. As a result we elected to go with the RPLidar A1M8 which is the a functional Lidar at an acceptable (if expensive) price.

	RPLidar A1M8	**Hokuyo UST-10LK**	**Velodyne Puck VLP-16**
Distance Range	0.15m - 12m	0.06m - 30m	~100m range
Measurement Resolution	<0.5mm	+- 40 mm.	+- 3cm
Scan Angle	360	270	360 by 30
Angular Resolution	<1 degree	0.25 Degrees	(Hl) 0.1 - 0.4 (V) 2.0
Sample Rate	2000 Hz - 8000 Hz	40Hz	5Hz - 20Hz
Sampling Points	360	1081	~600,000 points a second
Working Enviornment	Indoor	Indoor	Indoor
Weight	190g	130g	830g
Operating Voltage	9.9V - 15.5V	12V-24V	9V- 18V
Price	$100	$1,600	$8,800

Table 3: Lidar Specifications

3.8.3 Thermal Sensor

Thermal imaging, also referred to as Thermography is an infrared imaging technique detecting radiation in the infrared range of the electromagnetic spectrum. These images produced as a result of radiation are called thermograms. All objects with a temperature above absolute zero emit infrared radiation. It's got a swath of applications, though most instruments are limited to detecting surface temperatures only.

For our application, it's a bit of a stretch goal with an to perform thermal imaging, but we as a group would certainly like some form of secondary confirmation that the (person shaped object) detected by our camera and computer vision algorithm. Potentially we could even use the data gathered from our thermal sensor to direct the camera at a suspiciously hot object to determine if it's a person or not. We're interested in the technology and are using our senior design project as a extra circular course to investigate this burgeoning area within sensor technology.

Included below are some of the options we considered for our project, since we've got a particular interest in developing a thermal image as opposed to just a particular readout from a sensor. It was important for us to consider the IR Arrays, as they are composed of sensor(s) with multiple readouts, from these multiple temperature readouts an image is able to be interpolate that data onto a pixel grid to compose an image.

PANASONIC Infrared Array Sensor Grid-EYE (AMG88) The Panasonic GRID-EYE is an 8x8 (64 pixel) infrared array sensor with a 60 degree viewing angle offering an I^{2}C digital output for thermal presence, detection, and temperature values with a frame rate between 1 to 10 frames per second. It's got a variable input operating voltage (3.3V-5V) with both a high gain and low gain amplification factor. Typical readouts are accurate within +/- 4.5F for high gain, and +/- 5.4F for low gain. Setup time is a combined 65ms with 50ms being spent to enable communication for setup, and the other 15ms to stabilize the output after setup. Costs vary generally around the $25 price point.

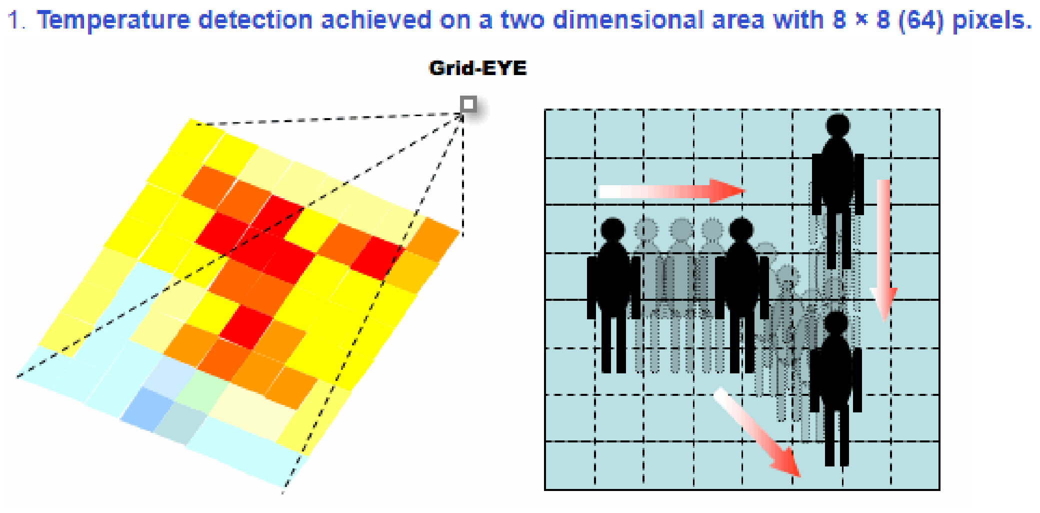

1. Temperature detection achieved on a two dimensional area with 8 × 8 (64) pixels.
Grid-EYE

Figure 3.8.5: GRIDEYE Thermal Image
(Permission Pending)

Melexis MLX90640 32x24 IR array The Melexis MLX90640 Far Infrared Thermal Sensor is a 32x24 (768 pixel) offering two field of view options: 55 x 35 degrees, and 110 x 75 degrees. Integrates easily with an I^{2}C digital interface, with a programmable variable refresh rate between 0.5 Hz and 64 Hz. With a stricter 3.3V supply voltage, and readout accuracy within +/- 1C. The setup time is, described in the data sheet as $T_{validdata} = 40 + 500, ms$ for a given example refresh rate of 2Hz which is the default value. Though, they note below that the thermal stabilization time necessary to reach specified accuracy can be up to 4 minutes. The price for this sensor ranges generally around the $60 price point.

FLIR Lepton 2.5 The FLIR Lepton 2.5 is a long wavelength (8μm to 14 μm) infrared camera roughly the size of a dime. This sensor offers a 80x60 (4,800 pixel) readout with a 50 x 63 degree field of view. This sensor is controlled through a 32 pin socket interface connector (CCI, claimed to be very similar to I^{2}C). With video interfacing over two wire serial control interface (SPI). Offering a 8.6Hz refresh rate, which is roughly under 9 frames per second.

Focused on smartphone application this sensor offers phone standard desired input voltage of 2.5V to 3.1V. FLIR offers an incredibly impressive sub 1.2 second promised time to image, with an integrated shutter to allow for internal correction and improved image quality. Thermal sensitivity is accurate to ¡50K or roughly (0.05 C). The price for this sensor comes in at $200.

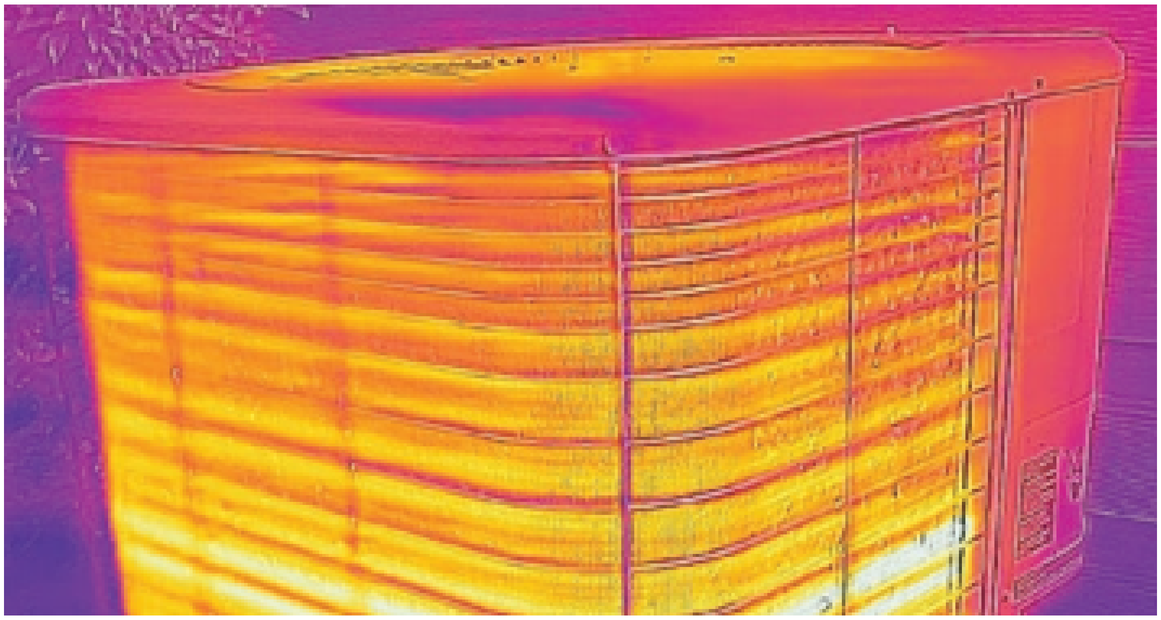


Figure 3.8.6: FLIR Lepton Micro Thermal Image
(Permission Pending)

Conclusion In conclusion, for our price point of the various options detailed above we found the GRID-EYE to be to low resolution for our application and the FLIR Lepton was a bit out of our student funded budget so we decided to go with the Melexis MLX90640. Unfortunately, due to imcompatibilities with the Jetson Nano this portion of the project was discarded in favor of better traditional image recognition.

3.8.4 Microphone

Another feature of our robot is to be able to listen in on the robot's surrounding. In a scenario of a break in loud audio levels will help inform the user of a unusual sound. Picking the right microphone will be a bit difficult because we don't need something very high end that costs a lot, nor do we want something very cheap that will distort any incoming audio. Another aspect we are looking for is the microphones form factor. As it will be mounted to the robot we don't need something big that create unnecessary weight and potentially block vision from our camera. The microphone must be compatible with our OS, and must not be wireless as they are more prone to catch interference, and require the need to be recharged. Our options that we are considering are all small form factor microphones, a ANTLION AUDIO MODMIC [10], Audio-Technica ATR4750 [11], and a SHURE Centraverse Lavalier Condenser [12].

When considering our options we look at all the specifications that are labeled in our Table 4 below. For polar pattern there are two types, omnidirectional and unidirectional. Omnidirectional microphones are capable receiving signals from all directions while unidirectional can only receive from one direction. In our case a unidirectional sounds most ideal, however it may pick up a lot of noise from the robot's movement. Microphone frequency or frequency response is the measurement of the microphones audible sound frequencies. With a more dynamic frequency response we are able to pick up all frequencies of sounds. The SHURE CVL[12] having the widest range compared to the other two. Microphone sensitivity is measured in decibels(dB) which is way of measuring sound intensity. All these microphones are very sensitive to sounds that the human ear can't pick up. All of our current options seem to very light and we wont have to worry about weight, however our main determining factors will be price, and compatibility with our robot design.

Microphone	MODMIC	ATR4750	SHURE CVL
Polar Pattern	Omnidirectional	Omnidirectional	Unidirectional
Frequency	100Hz - 10kHz	50Hz - 13kHz	50Hz - 20kHz
Sensitivity	-36dB	-48dB	-43.5dB
Weight	100g	180g	25g
Connector	USB - A	USB - C	Mini XLR
Price	$79.95	$24.99	$39.00

Table 4: Microphone Specifications

Conclusion In conclusion, we ended up selecting a webcam that had a built-in microphone that worked very well for our purposes, and did not purchase any of the above microphones.

3.8.5 Gyroscopes and Accelerometers

The difference between an accelerometer and a gyroscope is the gyroscopes ability to maintain and measure the rate of rotation around a particular axis. For the rate of rotation around the roll axis around an aircraft or autonomous drone this can be crucial. Accelerometers measure linear acceleration of movement or directional movement of a device, like an autonomous vehicle or of a phone. A practical example of sensor fusion of an accelerometer and gyroscope in a autonomous vehicle would be an accelerometer to measure both the linear velocity and the angular rotational velocity from the gyroscope, in practice this would mean directional movement of the vehicle and lateral orientation or tilt can be determined by the gyroscope.

Triple Axis Accelerometer - ADXL345 This low power, small thin three axis MEMS (micro-electromechanical system) accelerometer with high resolutions (13-bit) measurements up to ±16g. Digital (16-bit) ouput data with programmable memory ranges of (±2g, ±4g, ±8g, ±16g) is available over I^{2}C digital interface or SPI (3, or 4 wire). The ADXL comes with a lower power mode with intelligent motion detecting power management and custom threshold sensing. In addition to this, it has a active acceleration measurement at extremely low power dissipation. Specific to the ADXL345 are the special sensing functions for detecting free fall, inactivity, and activity sensors for detecting the presence or lack of motion and tap sensing (tap and double tap). All for the reasonable price of $9.95.

3-Axis Accelerometer - BMA456 This ultra small, three axis, low -g high performance acceleration sensor has specialized for low power consumption and demanding consumer electronics applications. Capable of detecting wrist tilts, tap or double tap, and step counting in wearable electronics. The BMA456 measurements using ±16g Digital (16-bit) ouput data, with programmable memory ranges (±2g, ±4g, ±8g, ±16g) and a 0.06mg resolution (in ±2g range). This accelerometer has a 1kB FIFO data package. Data is available over SPI and I^{2}C interface. This sensor comes in at the low price of $2.82.

High Performance 3-axis Accelerometer - IAM-20381 This three axis high performance, "MotionTracking" accelerometer is designed for automotive applications has a small form factor and 16-pin LGA package. Additional features include 512-byte FIFO that can lower traffic on the serial bus interface for reduced power consumption and burst reading of data in low power mode. The IAM 20381 has 16 bit output data with programmable memory ranges (±2g, ±4g, ±8g, ±16g). This data is available over SPI and I^{2}C interface. Available for $10.78.

Conclusion Of all the Accelerometers within the allotted budget, we found the MPU6050 to be equally as capable as the other Accelerometers adjacent in price point. In fact, despite being lower cost than the IAM20381 the MPU has nearly identical resolution and sensitivity, with a lower current consumption in all modes, double the size of the FIFO buffer of the IAM20381 and at a lower price point then the both the IAM20381 and ADXL345. Unmentioned in all of that, is that the MPU-6050 additionally has an on board gyroscope providing an additional 3-axis from that, and 3 additional access from it's on board processor which combines the axis. Ultimately, we never ended up pursuing an accelerometer in order to keep final cost down and LookSee remained functional without it.

Part #	ADXL345	BMA456	IAM-20381	MPU-6050
Number of Axis	Accelerometer	Accelerometer	Accelerometer	Accelerometer and Gyroscope
Number of Axis	3	3	3	9
Digital Resolution	16-bit	16-bit	16-bit	16-bit
Measurement ranges	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g, ±16 g
Sensitivity (calibrated)	3.9 mg/LSB	0.06 mg (in ±2g)	16384 LSB/g at ±2g	16384 LSB/g at ±2g
Digital inputs/outputs	I2C and SPI	I2C and SPI	I2C and SPI	I2C
Supply voltage (VDD)	2.0 to 3.6	1.62 to 3.6 V	1.71V to 3.6V	2.375V to 3.46V
I/0 supply voltage (VDDIO)	1.7 to VDD	1.2 to 3.6 V	1.71V to 3.6V	1.8V or VDD.
Current consumption				
- full operation	140 μA	150 μA	390 μA	140μA
- low-power mode	30 μA	14 μA	57 μA	10μA
FIFO data buffer	”32 Level FIFO”	1024 byte	512 byte	1024 byte
LGA package	3 x 5 x 1 mm^3	2 x 2 x 0.65 mm^3	3x3x0.75 mm^3	4x4x0.9 mm^3
Price	$9.95	$2.82	$10.78	$8.31

Table 5: Accelerometer Table

High Performance Gyroscope IAM-20380 This 3-axis high performance gyroscope has a small form factor with 16-pin LGA package. Additional features include 512-byte FIFO that can lower traffic on the serial bus interface for reduced power consumption and burst reading of data in low power mode. The gyroscope sensors are user programmable range of ±250 dps, ±500 dps, ±1000 dps, and ±2000 dps and integrated 16-bit ADCs. This data is available over SPI and I^{2}C interface.

3-Axis Gyroscope - A3G4250D The A3G4250D is low power 3-axis angular rate sensor touted for it's stability at zero rate level and sensitivity over temperature and time. The in chip interface is capable of communicating through SPI and I^{2}C the data outpt is 16 bits at ±245 dps. This sensor also comes with 8 bit temperature data output additionally. Avaiable for $14.55.

3-Axis Gyro/Accelerometer IC - MPU-6050 The MPU-6050 is a MEMS device, combining the 3-axis gyroscope and 3-axis accelerometer on the same integrated chip. Additionally on the chip is a onboard Digital Motion processor, which performs MotionFusion algorithms to create a 9-axis data. This data is adjustable and user programmable gyroscope with a full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g. This data is available over I^{2}C interface. All of this functionality comes at the low cost of $8.31. We believe this to be an incredible value for both it's use as a gyroscope and an accelerometer combined on chip with a unique onboard processor.

Conclusion Of all the Gyroscopes within the our budget, we found the MPU6050 to be equally as capable as the other Gyroscopes adjacent in price point. The IAM20380 has nearly identical measurement ranges and sensitivity. The IAM20380 has a lower full power consumption and more tolerant supply voltage, and I/O supply voltage. Though the MPU has twice as big a data buffer, and has an additional built in accelerometer which provides an additional 3 axis, as well from it's on board processor which combines the axis given from the accelerometer and the gyroscope. Discussed above in the conclusion for the accelerometers that this multi functionality, low cost, and similar performance to both costlier accelerometers and gyroscopes lead us to believe that this was the best decision for performance and cost. Though we did not end up purchasing a gyroscope, we believe that this would have been a good addition to our project and would have opened other implementations of SLAM, and thus given us opportunities beyond our limited capabilities given the current equipment.

Part #	IAM-20380	A3G4250D	MPU-6050
Number of Axis	3	3	9
Measurement ranges	$\pm$250 dps, $\pm$500 dps, $\pm$1000 dps, $\pm$2000 dps	$\pm$245 dps full scale	$\pm$250 dps, $\pm$500 dps, $\pm$1000 dps, $\pm$2000 dps
Sensitivity (calibrated)	131 lsb/dps at 250dps 16.4 lsb\dps at 2000dps	8.75 mdps/digit	131 lsb/dps at 250dps 16.4 lsb\dps at 2000dps
Digital inputs/outputs	I2C or SPI	I2C or SPI	I2C
Supply voltage (VDD)	1.71V to 3.6V	2.4 to 3.6	2.375V–3.46V
I/0 supply voltage (VDDIO)	1.71V to 3.6V.	1.71 to VDD	1.8V or VDD.
Current consumption			
- full operation	2.6 mA	6.1 mA	3.8 mA
- low-power mode	1.6 mA	1.5 mA	140 μA
FIFO data buffer	512 byte	32-slots (16 bit)	1024 byte
LGA package	3x3x0.75 mm^3	4x4x1.1 mm^3	4x4x0.9 mm^3
Price	\$12.37	\$9.05	\$8.31

Table 6: Gyroscope Table

3.9 Single Board Computer

Single board computers are full computers that have an operating system, some type of storage, and a variety of input/output options. Using and programming on a SBC is similar to how you would use your normal desktop or laptop making it easy to connect to, change code, and upgrade. As the members of our team are primarily computer engineers, we decided early and emphatically that we want our project to have a heavy focus on software development and have many opportunities for us to display our knowledge and skill in the languages we anticipate using most in the workforce. For this reason, we decided to use ROS (Robot Operating System), which runs code we write and facilitates its communication with the sensors and motors we will be using. ROS runs code in high-level programming languages such as Java, C++, and Python. For this reason, we are looking at single board computers that can run ROS, which runs on Linux. This board will handle all the heavy lifting when it comes to computation and networking. A separate microcontroller will handle sensor inputs and motor controls. This separates the base hardware control from the higher level computation.

3.9.1 Nvidia Jetson Nano

The Nvidia Jetson product line is made up of microcomputers designed and optimized for AI and robotics. With a powerful graphics processor, the Jetson is intended to allow programmers to use CUDA to accelerate complex computations such as those associated with artificial intelligence and computer vision.

The Jetson Nano is the smallest form factor on the product line, and is intended for a use exactly like ours - a small robot with several sensors that will need to use computer vision. However, as our group has no plans to use CUDA to accelerate our processing, it remains a question whether the additional cost for higher processing power would be worth it to our team.

3.9.2 Raspberry Pi 4

The Raspberry Pi product line is made up of microcomputers intended to power small inventions that require minimal computation, but more than that of the average microcontroller. Since the advent of the Raspberry Pi B, with 512 MB of RAM, newer Raspberry Pi models have become more and more powerful. The Raspberry Pi 4 can support up to 8 GB of RAM and a clock speed of 1.5 GHz, which, while not being the most powerful computer available on the market, rivals the personal computers, especially for the $75 price tag.

3.9.3 BeagleBone Black

Like its competitors, the BeagleBone black is a single board computer targeted at hobbyists and developers made by Texas Instruments. The biggest difference between it and its competitors is that the BeagleBone Black uses on board flash memory, meaning boot time is exceptionally quick. This also means getting up and running is quick and easy without the need to install any OS via a microSD. It also has more connectivity than the others in this list with two 46 pin headers providing an ample amount of connections for developers. The rest of the hardware is slightly outdated compared to that of the Raspberry Pi and there is only a single USB port for developers to connect to.

3.9.4 ODROID XU4

The ODROID XU4 single board computer is manufactured by a company in South Korea and has some interesting components that its competitors may lack. The XU4 is the cheapest out of the bunch but offers a decent amount of power for the price. The processor has eight cores and has a max clock of 2 GHz. It also has a gigabit ethernet port which many others SBCs lack. It also provides two storage options; microSD slot or 4GB 8-bit eMMC on-board flash storage. This option is toggled using a physical switch on the board. Unfortunately there is no bluetooth or WiFi built in making the ethernet connection the only way to connect to a network. Due to the more powerful processor a cooling fan also had to be added in order to keep temperatures in check. This is the only SBC that has moving parts which introduces additional points of failure.

3.9.5 Conclusion

After comparing various different single board computers using Table 7 we decided to go with the Raspberry Pi for initial prototyping. If we end up requiring increased computing power for AI applications we will opt for the Nvidia Jetson Nano. The ODROID XU4's lack of WiFi and additional moving parts resulted in it being dropped from consideration. The BeagleBone's extra connectivity isn't required for our project and the lackluster components compared to the Raspberry Pi at a similar price point made us also drop it from consideration. The massive price difference between the Raspberry Pi initially; though in the end, we decided to switch to the Jetson Nano for the previously mentioned reason as we ended up needing enhanced performance in object recognition.

	Nvidia Jetson Nano	**Raspberry Pi 4**	**BeagleBone Black**	**ODROID XU4**
CPU	Quad core ARM A57 @ 1.43 GHz	Quad core Cortex A72 (ARM v8) @ 1.5 GHz	AM335x ARM Cortex A8 @ 1 GHz	Exynos 5422 Cortex A15 @ 2 Ghz + Cortex A7 Octa core CPUs
GPU	128-core Maxwell	Broadcom VideoCore VI	PowerVR SGX530	Mali-T628 MP6
Memory	4 GB LPDDR4	2GB/4GB/8GB LPDDR4-3200	512 MB DDR3	2 GB LPDDR3
Storage	microSD	microSD	4GB 8-bit on-board eMMC	microSD
Pinout	J41 GPIO	40-pin GPIO	2x 46 GPIO	76 GPIO
OS	Linux	Raspbian	Angstrom	Linux
Power	5V / 4A DC power input	5V / 3A DC power input	5V / 2.5A DC power input	5V / 4A DC power input
Price	$30	$99	$60	$60

Table 7: Single Board Computer Specification Comparison

3.10 Microcontrollers

A microcontroller differs from single board computers in that they are single chip computers with no operating system, no drivers, and fixed storage and RAM. Microcontrollers generally require much less power, are smaller, and cheaper than single board computers. This makes microcontrollers ideal for handling low level hardware tasks. A microcontroller will handle the low level hardware communication and connect with the main control board to relay the sensor data. There are a variety of different microcontrollers that differ in speed, power, price, and communication potential. Many microcontrollers are broken into two different categories, ones where the core architecture is built by the vendor themselves and ones where they use an existing core architecture. In this section we take a look at some common microcontroller families and compare and contrast their different features in order to select the best one for the job.

3.10.1 MSP430G2452

This microcontroller is found on the MSP-EXP430G2 Launchpad development kit from Texas Instruments. This development board has been used by everyone in the group in various courses at UCF. This microcontroller is a low cost microcontroller with a 16-bit RISC based CPU and 8 KB of flash memory. The development environment for this microcontroller is Texas Instrument's own Code Composer Studio. This specific microcontroller is best suited when a low-power solution is needed.

3.10.2 ATMega328

This microcontroller is found in the Arduino Uno board. This microcontroller family is widely used in many different development boards. It is a low cost microcontroller with a 8-bit RISC based CPU with 4K to 32K of flash and 512 to 4 KB of RAM depending on the version. In order to develop on the ATmega family you will use the GCC compiler.

3.10.3 MIMXRT1062

The MIMXRT1062 microcontroller was developed for the Teensy board. It contains a 32-bit ARM based CPU with 1MB On-Chip SRAM. This chip offers a parallel camera sensor interface which would be ideal for our thermal image sensor. The development environment for this microcontroller is the same as the Arduino environment. It also sports a low-power run mode which operates at 24MHz.

3.10.4 Conclusion

There are more nuances to each microcontroller and each datasheet can be poured over in order to assess which one will best suited for the job. After comparing the various microcontrollers using Table 8 we decided to go with the Teensy board which contains the MIMXRT1062 microcontroller specifically for its parallel camera sensor interface.

	MSP430G2452	**ATMega328**	**MIMXRT1062**
Boards	MSP430G2452	Arduino Uno Rev3	Teensy 4.0
CPU	16 MHz	20 MHz operation	600 MHz
Memory	3.75 KB FRAM	1 KB EEPROM	1 MB SRAM
ADC	Single Eight channel 10-bit	Single Eight channel 10-bit	Two 16-channel 12-bit
Board Price	\$0.00	\$23.00	\$20.00
Chip Price	\$0.70	\$1.80	\$4.00

Table 8: Microcontroller Specifications Comparison

3.11 Software

In the section detailed below, we've included the various software subsystems that compose LookSee. Due to the spread nature of our group due to the spread of the COVID-19 virus, (precautions are such that we should socially distance ourselves and keep those who may be vulnerable to the disease sequestered away from society) we decided it would be best to break down each task into smaller systems that can't be delegated to the appropriate team member for them to work on and then reassembled and integrated into the group as a whole. With this principal in mind we tried to make sure each subsystem was modular with well defined inputs and outputs for ease of assembly.

3.11.1 Raspberry Pi OS

Raspberry Pi OS, formerly Raspbian, is the official operating system for all models of the Raspberry Pi, developed by the Raspberry Pi foundation. It is a free and open source software based on the Linux distribution Debian. ROS can run on Raspberry Pi OS with no additional configuration, and our current intention is to try to run all of our code on Raspberry Pi OS so that as much of the memory and processing power is available to us as possible. However, it may turn out that we cannot install all of the components we need on Raspberry Pi OS. If this happens, we will install another version of Linux.

3.11.2 Debian

Debian is a Linux Distribution developed by the Debian Project and is the basis for many other distributions including the above Raspbian, and below Unbuntu. The first stable release (version 1.1) on June 17, 1996. It's primary goal was that it was composed primarily of free and open source software.

3.11.3 Ubuntu

Ubuntu is a Linux distribution based primarily off Debian with three available versions for Core, Desktop, or Server users. It's initial release was October 20th, 2004 (15 years ago) and most recent update being Ubuntu 20.04 released on April 23rd, 2020. It has remained incredibly popular for it's ease of use, and stability as it was initially aimed towards personal computer usage, and has remained a popular choice as such.

3.11.4 Docker

Docker is an open-source solution for deploying applications in containers (segregated sandbox environments) which then run on a host operating system. The main benefit to this type of container service is that each piece of the software is run in a container which has its own dependencies and libraries. These containers not only decouple the application from specific libraries and dependencies but also from the host operating system. Docker containers can be run on a multitude of different operating systems and they are fairly lightweight and are not nearly as resource heavy as something like a virtual machine. These containers will allow individual developers to be able to get up and running really quickly and it ensures that if it works on the developers system, it will work on every other system. Since this project will require a fair amount of trying out different things, these containerized environments will allow us to change things without affecting the rest of the software stack. It will also allow developers to collaborate while writing most of the main software more effectively with less hiccups. These advantages are why many individuals and companies are using Docker in order to develop their own systems. In the image below [3.11.1] you can see that interest and research about Docker is increasing at a rapid pace and therefore it is not only advantageous for development but also to fit the direction the industry is moving.

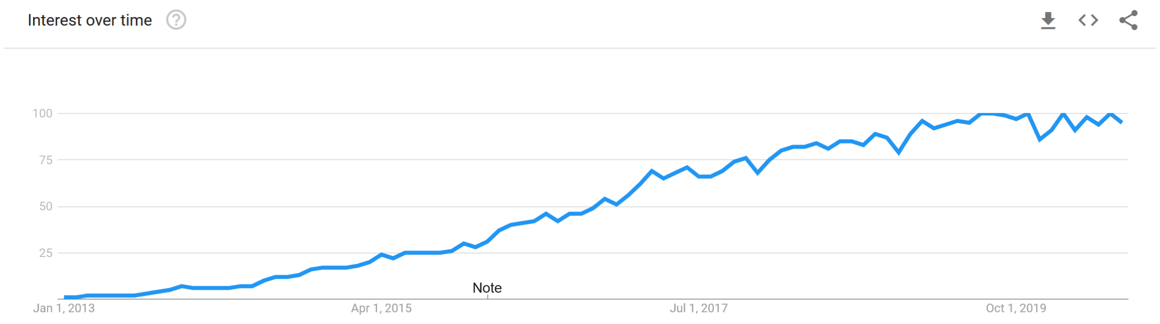


Figure 3.11.1: Docker Search Trend Line 2013 - Present

3.11.5 ROS

The Robot Operating System, or ROS, is a free and open-source software that runs on Unix-based operating systems, primarily Linux. The name is a bit of a misnomer, as it is not actually an operating system. ROS is a program that, while it is running on a computer, facilitates communication between devices and pieces of code. The intention is for developers to be able to write code that works with ROS, and then for other developers or end users to download that code and use it either alone or in conjunction with code they developed

ROS uses a publisher-subscriber model to facilitate this communication. This means that there are several nodes, or "publishers", that push messages to ROS that are available to be viewed by other nodes in data streams called "topics". The nodes viewing a topic are called "subscribers". Subscribers can use the information being streamed in code to extrapolate environmental or other data, and then publish those conclusions to a new topic, which can be used again. These publishers and subscribers can be used together to make a whole robot function.

Typically, all sensors on a robot are publishers, and code either downloaded from the internet or developed by the user subscribes to these sensor topics. Outputs such as motor commands or displayed information are generated by one of these subscribers, typically after a few iterations of processing by a subscriber and publishing again.

The major draw of using ROS is that, because it is already quite popular, it is very easy to find sensors whose developers have already written code to use on ROS. There is a plethora of code available to map based on several kinds of LiDAR, object detection code that is able to work with many types of camera, and navigation code that can be used to both interpret map data and control the driving of a robot. We decided early on that we would use ROS for this reason, and that any sensors we purchased would have to be compatible with existing ROS tools. It is our goal in this project to create a maximally functional robot given our time and situational constraints, and building on the code of other developers is an excellent way to start from a better position and climb higher.

Below is a visual explanation of ROS's publisher-subscriber model.

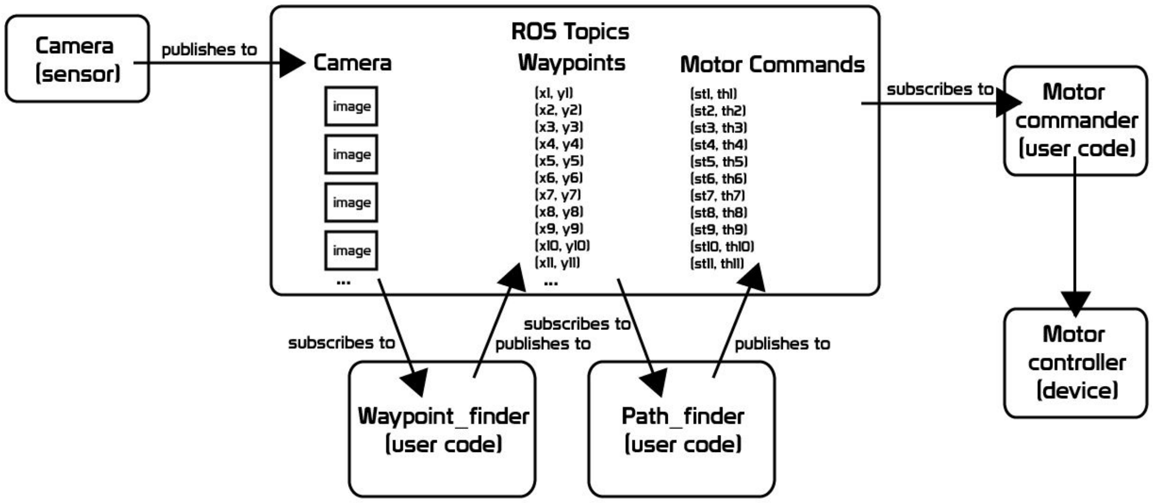


Figure 3.11.2: ROS Publisher-Subscriber Model Explanation

3.11.6 User Interface

The user interface is required in order to operate our robot when not in an automated navigation mode. The most modular, generic, and customizable solution would be to connect to the ROS core application itself using a desktop/mobile application or web interface. A package called rosbridge can provide a WebSocket interface which allows web pages to communicate directly with ROS using the rosbridge protocol. This will be our primary way to communicate with our robot. As far as which front end we will build, we believe the best option is a web interface. A web interface is the most customizable and generic solution. This option is not only easy for all users to access and use but it is also easy to develop for. There are many front end libraries which can make the user interface quick and easy to build while maintaining a professional look. A web interface is also a safe decision when thinking about the future. Almost every consumer device can access a website in some way and web pages will be around for the foreseeable future.

3.11.7 LIGHTTPD WebServer

Since our processing on the Raspberry Pi will be limited and shared among responsibilities it is important to pick a web server that puts an emphasis on performance. LIGHTTPD WebServer is a lightweight web server which has a small memory footprint and intended for high performance use cases. This makes it an ideal solution for hosting a web server on the Raspberry Pi. This web server will host our web page that contains all the components of our user interface to control and monitor the project.

3.11.8 Javascript Libraries

In order to create a frontend for our web interface we will need to use Javascript since we will be creating a dynamic web page that will react to a user input by accessing an external service. In our case the external service will be the ROS websockets. While we could use vanilla Javascript it would require a lot of extra work in order to keep the user interface in sync with the Javascript code. Different frontend frameworks have been created in part to alleviate stress surrounding this issue including but not limited to Angular, React and VueJS. One of the most modular, powerful, and easiest frameworks to use is VueJS. VueJS allows us to create simple or complex web page components which follow separation of concerns and communicate with each other through props. This library will make it easier to create all the components of the user interface while making it possible to make it as user friendly as possible.

3.11.9 Gazebo

Gazebo is a robot simulation software. There exists an official ROS package maintained by gazebosim. It allows you to build various robots and simulate different sensors in conjunction with each other in a robot you can create in the software native model editor. After constructing the model you can write a plugin which can be made to manipulate the model, or insert models into the world, or manipulate the world properties. This creation and manipulation extends beyond just models and onto sensors where you can emulate data from the data sheets into these sensors with varying levels of realism in regards to noise and actuation of the sensor. Gazebo has a fair amount of custom models, encouraging users to share their models, and generate APIs. These models can be made or downloaded and then controlled from ROS with a bit of programming.

3.11.10 Skype URI API

Skype is a desktop and mobile application that allows users to send text messages, conduct voice calls, and conduct video calls. Skype is owned by Microsoft, and a developer platform exists that we can use to create our own applications using the functions of Skype.

LookSee will need to call its emergency contact if an intruder is detected. The Skype developer API can be used to write a piece of software that will call another Skype user. We will install Skype on both LookSee and the phone of the emergency contact.

Later in the project, we switched to Twilio, a similar software that is able to run on a Raspberry Pi or Nvidia Jetson and is easily programmable in Python. A phone number was purchased from Twilio for the robot, and the human user can receive messages over SMS from it.

3.11.11 Jitsi Meet

Jitsi Meet is an open source meeting software that can run easily on any Linux platform. We used it on LookSee as our video call software. Visiting an open meeting's URL is the only thing necessary to join, so it was ideal for our use case.

3.11.12 Secure Shell (SSH)

LookSee's operating system will need a way to be accessed remotely. To accomplish this we will be using SSH or Secure Shell protocol. SSH is included in Linux and MacOS distributions which means we will be able to access it on any Linux distribution that we install on the Raspberry Pi.

3.11.13 Weaved

Weaved is an Internet of Things service that enables users to SSH into remote devices. They allow various web enabled services to turn the regular Raspberry Pi into an Internet of Things device, and effectively allow control a physical device over the internet. In our case this would mean remote access and control over LookSee. It's prepackaged with secure peer to peer connections and device authentications. It can also provide push notifications to your phone which could be useful, perhaps a chime for potential intrusion. Offering two devices (micro controllers) to be controlled for free with up to 300 notifications.

3.11.14 Google Cloud Text-to-Speech

Google provides a Google Cloud Test-to-Speech which converts text into natural sounding speech using an API powered by Google's technology which was based on DeepMind's speech synthesis expertise, and the API touts near human level quality. Google offers 1 million free characters a month and 16$ per million characters after the free quota is reached. Standard, non wavenet voices have a free monthly threshold of 4 million characters, and only 4$ per million characters after that free threshold.

3.11.15 Sublime Text

Sublime is a powerful text editor that can be installed on most Linux distributions. This will be our primary text editor when writing or editing code relating to the project. Sublime out of the box is very lean but there is an extensive amount of plugins and extensions which might be useful during development. This will ensure that we are not bloating our system with unnecessary software and keep our footprint relatively small.

3.11.16 Autodesk Inventor

For all of our CAD work we will be using Autodesk Inventor since we are most familiar with this software and they offer a free student license. Our group member Stavros is certified in this software so any design we intend to create shouldn't be a problem for us. We will get anything we need 3D printed from the UCF senior design lab.

3.11.17 EAGLE

Eagle is an electronic design automation (EDA) software built by Autodesk which allows users to design printed circuit boards (PCB). This software is used in Junior Design here at UCF so the group is familiar with it which is why we picked this EDA as opposed to others. With the strict time requirements we felt it was better to work software that we had some familiarity with rather than learning something new.

3.12 Hardware

In this section we will evaluate different types of hardware for each portion of our project. Each type of hardware was researched and evaluated to figure out the best fit for our project. We will be taking in to consideration what we research on our own as well as previous projects and real world applications that are related to our project.

3.12.1 Batteries

In this section we research and compare the different type of batter cells and choose the one best fit for our project. Our main criteria for our choice is as follows:

- Lightweight
- Rechargeable
- Long Lasting
- Fast Charging
- High Capacity
- Inexpensive

NiCd Nickel Cadmium batteries are one the most robust batteries out there. They are most commonly used in two-way radios, power tools, and emergency medical equipment. However they are very dangerous due to their toxic metal content. Commonly used in robotics as well as small remote controlled cars and other high function electronics.

Advantages and Limitations of NiCd Batteries	
Advantages	Fast and simple to charge
	Over 1000 charge/discharge cycles
	Long shelf life
	Very rugged
	Cheapest battery (cost per cycle)
	Comes in all sizes
Limitaions	Small energy density
	Enivormentally unfriendly
	High self-discharge

Table 9: Advantages and Limitations of NiCd Batteries

NiMH Nickel Metal Hydride originated in the 1970's so the technology behind it is very mature. They are extremely bulky in size and contain high-pressure steel canisters that cost thousands of dollars per cell [13]. Due to the environmental concerns about people disposing of NiCd, NiMH batteries have been replacing those.

Advantages and Limitations of NiMH Batteries	
Advantages	30 - 40 percent more capacity than NiCd
	Simple Storage and Transportation
	Enviromentally friendly
Limitations	Limited service life. 200 - 300 cycles
	Limited discharge current
	High self discharge
	Performance degrades in elevated temperatures
	High maintenance
	About 20 percent more expensive than NiCd

Table 10: Advantages and Limitations of NiMh Batteries

Lead Acid Lead Acid batteries have been around since 1859, they were the first rechargeable battery for commercial use. They are used today in most cars, construction equipment, and large uninterruptible power supply systems. The main downside to these batteries are they are not capable of fast charging as they take around eight to sixteen hours to charge depending on their size.

Advantages and Limitations of Lead Acid Batteries	
Advantages	Inexpensive and simple to manufacture
	Mature and reliable technology
	Low self discharge rate
	Low maintenance requirements
	Capable of high discharge rates
Limitations	Cannot be stored in discharge condition
	Low energy density
	Environmentally unfriendly
	Transportation restrictions on flooded lead acid
	Thermal runaway can occur with improper charging
	About 20 percent more expensive than NiCd

Table 11: Advantages and Limitations of Lead Acid Batteries

Li-ion Lithium ion batteries have been around since 1912, but it wasn't until 1970 that the technology for them was improved and they became rechargeable. "The energy density of the Li-ion is typically twice that of the standard NiCd. Improvements in electrode active materials have the potential of increasing the energy density close to three times that of the NiCd. In addition to high capacity, the load characteristics are reasonably good and behave similarly to the NiCd in terms of discharge characteristics (similar shape of discharge profile, but different voltage). The flat discharge curve offers effective utilization of the stored power in a desirable voltage spectrum. [13]" The most common Lithium ion batteries are the cylindrical cells that we see a lot today in portable chargers.

Advantages and Limitations of Li-ion Batteries	
Advantages	High energy density
	Relatively low self-discharge
	Low Maintenance
Limitations	Requires protection circuit
	Subject to aging, even if not in use
	Moderate discharge current
	Subject to transportation regulations
	Expensive to manufacture
	Not fully mature

Table 12: Advantages and Limitations of Li-ion Batteries

These batteries are so very dangerous when introduced to high temperatures. They are notorious for expanding in size when overheated and even bursting. This makes it impossible to solder wires to the battery terminals, in order to do so you would have to spot weld as it uses far less consistent heat on the battery. If we plan to do this, we will seek professional help, or do it ourselves by taking all the precautions necessary.

LiFePO4 Lithium iron phosphate batteries differ from lithium-ion because they use phosphate as anode material. They are most commonly found in electric bikes, motorcycles, and pure electric vehicles. They are capable of fast charging and offer many charge and discharge cycles.

Advantages and Limitations of LiFePO4 Batteries	
Advantages	Long life cycle
	Safety and stability
	Constant output power
	Environmentally friendly
	Low maintenance
	Quick Charging
Limitations	Lower energy density
	Protection required
	Transportation problems
	Poor performance under low temperature

Table 13: Advantages and Limitations of LiFePO4 Batteries

LiPo Lithium ion Polymer batteries use a dry solid polymer electrolyte. It resembles a plastic-like film that does not conduct electricity but it allows an exchange of ions [13]. "For the present, there is no cost advantage. The major reason for switching to the Li-ion polymer is form factor. It allows wafer-thin geometries, a style that is demanded by the highly competitive mobile phone industry."[13]

Advantages and Limitations of LiPo Batteries	
Advantages	Very low profile
	Flexible form factor
	Light weight
	Improved safety
Limitations	Lower energy density
	Decreased cycle count compared to Li-ion
	Expensive to manufacture

Table 14: Advantages and Limitations of LiPo Batteries

3.12.2 Wheels

There are a large variety of wheels available for robots of the size we are constructing. There are mecanum and omni-wheel drive systems, that allow a robot to strafe in any direction given adequate coding. There are hard plastic wheels that are especially resistant to wear and tear on heavy robots. However, we decided to go with the rubber tires that come standard on most RC cars. At this point, we do not know where and how we will be demonstrating our project at the end of Senior Design. We do not know on what surface LookSee will need to drive and for how long. We cannot risk the robot's functionality being impacted by the terrain. We chose rubber tires because they will provide nearly identical functionality on almost all standard types of flooring, whether indoors or outdoors on a concrete sidewalk.

3.12.3 AC Motors

AC Motors use alternating current to convert electrical energy into mechanical energy. While less frequently used in robotics due to requiring an AC source, they are generally simple in construction, efficient, quiet, and durable. These are generally used for more industrial applications however they do exist in some robotics applications. The two major types of AC motors are induction (asynchronous) motors and synchronous motors.

Induction AC Motors (Asynchronous Motor) Induction motors include two components: a stator winding (the stationary portion of motor constructed using a conducting coil) and a rotor assembly. The stator winding is an electromagnet that produces an rotating electromagnetic field which provides an electric current to the free spinning rotor through the process of electromagnetic induction. This means that the rotor does not have any physical connections which creates a more robust motor. This type of motor can create torque and get going on its own but it will always fall short to synchronize with the rotating magnetic field. This is what allows the motor to keep turning and working but introduces slippage. Without the need for parts like brushes there are no wear parts extending the life expectancy of the part and durability. The speed of these motors is controlled by varying the frequency of the input, this is usually handled by an adjustable frequency drive controller.

Synchronous AC Motors Synchronous AC motors have a greater efficiency than asynchronous motors and can run at a constant speed regardless of the load. A synchronous AC motor is able to synchronize the rotation of the rotor with that of the frequency from the input. In a synchronous motor, the stator is set up the same way however the rotor is instead fed a DC power source creating an electromagnet. This type of motor needs assistance to get started (usually from a smaller motor) but once it gets going the rotor will be locked at the synchronous speed of the input and the smaller motor can be turned off. This means this type of motor creates torque only at synchronous speeds which is why it can run at constant speeds regardless of the applied load.

3.12.4 DC Motors

DC Motors use direct current to turn electrical energy into mechanical energy. DC motors are more common in robotics applications due to the fact generally robots are run off a battery which is a DC source thus reducing complexity of the overall design. DC motors allow for more precise positional control, better speed control, and generate more torque than their AC counterparts.

Brushed DC Motors Brushed motors contain two magnets in the stator facing the same direction which create a powerful magnetic field. The rotor is composed of conductive winding, when power is applied to the windings it creates an electromagnetic field. The electromagnetic field creates a force which turns the rotor. In order to keep the rotor moving each coil on the rotor is attached to the power supply by the commutator, this mechanically provides power to specific coils as it turns, always advancing the rotation. The brushes are the contacts that provide power to the commutator. The direction or rotation can easily be changed by switching the polarity of the input. The speed of rotation is also proportional to the input current. Unfortunately the brushes wear out which decreases the life expectancy of a brushed DC motor. The brushes and commutator can also cause sparking if not properly aligned.

Brushless DC Motors Brushless DC motors do not require brushes which eliminates a lot of the issues caused by DC brushed motors. For brushless DC motors, the rotor is a permanent magnet that can rotate and the stator contains sets of coils. Specific sets of the coils in the stator are supplied power creating an electromagnetic field. The magnetic field in the rotor interacts with the electromagnetic field of the stator and the rotor turns to try and align itself with the stator's electromagnetic field. Hall effect sensors surrounding the stator determine when to activate the coils and which coils to activate.

Conclusion After weighing all the options we decided we would want brushless DC motors to drive our robot solution. We chose this because we want to minimize the wear and tear parts along with the fact that brushless motors improve on brushed dc motors in every way as seen in Table 15 below. Since these motors will be the primary drivers of our robot we believe it to be worth it to spend more money to go brushless.

	Brushed DC Motor	Brushless DC Motor
Maintenance	Brushes and commutator wear	No brushes, no wear parts
Life Expectancy	Brushes shorten life span	No brushes, longer life span
Efficiency	Brushes cause friction which minimize efficiency	No brushes or contacts, better efficiency
Commutation	Mechanical brush commutation	Electronic hall effect sensors commutation
Torque	Brushes cause friction which limit total torque output	No brushes or contacts, better torque output
Speed	Brushes cause friction which limit total speed output	No brushes or contacts, better speed output
Electrical Noise	Brushes cause arcs which create noise	No brushes or contacts, minimal noise

Table 15: DC Motor Comparison

3.12.5 Voltage Regulators

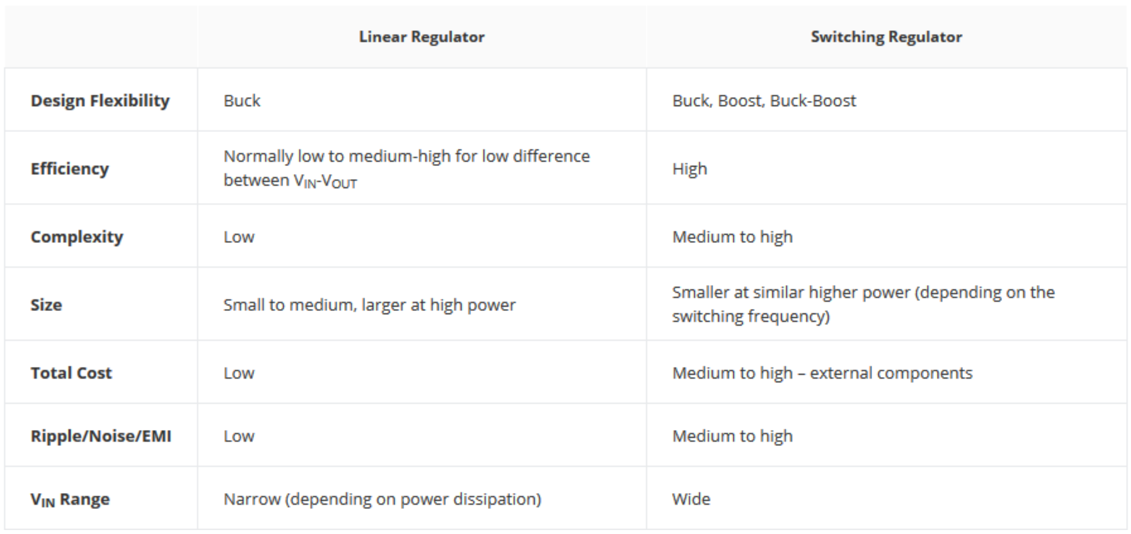
The entire project will be powered off of a battery or a series of batteries. All the components will have their own voltage requirements in order to operate properly. In order to supply power from the batteries to each component a power distribution board will need to be designed to fill the power requirements of each component. The power distribution board will consist of common components that needed to be researched in order to make sure they fit our design. A few types of components were researched based on perceived needs:

- Linear Voltage Regulators
- Switching Voltage Regulators

Linear Voltage Regulators Linear voltage regulators are the most simple voltage regulators however they do tend to be the most inefficient. From its name, linear regulators do not use a switching technique to regulate the output voltage. The input supply voltage must be greater than the output voltage, usually this minimum difference is detailed in the components data sheet as the dropout voltage. The efficiency of a linear voltage regulator is proportional to the difference of input voltage and output voltage. Therefore linear voltage regulators are good choices when the input voltage is close to the output voltage which will increase the efficiency of the circuit. This loss of power dissipates as heat which would need to be factored into the design in order to keep the temperature in an acceptable range.

Switching Voltage Regulators Switching voltage regulators utilize a switching technique to output the required voltage. Unlike linear voltage regulators, switching voltage regulators can not only step down voltage (buck converter) but they can also step up voltage (boost converter) and invert voltage. While these regulators are more complex than linear regulators they offer a much greater efficiency. This is due to the switching technique which ensures that either the elements are switched off or the elements (capacitor and inductor) are conducting fully. This frees us from worrying about the input voltage being close to the output which makes these circuits more flexible and useful. Also switching voltage regulators can output voltages higher than that of the input.

Conclusion Comparing the two types of voltage regulators using the Figure [3.12.1] the simple choice is the switching voltage regulator. While the design may be more complicated the trade offs are worth it as it offers better efficiency and less heat production. They also offer buck, boost, and negative voltage regulation.



	Linear Regulator	Switching Regulator
Design Flexibility	Buck	Buck, Boost, Buck-Boost
Efficiency	Normally low to medium-high for low difference between V_{IN}-V_{OUT}	High
Complexity	Low	Medium to high
Size	Small to medium, larger at high power	Smaller at similar higher power (depending on the switching frequency)
Total Cost	Low	Medium to high – external components
Ripple/Noise/EMI	Low	Medium to high
V_{IN} Range	Narrow (depending on power dissipation)	Wide

Figure 3.12.1: Voltage Regulator Comparison
(awaiting reprint permission from Renesas)

3.13 Movement

Our project will need a robust movement system that allows the movement through a variety of spaces and scenarios. The environment in which the robot will operate could change each time it moves through the space. Knowing that our robot's primary movement will be done using a drive system, we evaluated the following different types of robotic drive systems:

- Turn Steer
- Rack-and-Pinion Steering
- Differential Drive
- Holonomic Drive
- Tricycle Drive

3.13.1 Turn Steer

The turn steer system is the easiest to understand all vehicles on the road use this type of drive system. There are four wheels total, each parallel pair is mounted on a common axis and either one pair or both pairs are powered. The front pair of wheels are steerable, meaning they are able to pivot around 45 degrees with each wheel in the pair matching the other. The wheels would be pointed in the direction the robot needed to move. This system is versatile because it can be implemented in various configurations, however you lose some controllable degrees of freedom. This system works well when the robot would not need to strafe or rotate on the spot. Instead this system is ideally suited to make sweeping smooth turns. If an obstacle is directly in front of the robot it would require the robot to back up first before trying to maneuver around it. This type of drive system is the most intuitive for user controlled systems as opposed to the other methods that may require more training to become comfortable with.

3.13.2 Rack-and-Pinion Steering

The most common steering method we see today is the rack-and-pinion steering. It is used in almost all today's cars and SUVs. This method of steering is very similar to turn steering as it works by the using the steering wheel to rotate a rod attached to a pinion that locks it's teeth on the pinion case (rack) transforming big rotations from the steering wheel into small, accurate turns of the wheels, giving a solid and direct feel to the steering [14].

3.13.3 Differential Drive

A differential drive, similar to a differential in most cars, allows two wheels to turn at different speeds. The differential drive utilizes two drive wheels which are connected by a common axis. Each of the two wheels can be powered independently and in either direction. Usually the axle is positioned in the center of the robot with a third non-powered free turning wheel for stability. When both wheels are powered in the same direction and with the same speed, it will drive in a straight line in the direction that the wheels are rotating. This drive system also allows different types of turning which greatly increases its maneuverability. If one wheel stays stationary while the other wheel spins in a direction, the robot will rotate with the center of rotation located directly at the stopped wheel. If each wheel turns at the same speed but in opposite directions, then the robot will rotate with the center of rotation being the center of the common axis, usually the center of the robot as well. Turning similar to a car can be achieved by having both wheels spin in a common direction but at different rates. This will make the robot's center of rotation be defined by the spacing between the wheels and the velocity at which each wheel is turning. The different methods of turning that can be achieved using differential drive can be seen below in Figure [7.2.2] While this method of turning is less agile, it is useful in some applications when needing to make turns of a certain radius.

This method allows for greater control of the robot. If an obstruction is directly in front of the robot, you would be able to rotate 90 degrees in order to move around the obstacle without needing to back up. This method is also low cost since it would only need two wheels and motors as opposed to the four required for other methods. This also reduces complexity and overall weight of the design.

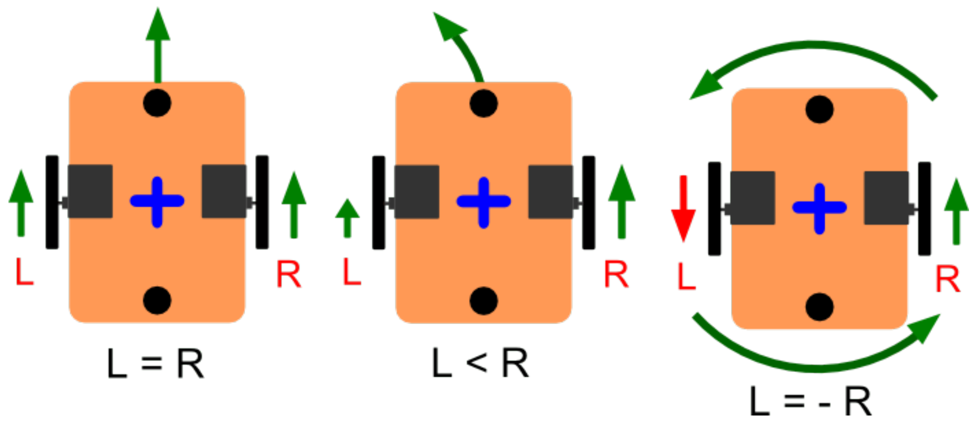


Figure 3.13.1: Differential Drive
(Reprinted with permission from Nick Berry)

3.13.4 Holonomic Drive

A holonomic drive system is defined by the controllable degrees of freedom equaling the total degrees of freedom. This drive system can be realized in different ways, primarily defined by the type of wheels used. Either wheels on controllable castors or omni-directional wheels can be used in order to achieve all controllable degrees of freedom. This allows for a new maneuver, strafing. The robot would be able to slide in any direction while facing in any direction. When using controllable castors as seen in Figure [3.13.2], each wheel would be able to change its direction with 360 degrees of freedom. This method would be extremely complex and require each wheel to have a motor to power the movement of the wheel as well as the rotation. The orientation of each wheel would need to be considered to ensure that all wheels are aligned based on the movement you would like to achieve. This adds hardware and programmatic complexity with the trade-off being unparalleled movement capabilities.

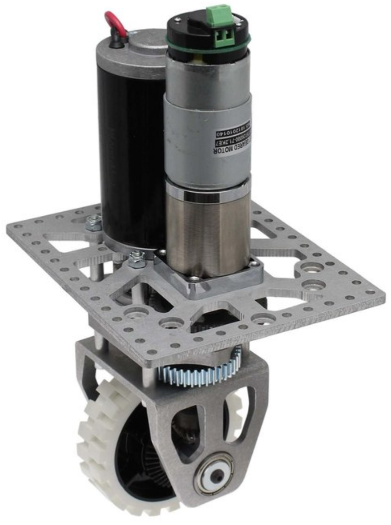


Figure 3.13.2: Controllable Caster Wheel
(Reprinted with permission from Andy Mark)

To decrease the complexity while still maintaining all degrees of freedom designs that utilize fixed wheels were considered. These types of design use specially designed wheels in order to achieve the same level of freedom. Omni-directional wheels as seen in Figure [3.13.3] have free moving rollers attached around the circumference of the wheel that rotate perpendicularly to the direction of the wheel. They can be used in a variety of configurations to achieve the desired freedom of movement such as a 3 wheel layout (Kiwi Drive), a 4 wheel layout, and an H-Drive. All these different layouts can be seen in the diagram below. Each has their strengths and weaknesses. Some of their common takeaways are that the ride is rough due to the different spaces between the rollers. Also inclines can be challenging because of the free rolling rollers.

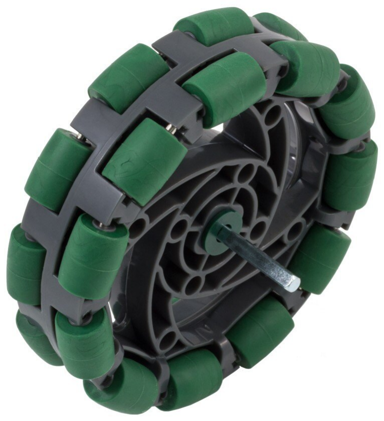


Figure 3.13.3: VEX OMNI-Wheel
(Permission Pending)

Mecanum wheels can be used to achieve the same level of freedom. Like omni-directional wheels, these wheels have free rolling rollers around the edges of the wheel however the rollers are longer and mounted at a 45 degree angle to the main axis of the wheel as seen below in Figure [3.13.4]. These wheels are mounted in a normal 4 wheel configuration at each corner of the robot with the rollers of each wheel pointing toward the center of the robot. This configuration allows for all degrees of movement based on the direction each wheel is turning. See the diagram below for a visual of how different types of movement can be achieved.

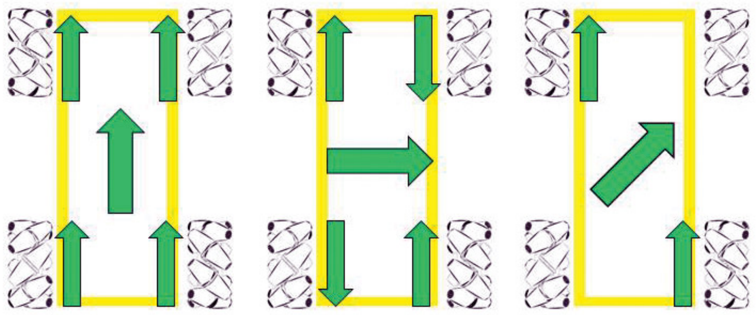


Figure 3.13.4: Directions of a Mecanum Wheel Robot
(Permission Pending)

3.13.5 Tricycle Drive

Tricycle Drive is a three wheeled robot design that consists of a single wheel in the front and two wheel in the back connected to same axle as seen in Figure [3.13.5]. This design is very similar to how planes drive when taxiing. We all used this design in our into to engineering class when we built a Boe-Bot Robot [15]. Their main disadvantage is that they can not spin like a differential drive robot due to their limited radius of turn [16].

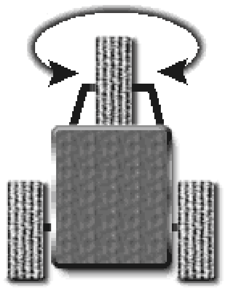


Figure 3.13.5: Tricycle Drive Robot
(Permission Pending)

3.14 Suspension Systems

A suspension system has two goals, to maximize the amount of contact the drive system has with the terrain and to minimize the jarring effects of an uneven terrain to the cargo. The terrain our project will move around in is unknown, therefore we have to be adaptable and adjust to a wide range of scenarios. While the drive system is important, the suspension system that will be used is equally important. Many of the drive systems can be paired with different suspension systems easily while others cannot. Some of the suspension systems we looked at are:

- Rigid Suspension
- Independant Suspension
- Articulated/Split-Body Suspension
- Rocker-Bogie Suspension

3.14.1 Rigid Suspension

Rigid suspension systems basically mean there is no suspension. The entire robot is a rigid body, this limits the type of terrain it can traverse. An example of a rigid suspension chassis can be seen below in Figure [3.14.1]. This system does nothing to increase the amount of contact the wheels have with the terrain nor increase ride quality. When a rigid suspension system encounters rough terrain the only way it could keep maximum contact with the terrain would be to flex the chassis. This design puts added stress on the chassis. However this is still a very common design approach as many robotic applications are limited to indoors or smooth surfaces. For our design we won't consider it due to its lack of adaptability over various terrain.

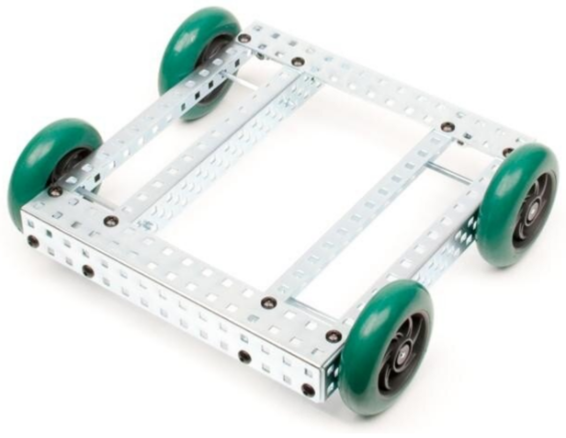


Figure 3.14.1: Rigid Suspension
(Reprint with permission from VEX Robotics)

3.14.2 Independent Suspension

Independent suspension systems are the most common in suspension systems in automobiles. Each wheel can react to changes in the terrain independently. This is used in automobiles because it blends the needs of increased handling and ride quality. This suspension system is usually applied to a vehicle with four wheels and can be modified to allow the vehicle to drive over small to medium types of obstacles relative to the tire size. The size of an obstacle a vehicle with this suspension system can climb is determined by how much traction the back wheels have and the height of the center of gravity. There is a greater tipping risk with this system when going over large obstacles since the loaded parts of suspension give way which throws off the height of the center of gravity. This system comes with increased complexity and design in order to ensure the vehicle is stable. This increased complexity also adds weight to the overall design and would require maintenance. There are many different implementations of independent suspension systems, we won't get into the details of each type because they all relatively accomplish the same thing. This system is commonly used on RC cars as seen below in Figure [3.14.2], giving RC cars a good amount of stability and ride quality.

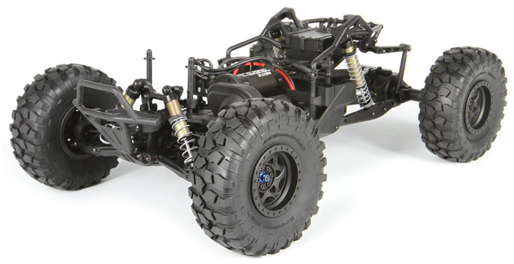


Figure 3.14.2: Independent Suspension on an RC Car
(Reprint with permission-pending from AxialRacing)

3.14.3 Articulated/Split-Body Suspension

Split-Body Suspension is similar to rigid suspension with fixed wheels on each side, but in order to try and maintain maximum contact with the terrain, the chassis of the vehicle is split into different segments. The individual pieces of the chassis can pivot in order to change the wheel position to maintain maximum contact with the terrain. These types of suspension systems were designed to allow the vehicle to overcome obstacles up to 50 percent larger than the wheel diameter but failed in testing. Similar to the rigid suspension, the maximum amount of ground clearance is only as big as the wheels. This suspension system is a simple concept but would add great complexity when designing the chassis of our robot. Each segment would need to be connected together but still be able to pivot in specific directions while not affecting any of the internals.

3.14.4 Rocker-Bogie Suspension

The rocker-bogie suspension system was developed by NASA in 1988 by Donald Bickler and has been used in all of NASA's rovers. A render of this Mars Rover using this suspension system can be seen below in Figure [3.14.3]. This system was chosen due to its ability to maximize the amount of contact with terrain giving it increased handling and its ability to climb obstacles 50 percent greater than the diameter of its wheel. The rocker-bogie suspension system is applied to six wheel vehicles, these extra wheels add traction which aid in it being able to climb such large obstacles. The bogie portion of the suspension includes the front two wheels on each side which are connected by a pivot in the middle. The rocker portion is the back two wheels on each side. These are connected to the two front wheels at the same middle pivot point. Each side is connected to each other through a differential bar which transfers weight from one side to the other in order to keep it level and gain traction. In addition to the climbing ability, this configuration allows the chassis to be the most stable for sensors as its position is an average of the angle of the side arms. While this design is complex, I would argue it is less complex than the independent suspension system since it is just a series of linkages and pivots, no complicated spring dynamics that have to be applied to each wheel individually.

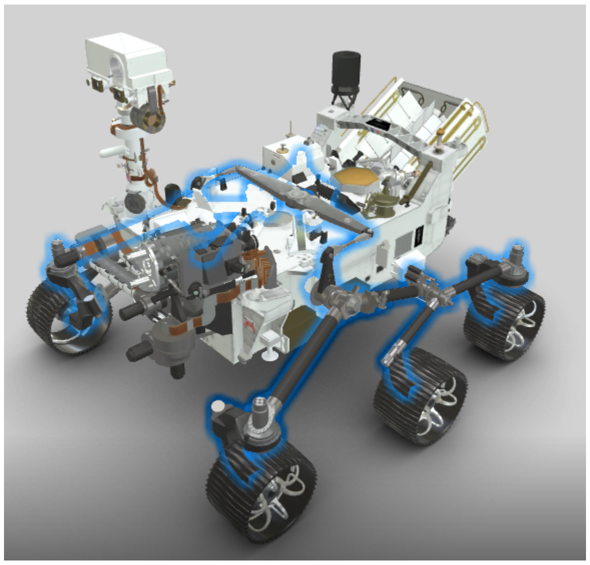


Figure 3.14.3: Mars Rover with Rocker-Bogie Suspension
(Reprint based on NASA Media Guidelines)

3.15 Chassis

In order to not "reinvent the wheel" we have decided to use a ready made chassis as a base for our robotic solution. There are several different types of robot chassis that could work that have the features we are looking for. Below we look at different types of chassis, their features, and how we could use them.

3.15.1 Rover Chassis

After weighing all the different options we believed a chassis similar to that of the Mars rover utilizing the rocker-bogie suspension system would be the best technical way to solve our issue. The suspension system would mean our movement would not be impeded by small obstacles and would even be able to traverse stairs. This special type of suspension was specifically built so that the main body of the chassis stays pretty level since it would only be operated using the on board camera. We believed this would be ideal in our case since we need smooth viewing for our camera as well. On the Mars rover the drive system can do both turn steer using the front wheels and can turn on the spot by pivoting the outside wheels. This means we would have a great degree of freedom when it comes to movement. However after exploring this option further there are a few issues that prevented us from using this type of chassis.

Availability After looking for a ready made chassis with these features we realized nothing like this currently exists. We then turned to making our own, there are several examples and open-source designs for building something similar. These designs included 3d printing many of the connections for the drive systems and enclosure. While this would be fine for a prototype we still thought the overall structural integrity of the project would suffer. Also none of the members of the team have access to a 3D printer and with COVID-19 we don't have regular access to the 3D printers on campus. We took a step back realizing this line of thinking was spiraling out of control as we are not mechanical engineers and we did not want to spend the bulk of our time designing a chassis leaving us no time to actually complete the project.

Complexity This chassis would by far be the most complicated but there are good trade offs making it something we wanted to pursue. While we would be unable to find another chassis design that would allow the same type of obstacle traversal we concluded that we could get most of the features from a different type of chassis with less complexity.

Cost The cost of pursuing this type of chassis design would be much larger than any of the chassis options because we would need to create this specialized design from scratch. This would inevitably lead to mistakes and revisions driving up the material cost. We have a limited budget as this project is not sponsored and we needed to keep costs low.

Ubiquity While there is a niche community of rover enthusiasts, this type of design is not common among robotics projects. We believed that if we encountered any issues in the design or build of this chassis we would not have much support to fall back on. Since this is not a main requirement for our senior design course we think our time would be much better spent focusing on the hardware and software interface and the custom PCB.

3.15.2 iRobot Create

The iRobotCreate chassis is built off of a refurbished iRobot Roomba which employs a differential drive system. It also comes with a rechargeable base station, something we would love to add to our project in our stretch goals. The differential drive system also gives us plenty of freedom when it comes to movement. This type of system is proven to work and is in many homes all over the world. Below we explore this chassis option further.

Availability This type of chassis is readily available on iRobot's website however shipping has been slowed due to COVID-19. There is nowhere we could get this type of chassis locally without making our own by gutting an existing Roomba which would come at an increased cost.

Simplicity This chassis is a simple design however it seems a little too simple. This chassis offers no clearance and would not be able to scale any type of obstacle in its way. Some of the team already own an iRobot Roomba and can attest to it frequently getting stuck on normal household items. We would like our solution to not be easily defeated by a stray piece of clothing or a steep incline.

Documentation This platform has very little documentation which would not normally be a huge issue for a robot chassis however we need to interface with all the electronics that already exist inside. This means instead of having control over the entire electronics system we would delegate to an under documented system that would be difficult to troubleshoot. This is not an open-source project and there is not a large community surrounding the product meaning there has not been much experimentation with this platform either.

3.15.3 RC Car

The chassis we have currently selected is a 1:16 scale 24.G RC Monster Truck. This would include all the previously decided hardware like wheels, DC motors, turn steer, and an independent suspension system. We selected this chassis because it seems like a good fit, though documentation is sparse and we don't desire to reinvent the wheel in terms of accessibility to the motors or integration with ROS. Other options are currently including rover platforms, and other RC cars.

We understand that it may be unconventional to select an RC car as our chassis. We have come to this decision through careful consideration, and for the following reasons.

Availability In the current global situation, it is difficult to obtain many specialized parts that are commonly used on robots. Between reduced availability of shipments from certain countries, and heightened demand of hobby and maker materials due to engineering professionals being asked to quarantine at home, we knew that our first priority for chassis selection had to be the ability to actually obtain the product. RC cars of many sizes, configuration, and levels of quality are available on a variety of websites, and even in most toy stores. Orlando has multiple RC hobbyist shops that can support any additional components we may need. No matter what happens going forward, we will always be able to purchase another RC car or any components related to it if necessary from one of multiple sources.

Cost Due to the high number of RC cars that are sold for adult enthusiasts, children, and robot hobbyists, RC cars exist in nearly every combination of size, features, and cost. LookSee is intended to be a low-budget product, but that is also related to the fact that our team has no sponsors and we are all paying for the project together out of pocket. It is important to us that we can purchase the car for as little money as possible, but also, it is important to us that we have the budgetary freedom to purchase additional chassis or components if necessary. If the vehicle were to become damaged at some point during the project, we can replace an RC car for less than $50. The cost of a professional robot kit of similar size typically starts in the hundreds of dollars.

Simplicity There are a great many impressive features a robot can have. Mecanum wheels, grasping attachments, and storage containers are all products a person can buy to help them build a robot. However, we know that building a robot for Senior Design is a daunting task, and one that most teams are eventually unable to complete to satisfy their original requirements. In order to keep our project as simple and therefore doable as possible, we wanted to have a chassis that comes with absolutely nothing other than the ability to drive and steer. An RC car, out of the box, contains all of the mechanical and electrical components necessary to be driven. Anything additional to these functions is not only superfluous, but potentially a drain to our time and effort. Our first priority is the successful completion of LookSee and the Senior Design program.

Ubiquity Because of the reasons outlined above, RC cars are a very popular base for hobbyist projects. The model car racing community has extensive documentation about each component of higher-tier RC cars, and there are thousands of Instructables, Wiki How guides, and other internet tutorials that show how people have used RC cars to create battle robots, autonomous vehicles, and school projects. During the course of this project, we may receive very little support in the form of expert advice and help, but the fact that there are enough resources available to us at home to properly understand and modify the drive system, the signal receiver, and the power system is extremely valuable.

4 Related Standards

This section explores the various types of standards that could relate to our project. We explore ABET design standards, technical design standards, and the impact our design might have on each of these sections.

4.1 ABET Design Requirements

As UCF is an ABET-accredited school, our senior design project will reflect the standards outlined by ABET for demonstrating proficiency in our chosen field of Computer Engineering. Below are a series of explanations about how we are striving to meet all standards and take into consideration the ABET design constraints in the design and execution of our project.

4.1.1 Design Experience

The senior design program is intended for students to demonstrate proficiency in the skills and courses covered in the course of our degree, as well as our ability to solve real world problems with realistic constraints in a reasonable time frame.

The skills that we demonstrated in the completion of this project are:

- Project management and design
- Product design to suit a customer use case
- Robot design
- Computer vision
- LiDAR environment data processing
- Autonomous navigation programming
- User interface design and implementation
- ROS proficiency and use
- Hardware purchase selection
- PCB design
- Integration of modular design

4.1.2 Economic

Obviously in a time like this with the current climate of disease, and Florida's seemingly tepid response to COVID-19. Economic certainty is questionable, and obviously without the ability to foresee the future it doesn't seem wise to build something frivolous. Our project presents aspects of design we we're interested in and desired more experience working with the subsystems indicated on our Team Responsibility Block Diagram (Figure 8.4.1). With this in mind we are students working within the confines in terms of environment we've set a reasonable budget for our project and desire to not break the bank.

LookSee whilst being a practical product, in all likelihood isn't designed in respect to mass production or to make a profit but rather as more pure intellectual pursuit. Thus we feel whilst maintaining a respectable, yet functional budget. The idea of minimizing cost to maximize profit or whom we'll be licensing our technology to aren't exactly a high concern pressing on our collective minds in the current sociopolitical climate.

4.1.3 Environmental

LookSee's environmental impact is minimal as it's primarily an indoor functioning robot using a battery and power provided by it's domicile. Though the creation of technology usually does involve the harnessing of resources, in contrast to the burning of fossil fuels; LookSee's impact isn't nearly all that significant provided after the power source has expired (and may need to be recycled, ideally) life cycle of the device may be extended by ordering another battery.

4.1.4 Social

With the substantial rise in cases, shown as of today via google. Florida is one of a handful of states within the United States that continues to have an increasing, significant, new number of cases with this past Sunday (July 12th) having a new record high 15,300 new cases in Florida. That number isn't to be confused with cumulative cases, but rather new cases! There seems to be a significant portion of the population within Florida that desires to not wear a mask, or observe relatively new precautions of staying a marked 6ft of distance between you and the nearest person. Guidelines given to the general populous about wearing masks, and staying away from gatherings of large people are generally ignored and as a result of this careless behavior, the global pandemic seems to exploding a life of it's own in our state. Whilst smaller island countries (like Australia, and New Zealand) with strict regulations enforcing new standards for health (everyone must wear a mask when in public, observe social distancing, do not have large gatherings) seem to be recovering and getting the number of new cases into a recession and eventually reducing the number of contagious peoples within their country to zero. In contrast, Florida seems to have the opposite idea, with theme parks like Disney reopening and the general inconsistency in which people as a whole here seem to observe guidelines for disease control has quite the opposite effect occurring as you can see in the figure below.

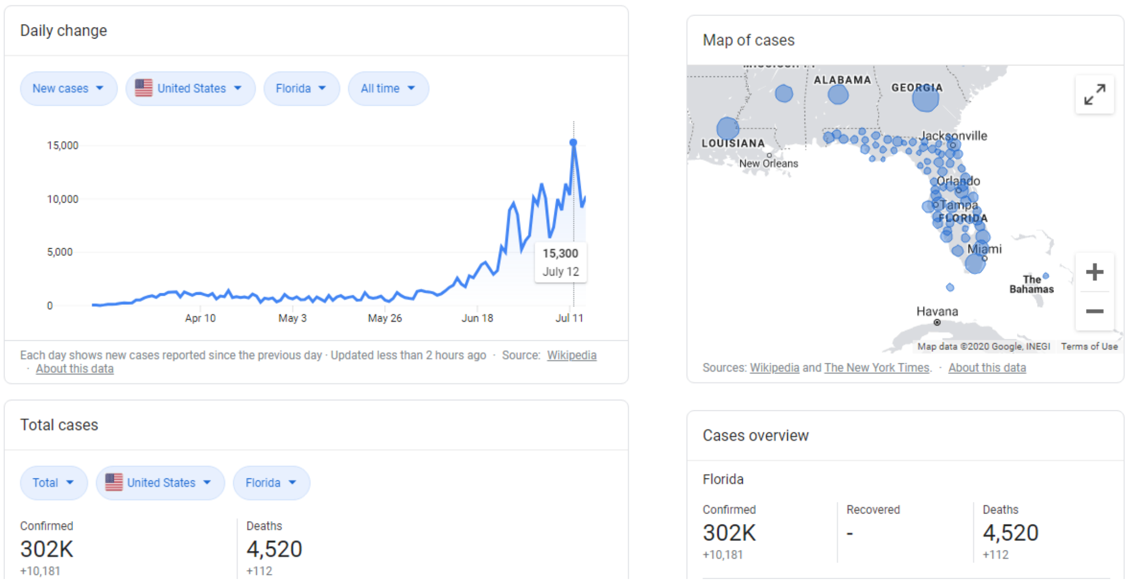


Figure 4.1.1: Google's COVID-19 Search Statistics Tab

Although much of Orlando's tourism industry is reopening, many other businesses remain closed and vulnerable for indefinite lengths of time as the mandate for distancing and new criteria for disease control have restricted industry. Employees, as a result are without work and the ability to earn a living wage the outlook remains grim though definitely survivable by all means. Additionally, the Black Lives Matter protests have, unfortunately, made businesses in urban areas vulnerable to opportunistic looters and troublemakers in a time when business owners cannot be present.

As businesses remain closed for weeks or longer while the employees are working remotely or have left the area entirely, there is a demand for peace of mind. LookSee's ability to watch and stream video without needing a human present means people do not have to spend time out in public to protect their property. Additionally, if people evacuate their homes to stay with family in less populated areas, there is also a demand for peace of mind about the security of the homes they have left unattended.

Robotics as a field is seen with much suspicion by people of all levels of education. Some people are afraid of where robotics will develop from where we are now, and some people are afraid that robotics will cause many industries to lose a "human touch" as fewer and fewer humans are involved. For that reason, the "Three D's" of robotics dictate which jobs should be targeted for improvement with the assistance of robotics - those which are Dull, Dirty, and/or Dangerous. Night security is a job that can fall into categories of being Dull and being Dangerous, which makes it a prime candidate for replacement by robots.

However, LookSee will not be endangering anyone's jobs. The target market for LookSee is businesses who cannot afford to pay for a full- or part-time employee to guard the premises, and want to compromise with the one-time purchase of a robot.

While some people will undoubtedly be uncomfortable with the idea of an internet-connected robot streaming video data of other people, we believe that because this will be up to the business owner and not affect the privacy of anyone else other than possible intruders, the level of public concern will probably roughly equal the level of concern that exists for the Roomba line of vacuum cleaners.

4.1.5 Political

It is not legal to record employees in the workplace. It would be up to the employer to ensure that their LookSee device did not violate any laws in its use. However, its presence in the workplace could cause concern and controversy for employees who are already suspicious of or dissatisfied with the people they work for.

Ultimately, it is likely that a large number of robots of similar cost and level of functionality will be going to market soon, and the popular opinion of robotics will come to the forefront of the sociopolitical conversation. Guidelines will be put in place that will govern how robots can and cannot identify and track people.

For that reason, we have chosen to make LookSee very simple. LookSee only stores the information of intruders during certain hours of the day, and makes no attempt to recognize or categorize anyone. While it may make some people uncomfortable, we are confident that it is not a violation of anyone's rights if used as intended.

4.1.6 Ethical

LookSee stands on the right side of ethical surveillance. For protecting your property and assuring everything is in order. Following the old adage 'If you haven't done anything wrong, you having nothing to fear.' Provided that someone is a law abiding citizen, they shouldn't have any issues entering a store being recorded as it's fairly standard practice in most brick and motor shops to have some form of sign with a recording device informing the participants of the shop that they are being recorded. Similarly LookSee could fall under this banner of recordings.

Though LookSee would in the best of case not encounter people on a regular basis after hours whom aren't designated to be in the use case. This leads that LookSee would generally only be surveying an area at night, and likely to not encounter the general populace and infringe on their privacy in any way.

4.1.7 Health and Safety

LookSee's ability to call the operator providing real time data for the operators decision where it's necessary authorities who then can decide whether that be the police, or an ambulance for a staff member who has become incapacitated and is alone without aid. LookSee promotes the safety of the crew whom may be (opening/closing) the store with fewer numbers than a standard daytime shift, by being their additional crew mate and potentially alerting the operator who can then make the most appropriate call for the given situation they've become alerted to. In regards to safety LookSee removes the risk when confronting a trespasser by being your virtual guard dog, shielding you (the operator) from hard and letting you rest easy as LookSee has the necessary intelligence to be alright without direct supervision.

4.1.8 Manufacturability

Given foremost that LookSee was designed for in respect to intellectual pursuits long term manufacturability was never in the forefront of our minds. Instead we chose technologies in expanding fields and industries to further grow out our skill set as engineers in the area of expertise of our choice. Manufacturability was never in the forefront of our design decisions. However in the event that LookSee is seen and desired to be purchased to the technology to be licensed to some interested party.

We believe, provided that the interested party is willing to pay for the cost of equipment and skilled workers, that LookSee could be created in mass and definitely with some margin of discount provided they order the parts *en masse* or perhaps do a little research and development with some mechanical engineers or a most apt chassis they could essentially break apart LookSee into it's parts and form some form of skilled assembly line working and start production.

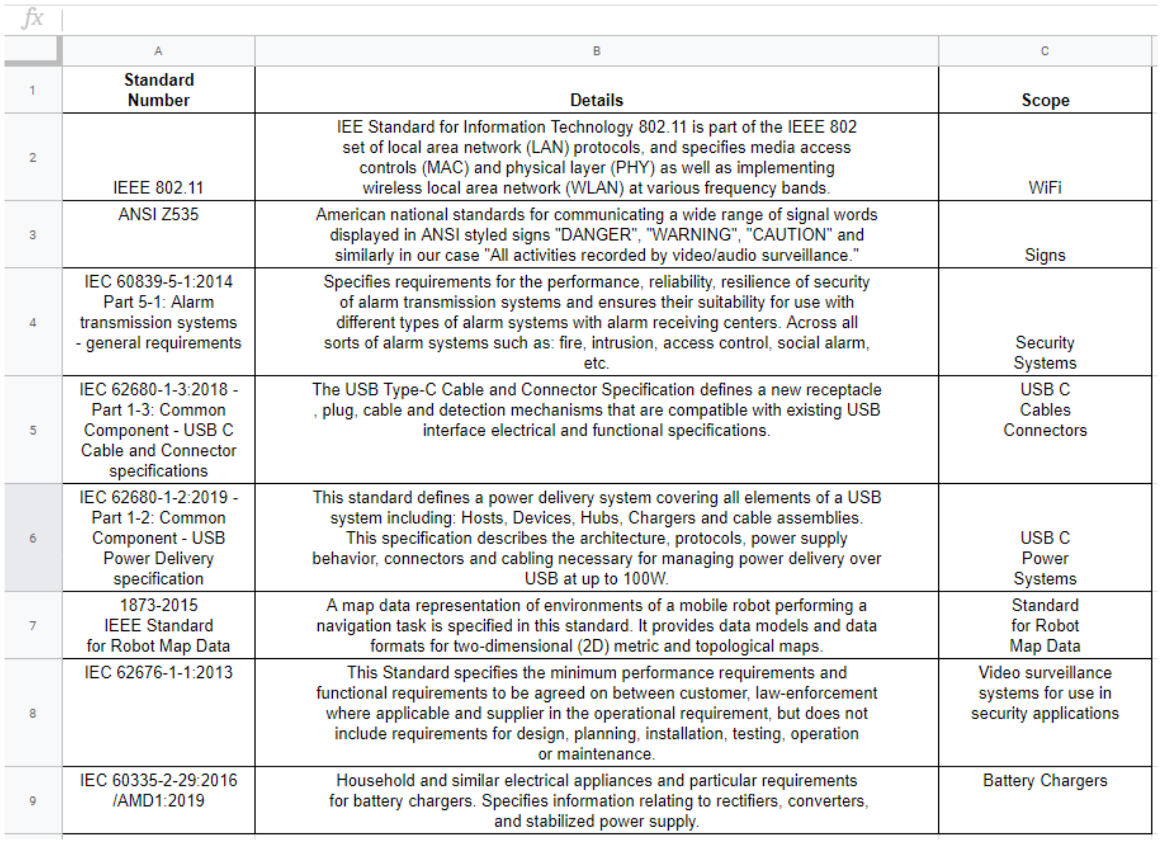
4.1.9 Sustainability

Provided that the products used to create LookSee remain available (or some variation of parts exists) to create LookSee and the technology loaned through open source libraries remain open source, then for the foreseeable future LookSee will stand the test of time. As potential improvements in technology used in operation of LookSee may improve; so shall the behavior of the robot. With all the new innovation in computer vision and autonomous driving on the forefront of technological advancements.

4.2 Design Impact

As our project will have nothing but the PCB manufactured, and effectively these standards will not have an enormous impact on our project as we will not be manufacturing hardware and generally using established open source libraries. We will be using relatively standard electrical components which generally abide by the standards listed below in figure 4.3.1, though our PCB will control power distribution and motor controls and our device will hold a battery which needs to abide by a particular charging standard. We included details about both surveillance and the display or alert (via sign) of surveillance as precautionary measure even if it isn't a direct impact on our project.

4.3 Standards Search



Standard Number	Details	Scope
IEEE 802.11	IEE Standard for Information Technology 802.11 is part of the IEEE 802 set of local area network (LAN) protocols, and specifies media access controls (MAC) and physical layer (PHY) as well as implementing wireless local area network (WLAN) at various frequency bands.	WiFi
ANSI Z535	American national standards for communicating a wide range of signal words displayed in ANSI styled signs "DANGER", "WARNING", "CAUTION" and similarly in our case "All activities recorded by video/audio surveillance."	Signs
IEC 60839-5-1:2014 Part 5-1: Alarm transmission systems - general requirements	Specifies requirements for the performance, reliability, resilience of security of alarm transmission systems and ensures their suitability for use with different types of alarm systems with alarm receiving centers. Across all sorts of alarm systems such as: fire, intrusion, access control, social alarm, etc.	Security Systems
IEC 62680-1-3:2018 - Part 1-3: Common Component - USB C Cable and Connector specifications	The USB Type-C Cable and Connector Specification defines a new receptacle , plug, cable and detection mechanisms that are compatible with existing USB interface electrical and functional specifications.	USB C Cables Connectors
IEC 62680-1-2:2019 - Part 1-2: Common Component - USB Power Delivery specification	This standard defines a power delivery system covering all elements of a USB system including: Hosts, Devices, Hubs, Chargers and cable assemblies. This specification describes the architecture, protocols, power supply behavior, connectors and cabling necessary for managing power delivery over USB at up to 100W.	USB C Power Systems
1873-2015 IEEE Standard for Robot Map Data	A map data representation of environments of a mobile robot performing a navigation task is specified in this standard. It provides data models and data formats for two-dimensional (2D) metric and topological maps.	Standard for Robot Map Data
IEC 62676-1-1:2013	This Standard specifies the minimum performance requirements and functional requirements to be agreed on between customer, law-enforcement where applicable and supplier in the operational requirement, but does not include requirements for design, planning, installation, testing, operation or maintenance.	Video surveillance systems for use in security applications
IEC 60335-2-29:2016 /AMD1:2019	Household and similar electrical appliances and particular requirements for battery chargers. Specifies information relating to rectifiers, converters, and stabilized power supply.	Battery Chargers

Figure 4.3.1: Related Standards

5 Design Constraints

Listed are the constraints we will face during this project, as well as constraints we anticipate will accrue. We will and are currently running into product availability issues constraining us to a lot fewer options that we hoped for due to the current pandemic situation effecting the price of electronics.

5.1 COVID-19

By far the biggest constraint for this project is the Corona Virus 2019 Pandemic. Due to this we have had to change a lot of our planning as well as future plans. Senior Design is a only offered on campus and due to the virus all students were moved to remote instruction making the class structure very hard and unorganized. We were unable to meet with sponsors as the class in tented so we have to be very conservative with our budget.

Another challenge we are facing is ordering parts that originate from China, such as PCB and circuit. Also, it is difficult not being able to meet with the class to discuss our projects, instead we are all quarantined and communicate to our groups only though online chat.

This constraint will eventually lead to more difficulties for when we start Senior Design II. Senior Design II for the Fall of 2020 has been moved to online lecture only. This may include on campus labs will be closed as well so we have to design our project around the fact we cant use the tools and equipment the machine shop and electronics lab have.

Meeting with our group to build our project is going to be near impossible as most of us cannot afford to get sick as we have families with health conditions. As of now, we are not sure how everything will be in the next couple months, but presenting our projects might not be possible as the virus is still spreading more than it has when we first canceled classes.

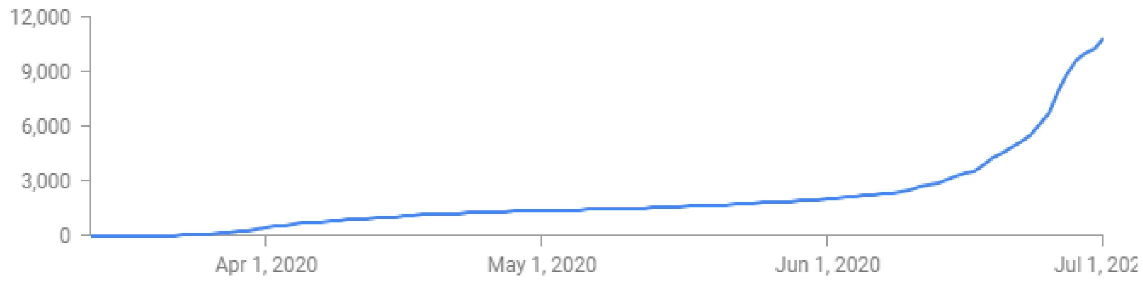


Figure 5.1.1: Cases over time of COVID in Orange County Florida

5.2 Cost

Due to the nature of our design's concept as a low-cost alternative to an extensive security system, it is important that our budget for the total hardware of the project not exceed $1000. We are aware of the possibility that we may exceed our budget due to products failing, change in design, and overpaying for items as electronics have gone up in price due to the COVID-19 outbreak.

5.3 Time

Time is one of our biggest constraints as we enrolled for the summer term meaning instead of 16 weeks we have 12 weeks to get all of our research and planning completed, as well as finish out entire documentation. With any extra time we have we will begin prototyping our robot and purchasing all necessary components.

We are also aware of the possibility that one or more of us can get sick from this pandemic putting us out for two or more weeks. We are making sure that everyone is aware what everyone is doing and their progress, so that if the unfortunate event accrues that one or more of us gets sick, the other group members know where to pick up from.

5.4 Size

Since our entire robot is built off a RC car we are constrained to the dimensions of the car being 48cm long, 27cm wide, and 25cm tall. However, the height of the final product will change as we start to install all of our electronics.

We do plan to remove the pre-installed cover on the RC car and fabricate a base plate to sit on top that will allow us to mount all our hardware. We will not be extending the length or width of our car as in the event there is a collusion our RC car's wheel will hit the object first, rather than our base plate that has all of our electronics and hardware.

5.5 Power Consumption

The RC car is rated for 20 to 30 minutes of driving time, once we begin prototyping and testing we will use higher density batteries and possible solder multiple in parallel to achieve a higher capacity.

All of our electronics will run off a external battery with a 30000mAh capacity rated for 11Wh. The output from this battery is 5V DC at 3A. Which meets the power requirement to power on our processor and sensors.

5.6 Environment

Our design is targeted for large non crowded area such as warehouses and empty convention centers. The environment must also be in a room with a strong network connection as it is needed for communication with the user. For our robot to operate as intended the area of surveillance must meet the following requirements:

- Must be indoors
- Must be even terrain (no stairs)
- Must be well lit
- No moving machinery or equipment
- No personnel
- No narrow hallways or walkways
- No rooms with pools, or other water features build into the floor level

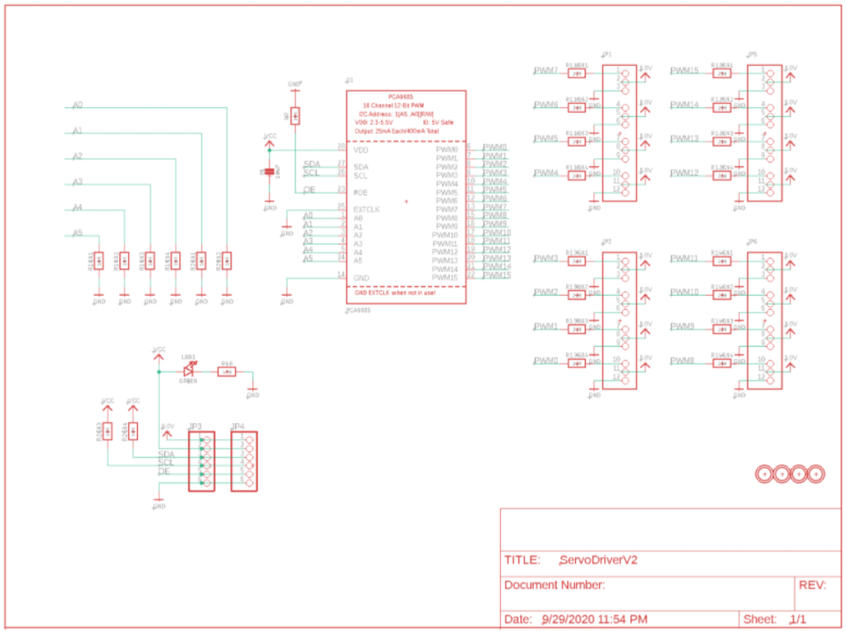
6 Hardware Design

This project requires a combination of electrical and mechanical systems which are combined together in order to create a working device. Each aspect of the construction of the robot must be taken into account to make sure that it works mechanically and electrically. This section explores the different components of our hardware design.

6.1 PCB

For our PCB design we designed a driver that has multiple inputs for all the motor and servos we have. The brain for this PCB is a PCA9685 chip that is a I2C-bus controlled 16-channel controller. The chip is mainly use for LED application as it is capable of Pulse-width modulation (PWM) to reduce the power delivered by an electrical signal. We can use this technology to control the amount of power we want delivered to our motors and servos. This is a very essential part of project since we want no delay in our motor output commands, versus having our microprocessor do this and lead to output delays.

We designed the PCB using Autodesk EAGLE[17]. EAGLE has a massive library of components with millions of parts to choose from that come from all the different manufacturers out there. Their software is very user friendly with a lot of sources out there to help aid us in our design. They even have features in the software to help with the wiring layout of the board to find the best and cheapest route for our design.



TITLE: ServoDriverV2

Document Number:

REV:

Date: 9/29/2020 11:54 PM

Sheet: 1/1

Figure 6.1.1: Eagle Schematic

We designed the schematic using the servo motor from Adafruit as inspiration. We then transferred that design to a board and lay out all out components and wires. We then sent the design to be manufactured by JLCPCB.

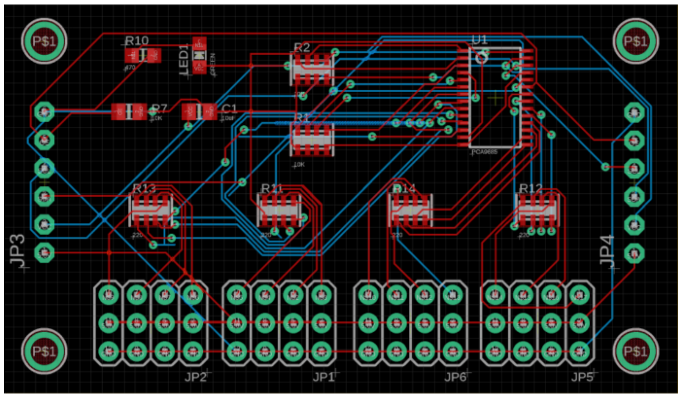


Figure 6.1.2: PCB Schematic

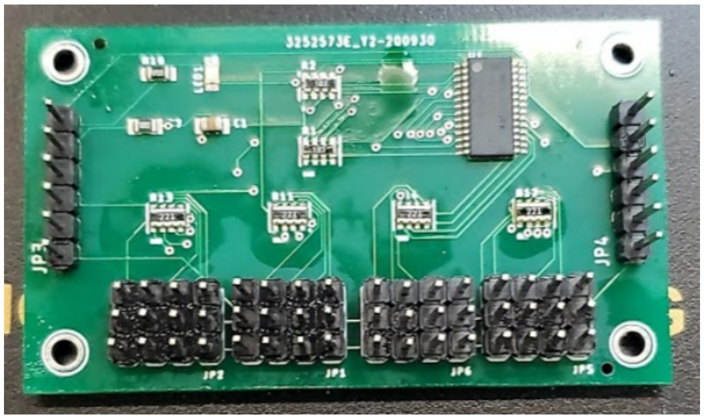


Figure 6.1.3: PCB

As shown in the diagram below, the entire movement of the robot depends on the servo driver that we made. The main brain of this robot is the Jetson Nano where the PCB will act like the nervous system that will be responsible for all motor commands and functions. Without it we would have to to use a different form of a motor controller, or solely depends on the Jetson to do all that, but as stated before we would run into output delays, and hardware capability issues since the motors cannot plug directly into the Jetson.

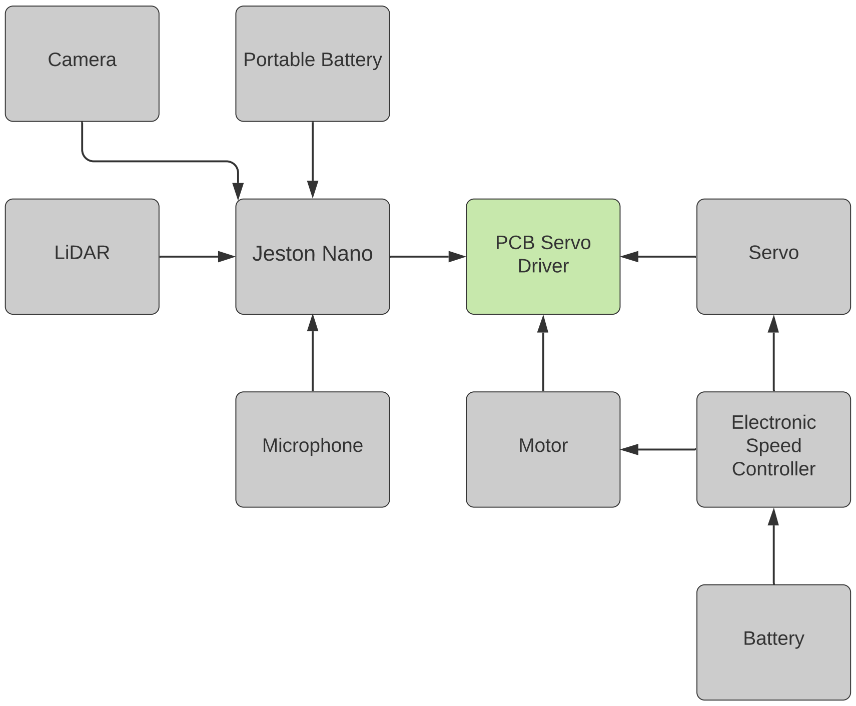


Figure 6.1.4: PCB Diagram

6.2 Mechanical System

This section details each mechanical component of our build. The specifications for the chassis are shown in Table 16. The reasoning behind using a RC Car was for its affordability and its availability. We also want to advertise this project as a low cost project. Using an expensive base to use as our robot would not be ideal for our budget.

RC Car Specifications	
Dimensions(cm)	60 x 23 x 10
Weight (lbs)	4.1
Material	Composite Nylon
Clearance (cm)	2
Wheels/Axles	4 / 2
Motor (electric)	RC380
Steering Servo	Sp6003
Differential	Planetary
Steering	Single Point
Gear Ratio	2.72 (Final Drive 12.34)

Table 16: RC Car Mechanical Specifications

6.2.1 Chassis

The chassis for our robot is taken from a RC monster truck with all the unnecessary components removed such as the truck body and the transmitter as seen below in Figure 6.2.1. We purchased a base plate from the Donkeycar website and 3D printed it. This base plate has both the Jetson Nano and the servo driver PCB mounted to it. The base plate is attached to the chassis using the same mounting hardware that the original body of the car used. The base plate is held in place by a simple slide in fastener.

Our base plate is 3D printed using high grade filament. Further modifications to the chassis were necessary as we started to change and add hardware. We needed to build a mount for our LiDAR and an adapter so that our Jetson would be able to fit. We did this by using a 3D resign printer that a group member had at home. Many iterations were printed and each time the design was altered until we achieved a perfect fit for the both the LIDAR and the chassis.

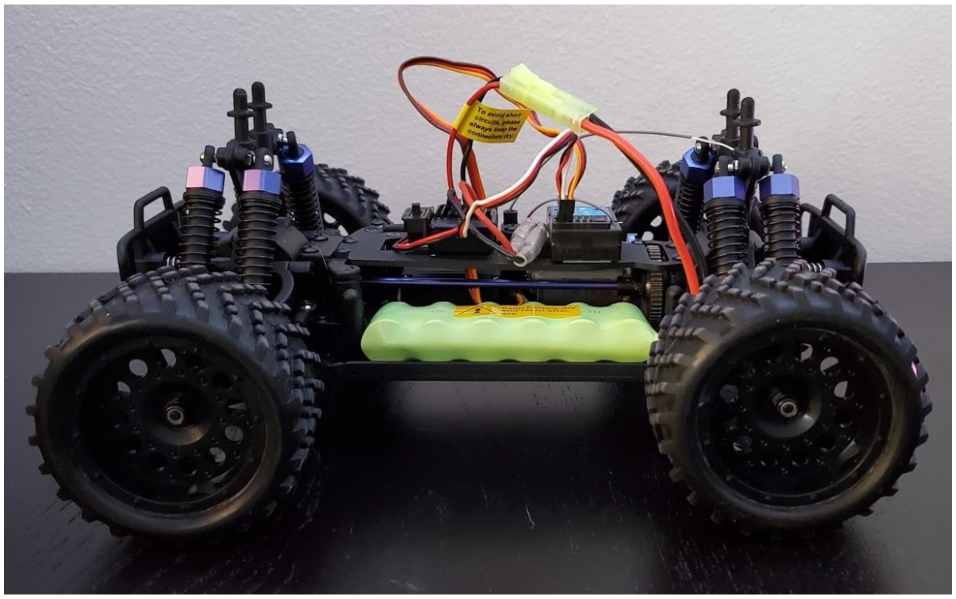


Figure 6.2.1: Chassis

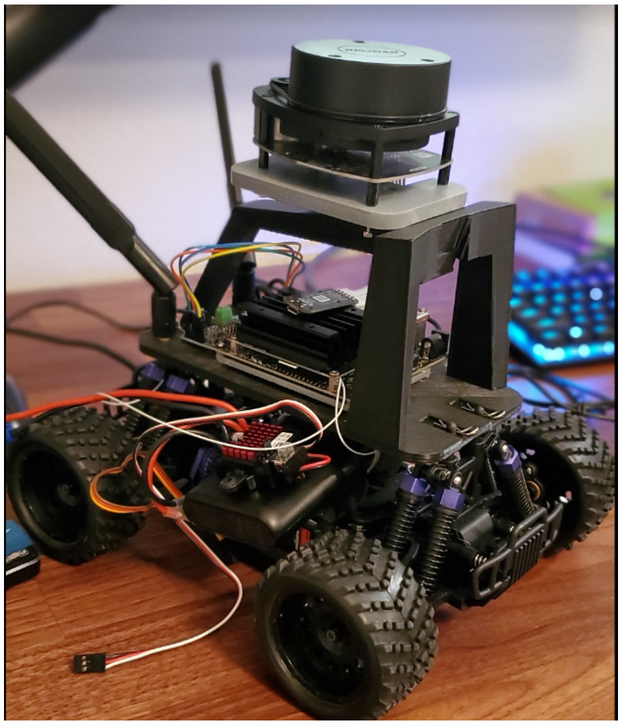


Figure 6.2.2: Base plate

6.2.2 Drive System

The drive system for our RC car is the turn steer drive system. The RC car can move forward and backwards and has one axle detected to steering to be able to make turns. The functionally is exactly the same as to modern day cars. Our turning radius is our main bottle neck as making complete U-turns in narrow hall has caused some difficulties. We choose to go this route as it was very cost efficient and it provides us with more flex ability with upgrading components as they are widely available and inexpensive.

The RC car runs off a single motor that powers all four wheels by using a differential. The turning is ran by servo that operates a rod to push and pull the steering rod to angle the wheels as seen below.

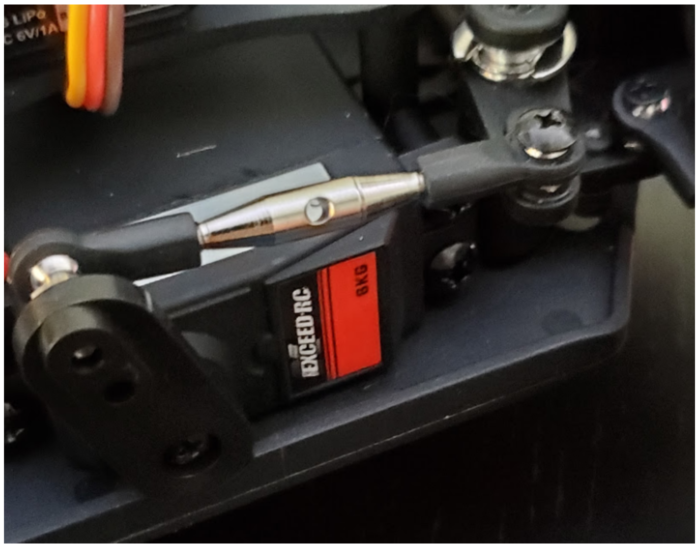


Figure 6.2.3: Servo Steering

6.2.3 Wheels

The wheels are non-slip wheels that came with the RC car as they are rated to be driven on a variety of terrain, which is more than enough since we will only be on flat surfaces. All four wheels are on independent shock absorbers that will help with keeping the base plate leveled and prevent any wobble that may accrue from uneven weight distribution.

Figure 6.2.4: Wheels

6.2.4 Motors

The RC car consists of one motor to operate the throttle, and one servo motor to operate the steering. The stock wiring of the motor and servo will need to be modified. The motor is a Exceed-RC RC380. The servo is Exceed-RC Sp6003 High Torque servo, both are connected to the electronic speed control (ESC) that is used to control and monitor the output and speeds. The motor is built for speed as this is a racing RC car, however our implementation never uses full throttle since we want slower and steady movement.

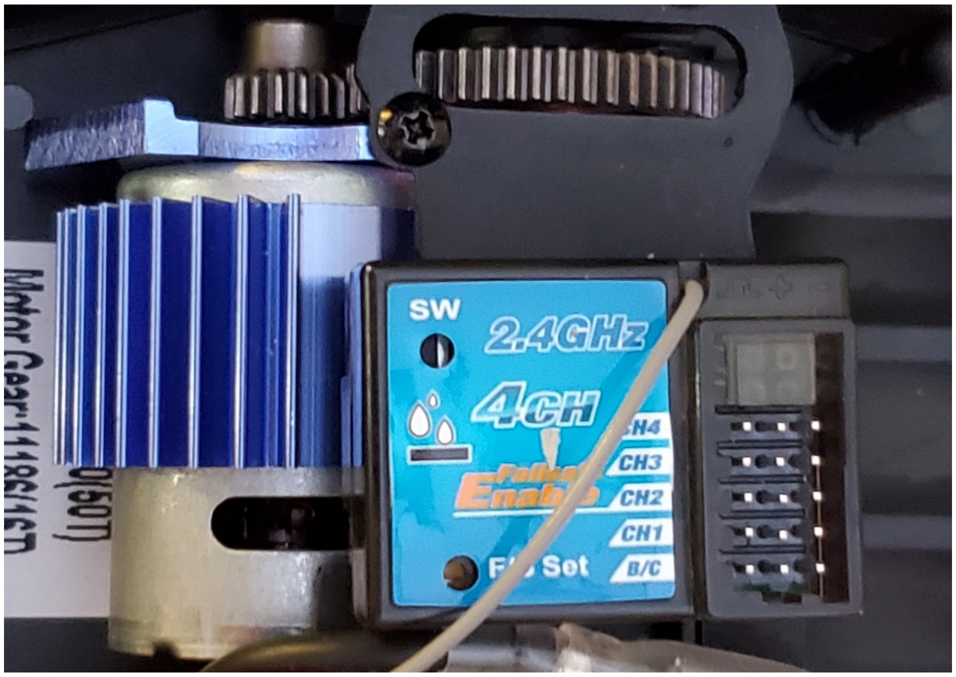


Figure 6.2.5: Electronic Speed Controller and Throttle Motor

Power from the battery goes into the ESC. From the ESC two wires come out for each the motor and servo. Those are then connected to the RC module which we did not need. Instead we got rid of the RC transmitter and connecting them to a SunFounder PCA9685 16 Channel 12 Bit PWM Servo Driver [18]. The servo driver is wired to our Jetson Nano, this will give us full control of the car. The purpose of this servo driver is to control multiple servos and motors simultaneously.

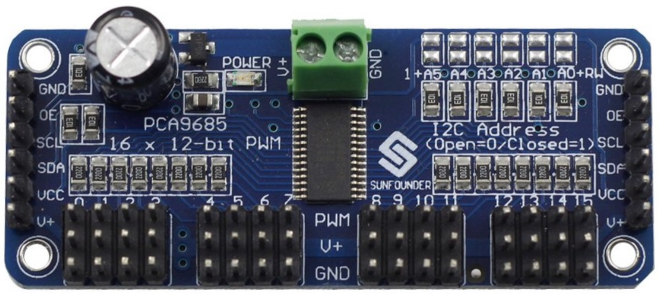


Figure 6.2.6: SunFounder Servo Driver

The throttle motor gear set up is initially set up for high speeds, since our project doesn't call for being fast, we thought about changing the gear ratio for more torque if necessary as we might need the extra power to be able to move the weight of all the hardware we install on the chassis. The current gear ratio is 2.72, as we try to aim for something around 5 with the driver gear having far less teeth than the follower gear. We did not end up needing to do this as we were able to pull the weight of the robot just fine.

6.3 Electrical System

In this section we will be going over the parts and specifications for each of our electrical systems such as, batteries, chargers, sensors, and our microcontroller. The main focus on theses parts are to be inexpensive without lacking quality, reliable, small, and lightweight.

6.3.1 Battery

The batteries that we ended up using are two independent batteries. One for the RC Car that is rated for our drive system and one is a portable power bank to power our Jestson, to which all our sensors are plugged into. We ended up needed to purchase a bigger battery for the RC car as the stock battery was not giving us enough drive time to be able to run multiple runs. For our power bank we had to get something that outputted more amps and was bit smaller as the Jetson need 4.8amp to run in full power mode. We also had a weight issue were the battery we used for the prototype weighed down the robot too much.

We ended up getting a Venom 5000mAh battery for the RC car and a Anker 20100mAh portable battery that is capable of outputting 4.8A.

RC Battery Specifications			
Model	Traxxas 2926X	Exceed-RC	Venom 5000
Type	NiMh	NiMh	LiPo
Capacity	3000mAh	1100mAh	5000mAh
Voltage	8.4 V	7.2 V	7.2 V
Run-time	15+ Minutes	20+ Minutes	40+ Minutes
Dimensions (mm)	139 x 42 x 48	100 x 35 x 18	135 x 45 x 45
Weight (oz)	8.4	4.0	10.2
Price	$35.99	$19.99	$44.99

Table 17: RC Battery Specifications

The two batteries we used for the RC car are the stock Exceed-RC battery and the Venom 5000 that we ordered.

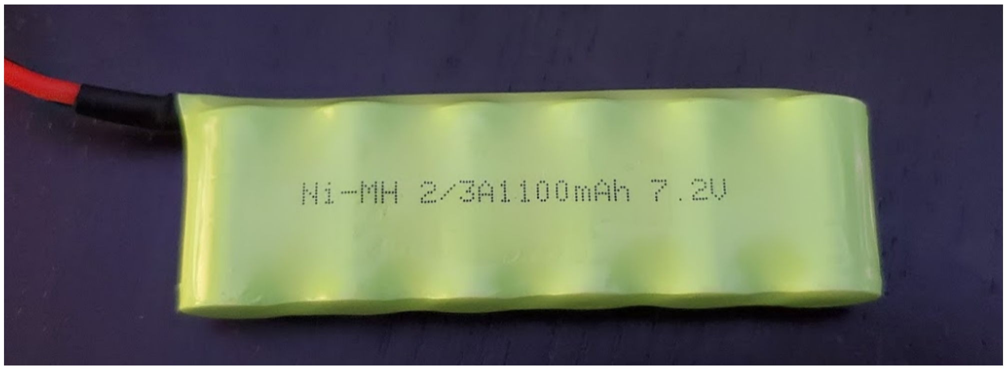


Figure 6.3.1: Exceed-RC Battery

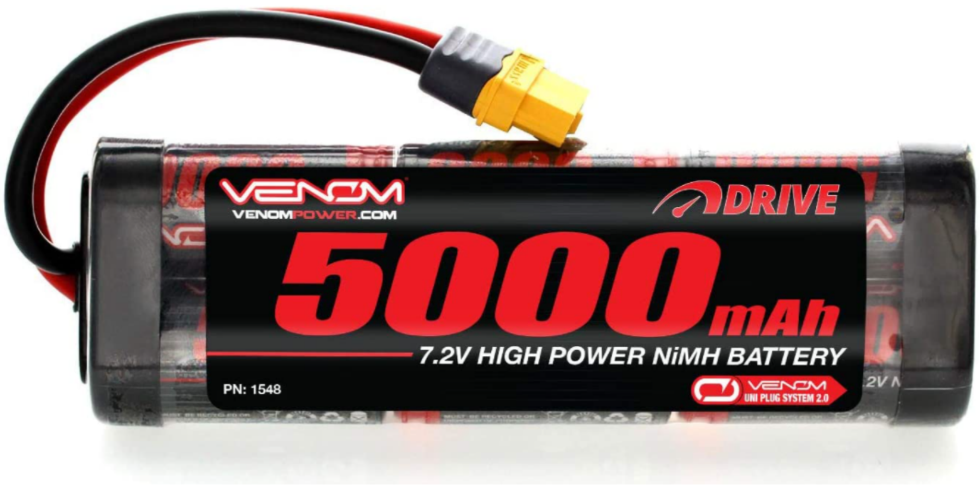


Figure 6.3.2: Venom-RC Battery

For our Jetson we used the 20100mAh portable power bank from Anker. This battery meets the Jetson's power requirements of 5V 4.8A. This also meets our requirements of being a high capacity battery. All the power going to the Jetson will be powering all the sensors and servo driver.

Portable Power Bank Specifications	
Model	Anker
Capacity	20100mAh
Charging Time	10 Hours
USB-C- Input/Output DC	5V 3A, 9V 2A, 12V 2A, 14.5V 2A
USB Output 1 DC	5V 4.8A
USB Output 2 DC	5V 4.8A
Dimensions (in)	6.6 x 2.4 x 0.9
Weight (oz)	12.5
Price	$49.99

Table 18: Portable Power Bank Battery Specifications

Figure 6.3.3: Anker Portable Battery
(Reprinted with permission from Anker)

6.3.2 Power Distribution

We had two methods of distributing power throughout our robot. We can run the entire drive system with the Jetson on the same battery. Or we could independently run the drive system and the Jetson each on their own battery.

The drive system draws power from the RC car batteries. The Jetson that powers all the sensors draws power from a portable battery that we have mounted to the robot. The reasoning behind using two batteries is for two main reasons. We don't want to drain the RC battery more than it already does while running. Also, when activating the throttle there is a voltage drop to the Jetson that causes it to either turn off or switch to low power mode which affects our facial recognition software.

6.3.3 Single Board Computer

We decided to go with a Jetson Nano as they are readily available and their technology and community has matured a lot over the years.

Jetson Nano Developer Kit Specifications	
GPU	128- Core Maxell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	64Gb microSD
Video Encode	4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet and M.2 Intel® Dual Band Wireless-AC 8265
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Dimensions (mm)	69 x 45 , 260-pin edge connector
Price	$99.99

Table 19: Jetson Nano Developer Specifications

6.4 Sensors

In this section we will be going over our chosen sensors that are used in our robot. The main requirement for these sensors is that they must be compatible with our hardware and software design. We choose theses sensors since they fit all of our need as far as size, power draw, and usability. There was many other options, but we would have run into issues that required us to change our design as well as budget.

6.4.1 Camera

For our camera we originally went with a Raspberry PI Camera Module by Kuman. It's a 5MP camera capable of taking images at 2k resolution and videos at 1080p 30 frames per-second, or 720p at 60 frames per-second. It is equipped with two infrared sensors for nigh vision capabilities in the even the robot needs to see in the dark. However, after switching to the Jetson we ran into a compatibility issue and we had to use a USB webcam. We ended up using a Logitech C615 Webcam.

Using this webcam we are able to get much more FPS than we were with the Raspberry Pi since the Jetson is much more powerful when it comes to image processing therefore giving us a much better video feed, and detection accuracy.

Figure 6.4.1: Logitech C615 Webcam

6.4.2 LIDAR

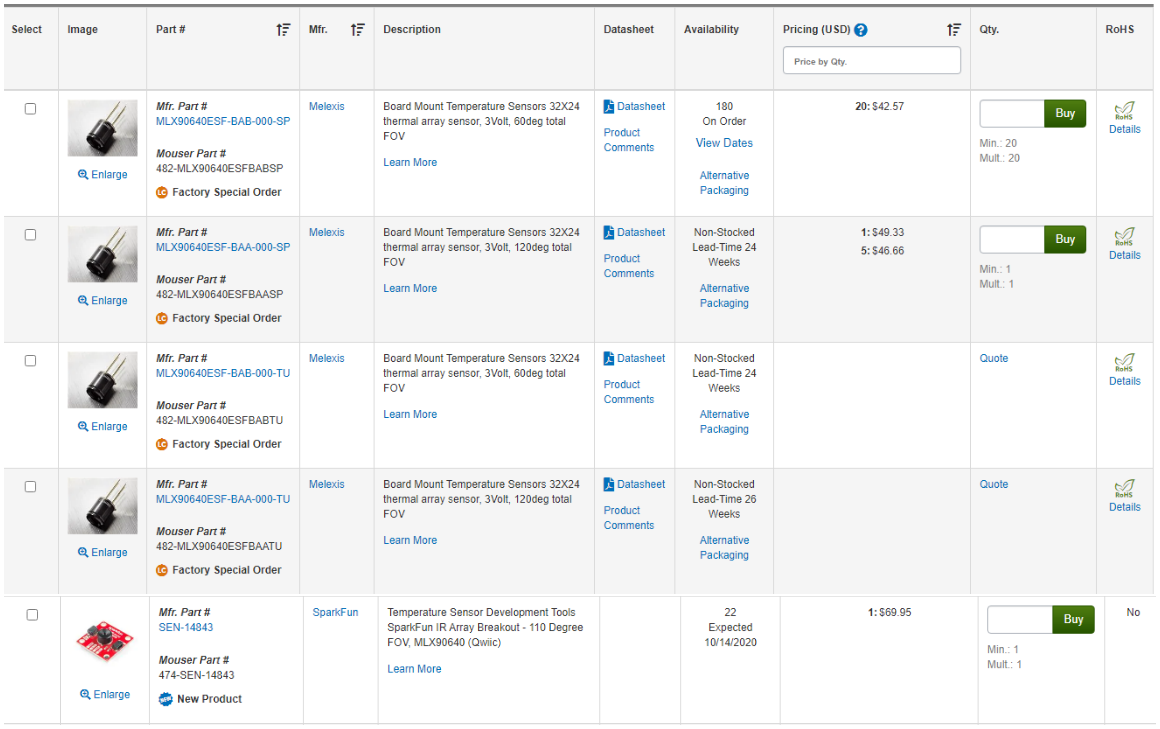
Although there are a wide range of Lidar sensors available the vast majority of prices are absolutely exorbitant for a student budget. Though Lidar is integral for the project we decided to settle on the RPLidar A1M8, as seen in Figure 3.8.2, for the 360 degree omnidirectional laser scan, as opposed to the Hokuyo 270 degree field of view. Additionally the RPLidar has a 12m range as opposed to the Hokuyo 8m range. More importantly because the Hokuyo is nearly $2,000 dollars it would only be available on loan from the robotics team if we were able to access the resources available on campus though since that isn't the case due to restrictions with COVID-19 and accessibility's to the campus. We decided to go with the $100 option in the RPLidar A1M8 that would come on time and still provide us with a reliable Lidar.

6.4.3 Thermal Sensor

The Melexis MLX90640 Far Infrared Thermal Sensor is a 32x24 (768 pixel) IR array with a 110 x 75 degrees field of view was the clear winner in terms of pixel to price performance. In the sweet spot between Panasonic GRID-EYE 8x8 (64 pixel) infrared array sensor with a 60 degree field of view and the FLIR Lepton 2.5 80x60 (4,800 pixel) readout with a 50 x 63 degree field of view. Additionally for the extra 30$ in price the jump in quality and pixel count of the MLX90640 was significant enough to warrant the increase in price when in comparison to the $140 dollar price jump to the FLIR; it became a cost difficult to justify.

Once the decision was settled to order the Melexis MLX90640 other difficulties began to arise. When searching across several sights for this sensor we found SparkFun SEN-14843 MLX90640 (Qwiic) at $69.99 offering both variations of the 60 degree and 110 degree field of view, connected through with a Teensy 3.2 in mind as the Arduino Uno (or equivalent) doesn't have enough RAM or flash to complete the complex computations required to turn the raw pixel data into temperature data; as claimed by SparkFun.

Though trouble arose when the part was unavailable everywhere we looked, one of the only providers whom even on their own website claim to be out of stock Pimoroni were producing a breakout compatible with our Raspberry Pi from it's conception and having available stock on a few secondary retailers like Mauser, we decided that this would be the product we chose. With board mounted sensors requiring a minimum order quantity of twenty or with a unknown restock time on Mauser Electronics, and restock time on sensors from Sparkfun looking nearly 3 months in advance it was truly our only viable option. Shown in the image below is a screenshot of the inventory of Mauser Electronics which reflects similarly on other electronics retailers substantiating the claims made above. Whether as a result of the global pandemic, or just lack of production to meet demand from Melexis, this part is relatively rare. Ultimately due to this aforementioned rarity throughout the crisis, and potential to break the calibration settings of the MLX90640 with the Jetson due to incompatibilities in the libraries provided by Melexis we chose not to implement this portion of the project as it would have been irreplaceable given a single mistake.



Select	Image	Part #	Mfr.	Description	Datasheet	Availability	Pricing (USD) Price by Qty.	Qty.	RoHS
	Enlarge	Mfr. Part # MLX90640ESF-BAB-000-SP Mouser Part # 482-MLX90640ESFBABSP Factory Special Order	Melexis	Board Mount Temperature Sensors 32X24 thermal array sensor, 3Volt, 60deg total FOV Learn More	Datasheet Product Comments	180 On Order View Dates Alternative Packaging	20: $42.57	Buy Min.: 20 Mult.: 20	Details
	Enlarge	Mfr. Part # MLX90640ESF-BAA-000-SP Mouser Part # 482-MLX90640ESFBAASP Factory Special Order	Melexis	Board Mount Temperature Sensors 32X24 thermal array sensor, 3Volt, 120deg total FOV Learn More	Datasheet Product Comments	Non-Stocked Lead-Time 24 Weeks Alternative Packaging	1: $49.33 5: $46.66	Buy Min.: 1 Mult.: 1	Details
	Enlarge	Mfr. Part # MLX90640ESF-BAB-000-TU Mouser Part # 482-MLX90640ESFBABTU Factory Special Order	Melexis	Board Mount Temperature Sensors 32X24 thermal array sensor, 3Volt, 60deg total FOV Learn More	Datasheet Product Comments	Non-Stocked Lead-Time 24 Weeks Alternative Packaging		Quote	Details
	Enlarge	Mfr. Part # MLX90640ESF-BAA-000-TU Mouser Part # 482-MLX90640ESFBAATU Factory Special Order	Melexis	Board Mount Temperature Sensors 32X24 thermal array sensor, 3Volt, 120deg total FOV Learn More	Datasheet Product Comments	Non-Stocked Lead-Time 26 Weeks Alternative Packaging		Quote	Details
	Enlarge	Mfr. Part # SEN-14843 Mouser Part # 474-SEN-14843 New Product	SparkFun	Temperature Sensor Development Tools SparkFun IR Array Breakout - 110 Degree FOV, MLX90640 (Qwiic) Learn More		22 Expected 10/14/2020	1: $69.95	Buy Min.: 1 Mult.: 1	No

Figure 6.4.2: Scarcity of Melexis MLX90640

6.4.4 Microphone

The microphone we will use will have to be determined after we install all other components to the robot so we can conduct test on the noise level the robot produces when operating. We will be using a decibel reader and from that data we can determine what microphone will fit our needs, since we are trying to avoid any sensitive microphones that will alert us just by the sound of the robot motors. If sound from the robot being picked up becomes an unavoidable problem we will keep the microphone feature but not have it influence our code to alert the user of an intruder. It will be used for remote communication with the people around the robot if needed.

7 Software Design

Our software is an intermediary between the sensor data and the output motors. The robot requires software to process all the information coming in from the sensors and make determinations about the environment and whether or not to act upon the processed data. Our software stack enabled us to create modular software that makes it easy to maintain or update in the future. This was done by using common programming paradigms which is discussed in the subsequent sections.

7.1 High Level Software Design

To facilitate the use and programming of our sensors, we used ROS (Robot Operating System) for the robot's operation. ROS uses a publisher-subscriber model, which means that our code is made up of individual nodes that each have a specific purpose. The nodes communicate with each other by publishing to topics or subscribing to topics.

One of our nodes is the autonomous navigation or LIDAR node. Our LiDAR sensor publishes spatial data to a ROS topic, which the navigation node subscribes to in order to run the navigation algorithm (find the gap) and produce motor commands. If user directional control input is received from the web server, the navigation node switches to teleop mode, where it will only forward commands from the manual input on the web user interface to produce motor commands. If no commands are received for one consecutive minute, the navigation node returns to autonomous mode.

Another node is the detection node. Our camera will publish vision data to this node where computer vision is used to to determine if a person is in the frame. If the confidence rate is above a 70 percent the topic then publishes the detection to the alert node.

The third node is the emergency user contact routine or alert node. If the intruder detection node publishes that a person has been detected, the emergency user contact node will send a text message to the user inviting them to a video call. The user can then join the video call to see the footage from the front of the robot, and use the microphone and camera to have a conversation with the intruder.

The user can also access a web interface that contains current video from the camera on the robot and a joystick to drive the robot. If the user begins to use these, the robot switches from autonomous to teleoperated drive, and will resume autonomous patrol if one minute passes with no interaction from the user.

7.2 User Interface

The user interface consists of a web page that the user can access which is hosted on the robot. The robot will also be able to make a call to the user or send a text message.

7.2.1 Graphic User Interface

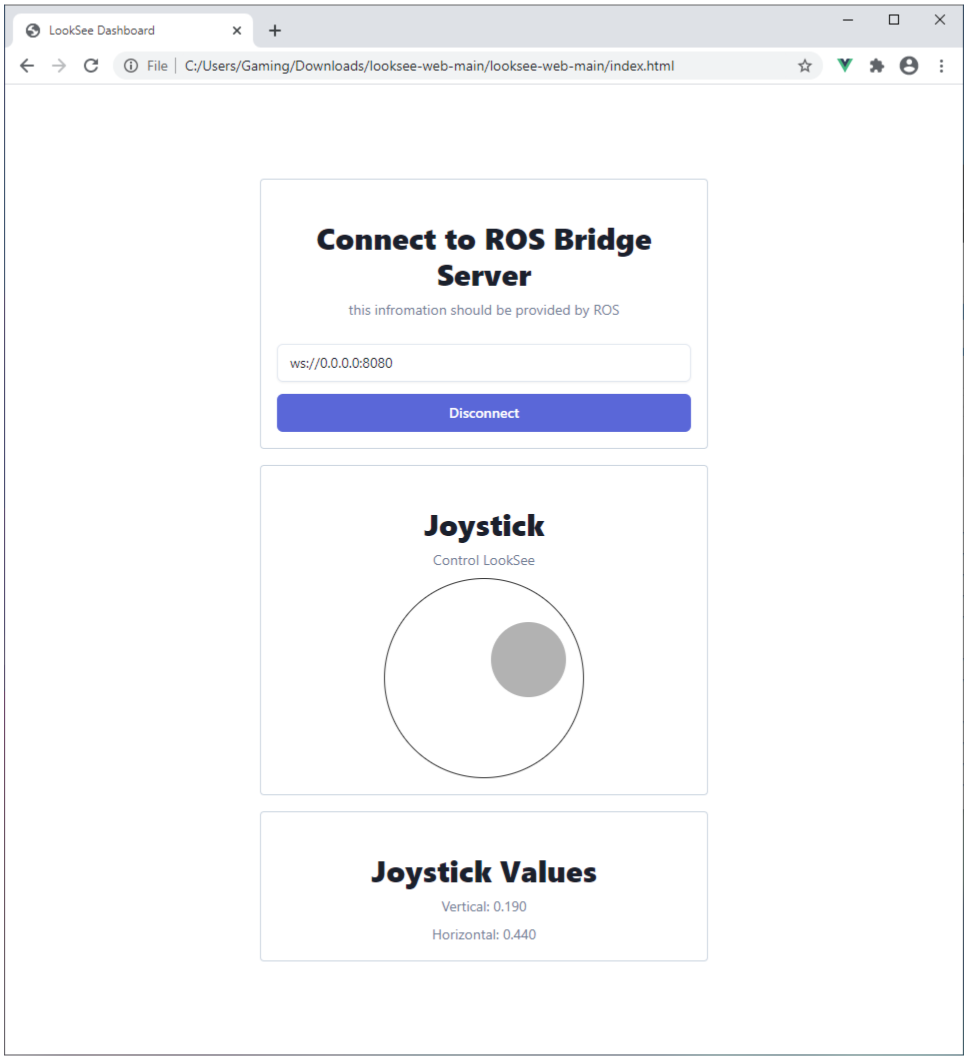

LookSee Dashboard
File
C:/Users/Gaming/Downloads/looksee-web-main/looksee-web-main/index.html
Connect to ROS Bridge Server
this infromation should be provided by ROS
ws://0.0.0.0:8080
Disconnect
Joystick
Control LookSee
Joystick Values
Vertical: 0.190
Horizontal: 0.440

Figure 7.2.1: LookSee Graphical User Interface Mockup

ROSBridge is a ROS component that creates a JSON API for a ROS system so non-ROS programs can interact with it. We used ROSBridge and utilized the exposed API using roslibjs, a javascript library for making ROS API calls.

A frontend dashboard was created using HTML, CSS, and Javascript. Both a CSS and Javascript framework were used. TailwindCSS is a utility first framework that we chose for the styling of components and Vue.JS is the framework we chose for the interactive elements of our interface. Our Vue application makes calls to the exposed ROSBridge API through roslibjs. A joystick was created using VueJS which published ROS Twist messages to the robot_driver node with a corresponding velocity and angle component. This can be used to control the motion of the robot.

The web page is hosted using a python SimpleHTTP server and is accessed by visiting the IP address of the robot and the exposed port on an external device connected to the local network. Then the UI connects to the ROSBridge API by providing it with the ROSBRidge address URL which is automatically filled in for you. Once connected the controls and video feed appear and allow you to control the robot.

The original plan was to use LIGHTTPD web server, but as responsibilities were shuffled around on the team, one member discovered ROSBridge and realized that the available virtual joystick it provides would be much preferred to the buttons that would be necessary if the other web server was chosen.

The idea with the web server interface is to allow the human user to take over driving of the robot if they see something that they would like to investigate on the camera feed. For example, if the human user is watching the robot over video, and notices that a piece of furniture has been knocked over, they can take over manual driving to find the source of the problem. If the user stops using the virtual joystick, autonomous driving will resume one minute later.

7.2.2 Video Stream

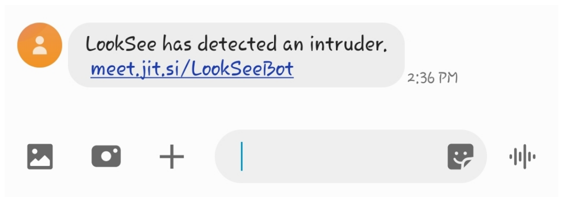
The user interface was supposed to include a video stream however we were unable to get this aspect working. While the detect node did output a video stream with the detection bounding boxes, we were unable to get the video stream publishing to the dashboard. I believe we did not yet find a library capable of handling the way in which we streamed the video from the detection node. The video was encoded using h264 and then streamed using the real-time transit protocol (RTP). The library we chose to handle viewing the stream, mjpegcanvasjs, was unable to process the RTP stream it received.

7.2.3 Twist Command Interface

Through the user facing web page, the user is able to operate a joystick which sends both throttle and steering commands via a twist message. Using the roslibjs library, we were able to have our dashboard application publish Twist messages directly to the cmd_vel topic. From there a servo interface, the same one the autonomous navigation code interfaces with, decodes the throttle and steering commands and sends them to the servo driver.

7.2.4 Phone Call Software

There is a node that subscribes to the (detectnet/detections topic) and, if an object is detected, checks to see whether the detected object was a person. If so, the node causes Twilio to send a text message to the user with an invitation to a Jitsi Meet video meeting.



LookSee has detected an intruder.
meet.jit.si/LookSeeBot
2:36 PM

Figure 7.2.2: Text from LookSee
(Screen capture from team member's phone)

7.2.5 Connection Timeout

The connection timeout checking is one of our stretch goals. We will have an additional process that periodically checks for internet connectivity. Each repetition, the last time that internet was detected will be saved in a variable. If internet is not detected, the variable is not updated, and the current time is subtracted from the time saved in the variable. If greater than ten minutes have passed, the time when internet was lost is placed in a text file. When internet is restored, the time internet was restored will be placed in a text file. This will allow the user to know when connectivity with LookSee was not possible, or when a lapse in surveillance may have occurred.

7.2.6 Scheduled Start

Scheduling a start time for LookSee is one of our advanced goals. This would allow the user to input a time of day in the UI and LookSee would start its surveillance automatically at that time. In order to achieve this, a task will run in the background of the robot at all times. This task, unconnected to ROS, will reference a text document. Entering a time in the UI will update this text document. At the scheduled time, the task will start the ROS package to begin LookSee's normal operation.

7.3 Sensor Processing

As LookSee is autonomous, all of its ability to traverse the environment and trigger other tasks is dependent on its ability to process sensor data and make appropriate judgments. The camera is used to collect image data from the area in front of the robot, the NVIDIA ROS deep learning nodes process that data to determine whether there is a human in the image, among other various objects. The LiDAR is used to collect environment data, and our code processes it to come to decisions about how to traverse the environment. The microphone is fed through the program we write using the Jitsi Meet API to conduct phone calls. The below section details how the robot will process sensor data.

7.3.1 Camera Human Detection

Object detection using computer vision was done using NVIDIA's DetectNet node which is built on TensorFlow and uses TensorRT for optimization. When looking for different solutions we used a Google research paper that compared and contrasted the different neural network architectures based on performance [1]. We found that the combination of the SSD and mobilenet architectures was the most lightweight and performant pre-trained model that fit our accuracy and performance requirements. The Single Shot Detector using the MobileNet for feature extraction performed the best when it came to the accuracy metric mean average precision (mAP) and GPU time in milliseconds for each model. We ended up using the Single Shot MobileNet v2 (ssd_mobilenet_v2) which has been trained on the COCO (Common Objects in Context) data set. It was able to accurately detect a person just by their legs, arms, torso, head, etc along with additional objects.

The NVIDIA ROS integration of the detectnet software provided the detections and the output video as topics in ROS that other code could subscribe to. This allowed for a simple and seamless implementation.

7.3.2 Microphone Voice Streaming

The microphone that we purchased for LookSee will be used by the Jitsi Meet phone call code, but will not be used directly by ROS. For this reason, we will ensure proper configuration of the microphone so that it can be utilized by our phone call code, but the voice streaming will be handled by Jitsi Meet.

7.3.3 Microphone Speech Transcription

Microphone speech transcription is considered to be an advanced goal, but it will likely be one of the first advanced goals we implement. In the core configuration, the microphone will not be used by ROS, but in the advanced configuration, we will use the ROS package speech_recog_uc to listen for words. speech_recog_uc uses the Google Cloud Speech API to identify words real-time in a constant stream of audio. Our code would use speech_recog_uc to find words while the robot is running, and if more than 3 words are identified in the space of one minute, the transcription of those words will be sent via text message over Skype to the emergency contact. Depending on how loud the robot's normal operation is, we may elect to have the robot pause navigation if a suspected word is heard so that the rest of the audio is more clear.

7.3.4 LIDAR Environment Processing

The LiDAR unit we selected has a ROS node supplied by the developer that will publish its measurements to a ROS topic. This will manifest as a stream of (degree, distance) coordinates of locations that have been pinged against in the most recent scan. With a full set of these coordinates, we can analyze data and evaluate the environment LookSee is within.

7.3.5 Gyroscope Tamper Detection

Depending on the data available for this Gyroscope/Accelerometer chosen, likely the MPU-6050 3-axis gyroscope and 3-axis accelerometer on the same integrated chip. With it's embedded motion processor 9-axis of data are provided from the IC. For it's low cost and high functionality we believe it to be the best option. As a result of adding this device to LookSee new functionality can be added in the form of Tamper Detection. Primarily we'll use the data given as additional telemetry information to provide to the autonomous navigation algorithms, but in the case that LookSee is flipped in orientation, it can alert the owner for it to be righted, or if some perpetrator thrashes the robot it will still be capable of providing information to the owner in the event that other critical subsystems (ie. the camera feed is upside down, thus the robot must be upside down) are compromised by an unforeseen event. Though we never purchased a gyroscope, we believe this may be an important function for future iterations and have chosen to include this.

7.3.6 Video Backup

The video backup feature is a stretch goal. In the event that we do implement it, we will use the ROS package image_view, which contains a tool called video_recorder. When a flag is raised on the (intruder_detect) topic, a phone call will be made to the emergency contact. If the emergency contact does not pick up, LookSee will begin recording video for ten minutes using video_recorder. This video will be able to be retrieved later by the user.

The reason for using ROS to implement this is that technically, only ROS is accessing the camera, and all of the other programs using the camera, such as the video recorder, the web server, and the computer vision algorithm, are actually only subscribing to the ROS topic containing the camera feed. This bypasses the limitation that, normally, only one process can access a camera at a time, and allows us to use the camera for as many functions as we would like.

7.4 Navigation

LookSee is able to achieve it's designated proposed requirements of autonomous navigation and surveillance. In addition to autonomous surveillance, manual control is also possible through a web dashboard. Below we detail all movements and navigation code that LookSee utilizes.

7.4.1 Find the Gap Navigation

At this point, we fully settled on using Find the Gap navigation, but the concept is that a 360 degree scan will be made of the room. ROS will deliver that scan to our Find the Gap node, which will evaluate the distance between each of the consecutive points. To cut down on processing time, we will simply subtract the radius of one point from the next, to identify traversal gaps for LookSee and approach the furthest point. After inflating walls (by roughly half the width of the car extend the short wall over the further wall), identified through these discrepancies in distance readouts (or gaps). LookSee has effectively converted itself into the point object space, where effectively if LookSee is able traverse the gap in Lidar readouts then in the real world there will be no issues in traversal.

7.4.2 User Controlled Navigation

User controlled navigation is handled through a virtual joystick on the web dashboard, which publishes to the ROS movement topic cmd_vel. Every time the joystick is manipulated, a timer will set to one minute since the last time the user interacted with the joystick. Once that timer reaches zero, the robot will resume autonomous navigation.

The user controlled navigation will be restricted to the joystick. A joystick was the best option for manual operation because all pieces of the steering and throttle can be operated with simple click and drag movements. It also allowed the user to apply varying throttle speeds without any effort. The joystick has an added benefit that most people can intuitively use a joystick without an training.

7.4.3 Path Recording

We considered path recording to be an advanced goal. This is when the user can drive the robot around a path and the robot will remember the directions taken so that those directions can eventually be recited by the robot to reproduce the same path. There is some danger in using this method, as the robot is effectively operating on dead reckoning. Minuscule changes in the direction or location where the robot is placed will result in a vastly different resulting path. However, with augmentations such as a docking station to standardize the start point or visual cues, this could become a legitimate way to script LookSee's patrol. We were unable to realize this stretch goal, though in further iterations of the Robot we would've liked to see this as well as some form of mapping implemented.

7.4.4 Motor Commands

Our robot will require both teleoperated and autonomous driving. Both of these types of driving will use a single node called the servo_interface which will convert throttle and steering commands that consist of values from -1 to 1 to the actual values that get transmitted to the ESC and servo. This servo interface subscribes to the cmd_vel topic via a Twist message. This Twist message consists of linear and angular commands, the linear controlling the throttle and the angular controlling the steering. The commands take values from -1 to 1 for both throttle and steering. The value -1 represents either full throttle reverse or max left steering angle while 1 represents full throttle forward or max right steering angle. This twist message is what both the teleop and autonomous driving nodes will send.

The serov_interface node takes these cmd_vel commands and converts them to the actual PWM signal that both the ESC and servo receive. This servo_interface also has a timeout function in case a controlling node disconnects that sets the throttle to zero and the steering to neutral if no command is recieved after five seconds. This prevents the situation of the controlling node disconnecting while sending a full throttle command and the robot running away.

The manual control from the web dashboard simply publishes to the cmd_vel topic. If no signal is received from the web dashboard, the ROS node will run the autonomous code. The autonomous code that runs the 'find the gap' algorithm also publishes to the cmd_vel topic when that node is in use.

7.5 Developer Interface

The developer will able to interact with LookSee through ROS and once the code has been written the developer will create individual packages navigating to the src subfolder within the ROS. The overall structure of the packages was was created using the catkin_create_pkg ROS command, each with specific dependencies. The 'robot_bringup' directory contains required files to launch the robot. This will host all our launch commands. Inside the 'robot_bringup' directory there is a 'config' directory contains the parameters file and the 'launch' directory which is the entry point for entire robot. There is a 'robot_control' directory which contains required files to control the robot's movement including the servo driver PWM node and the servo interface node.

The code can is compiled using the 'catkin_make' command in our source directory.

Two methods were used to connect with the Jetson Nano, which is the robot's primary computer. The first method is to literally plug an HDMI cable, keyboard, and mouse into the device and program on it like a regular PC. However, once we install the Nano onto the chassis, it will be difficult to interface with it in this way. An SSH connection was setup, once started we will be able to SSH into LookSee, transfer files, edit code, and run the ROS core from any of our own laptops.

7.6 Software Diagrams

Our project has been deliberately designed so that the code is modular. Code written by each member of the team interacts with code written by other members, but remains separate. Each of these pieces of code will also interact with both ROS topics and pieces of the hardware.

Because the relationships between each of the modules will result in a large and complex system, we have assembled three software diagrams that demonstrate what our project would have looked like if we had implemented all goals from the core, advanced, and stretch lists. These diagrams were only an estimate, and it was entirely possible that we would complete all of the core requirements, half of the advanced requirements, and one stretch requirement, or any other combination thereof so long as the core requirements are met. The diagrams below simply demonstrate how all pieces of software and hardware will relate to each other in three hypothetical cases.

These diagrams, due to the large number of modules, are necessarily rather large, and unfortunately must each be placed on their own separate page. In order for the diagrams to remain readable, each diagram will be on a page, and the accompanying explanation will be on the page before it.

7.6.1 Core Requirements Software Diagram

Figure 7.6.1 demonstrates the code structure of our core requirements. At the end of the project, these requirements were implemented, with some changes to the structure.

The code in this configuration is largely broken up into two major sections: those modules dealing with the navigation of the robot, and those modules dealing with the intruder detection and related functions. The only module that interacts with both is the web server, which contains the user controlled button to initialize a Skype call, the camera video stream, and the buttons to send navigation commands.

In the navigation section, everything starts with the LiDAR, which is the only sensor used for navigation in this configuration. The LiDAR populates a pointcloud2 type ROS topic, which is handled by a package provided by the manufacturer. The autonomous navigation code, which will subscribe to the pointcloud2 topic, will run find-the-gap or another behavioral navigation algorithm, and publish instructions to a navigational commands topic. This topic will be populated by either the autonomous navigation algorithm or the user interface, which contains buttons that trigger a terminal command to write to this topic. In the event that a navigational command is sent from the web server by the user, the autonomous code will stop running for one minute, and the one minute timer will be reset every time a directional button is pressed again. Finally, we will have a motor command node, which reads the commands that have been published to the navigational commands topic, and triggers a motor movement, thus moving the robot.

In the intruder detection section, the code starts with the camera. The camera publishes to the image topic, which is run by a piece of code that is available natively through ROS. The image topic is also the source of the video stream on the web server. The computer vision code will subscribe to this topic, and will run its algorithm each time an image is published to it. Once the computer vision algorithm has completed running, if a person is suspected to be in the image, a flag will be published to the camera suspicion topic.

If a flag is pushed to the camera suspicion topic, the thermal vision code, which subscribes to it, will begin to run. The thermal vision code subscribes also to the thermal camera topic, and will have access to the data from the thermal array in real time. When a flag has been pushed, asking the thermal camera to check for a warm area, the thermal vision code will look for any areas of the image that are as warm as a human could reasonably appear to be. If an intruder is indeed found, the thermal vision code will push a flag to the intruder alert topic.

Once a flag is pushed to the intruder alert topic, the Skype call code will initialize a phone call with the emergency contact, push the audio through the speakers, and take microphone input to be passed back through the call. The audio from the speakers and microphone will never pass through ROS in the core configuration.

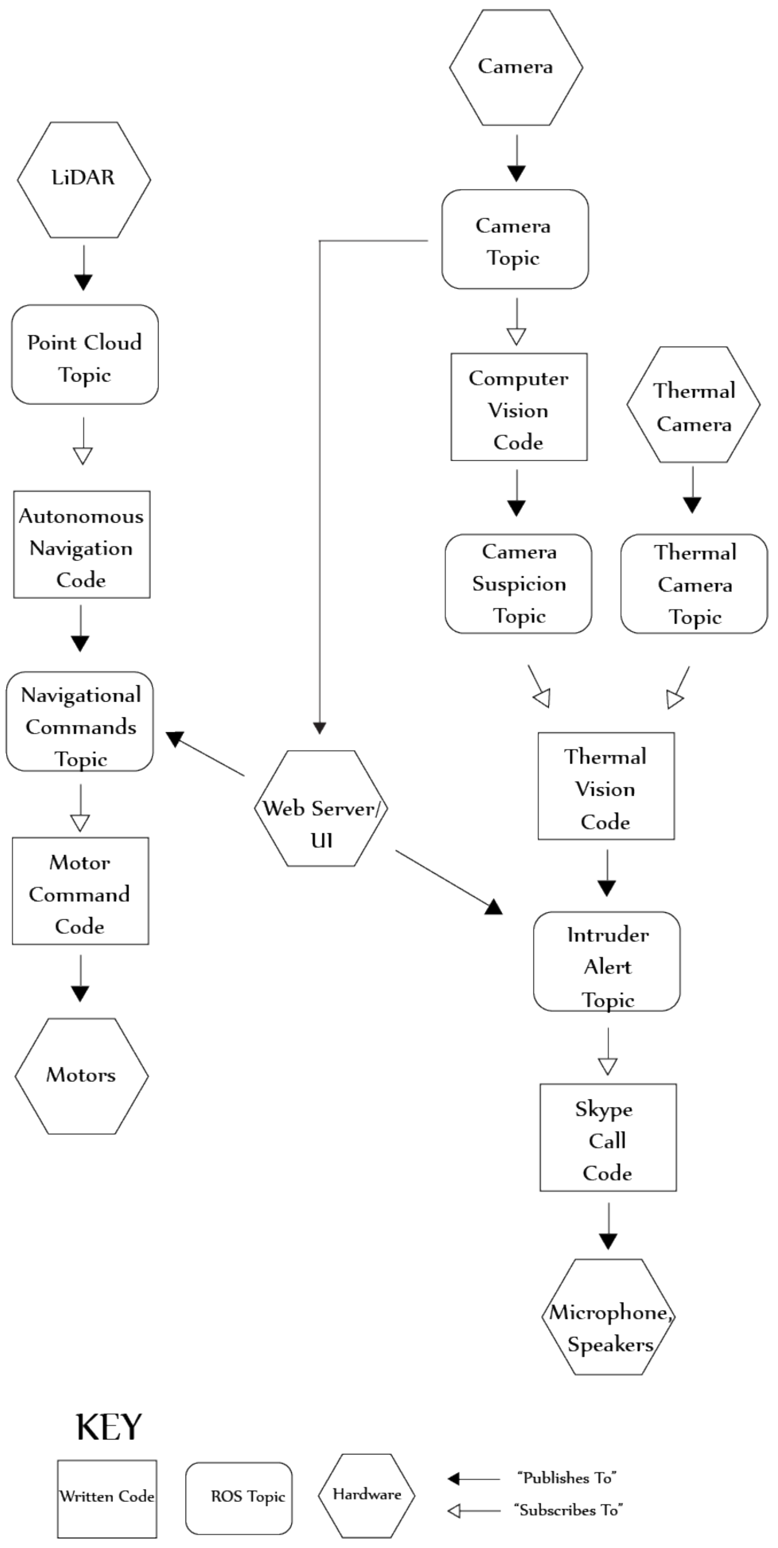


Figure 7.6.1: Core Requirements Software Diagram

7.6.2 Advanced Requirements Software Diagram

Figure 7.6.2 shows what the structure of our code, ROS topics, and hardware would look like if we completed all of the requirements for the core functionality of LookSee and moved on to complete all of the advanced requirements as well. Where the previous configuration appeared to be two long ”lines” of modules depending on each other, in this version, the code begins to become more interconnected as modules add more functionality and rely on additional parts of the code.

To begin, all of the modules that were present in the core configuration are still present and still connect to each other in the same way. The first difference is the addition of a third way for the robot to navigate. One of our advanced goals is for the user to be able to pre-record a route for LookSee to traverse each time the code begins. This recorded route is going to use only dead reckoning, which means that no environmental measurements will be used, and the robot will follow only literal commands to drive forward, drive in reverse, or turn. It will be constructed from a sequence that the user has entered by clicking buttons in the web server on a previous occasion, and stored on the Raspberry Pi's internal storage unit.

The next change to the previous configuration is the addition of code that records video from the camera topic and saves it to the Raspberry Pi's internal storage. This code is triggered by a flag being raised in the intruder alert topic, and it will run for ten minutes.

In this configuration, a gyroscope has been added to LookSee for tamper detection. Using a ROS package provided by the gyroscope manufacturer, motion data will be published to a ROS topic. Code that we write will examine the gyroscope data each frame and will raise a flag to the intruder alert topic if the motion of the car wildly exceeds the normal range to be expected by the motion of the car for more than 3 consecutive frames. This alarm is intended to be tripped if, for example, an intruder were to attempt to pick up and move the car or otherwise impede its normal operation. If this were to occur, the video recording would start and run for the next ten minutes, and the emergency contact would be called.

The final change that occurs in this configuration is the addition of the voice transcription module. In the core functionality, the microphone is only used by the Skype call code and not by ROS. In the advanced functionality, the microphone publishes its audio data to a ROS topic. A new piece of code subscribes to this topic and runs a voice to text transcription algorithm on any sounds that are deemed to be outside of the range of sounds to be expected from normal operation. If more than three intelligible English words can be made out in a window of less than one minute, the code will continue to listen until one minute has passed where no speech has been detected, and will send a direct text message to the emergency contact over Skype containing the transcribed words.

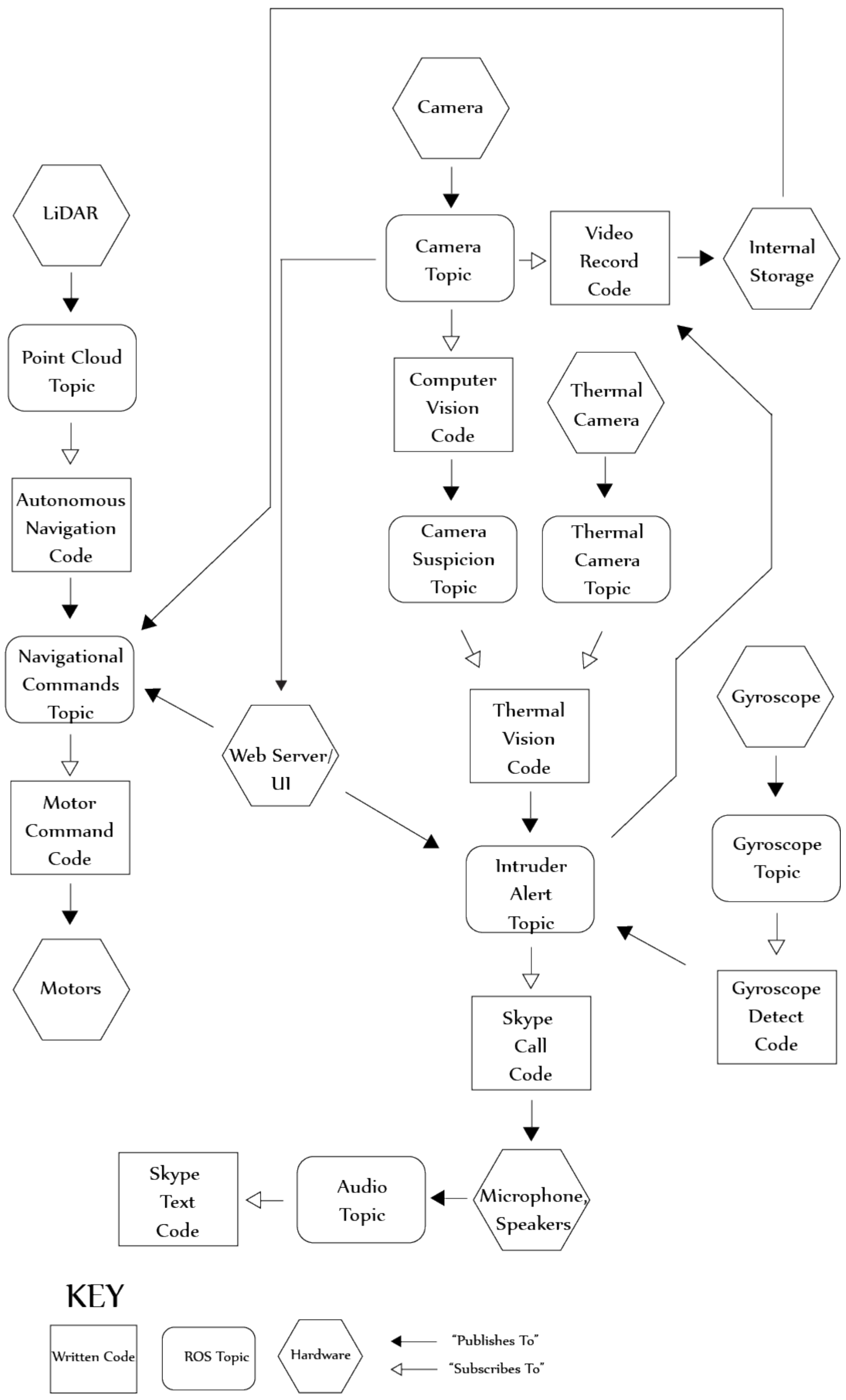


Figure 7.6.2: Advanced Requirements Software Diagram

7.6.3 Stretch Requirements Software Diagram

Figure 7.6.3 shows the complete interconnectivity of all software modules and sensors if all proposed goals are met, including all goals from the core requirements list, the advanced requirements list, and the stretch requirements list. In actuality, the advanced and stretch do not depend on each other and can be completed entirely independently, but we do estimate that if we were to attempt any of the stretch goals, we would have had at least some success with the goals on the advanced requirements list. The following are the changes between the advanced configuration shown in Figure 7.7.2 and the stretch configuration shown in Figure 7.7.3.

The first and most drastic change to the robot is the addition of a SLAM module between the pointcloud2 topic and the autonomous navigation code. We did not end up implementing the SLAM module, but it was something we were all very interested in, and even put a few months worth of effort into attempting. The SLAM module would create a new map of the environment every time the robot began its surveillance, and would continue to search for intruders while it roams. In the case that we were able to successfully implement SLAM, we would meet to consider whether to make any additional changes to the user interface or robot behavior, as this would drastically change the capabilities of LookSee's navigation.

One addition that may impact which version of SLAM to implement is that we may combine camera data with the LiDAR data to assist the mapping algorithm. Sensor fusion between LiDAR and cameras can be used to create more accurate maps more quickly. Our SLAM code, which would likely be modified from an existing package available through ROS, would be augmented by computer vision code that can identify landmarks. The camera data could help the car verify how far it is to the left and right from the wall in fewer frames than would be required with only the LiDAR. In a hypothetical far future version of the LookSee project, the robot could navigate towards a detected intruder.

The final addition to the stretch requirement configuration is the error logger. The error logger uses the system clock to detect when ten minutes have passed with no access to the internet. If such a situation occurs, LookSee will denote the loss of connectivity in a location accessible to the owner upon their return. While not constituting an emergency, if something were to happen during the time when LookSee could not get into contact with the human emergency contact, the fact that connectivity was lost during a certain time could be useful.

Overall, the completed stretch variation of LookSee, while still small and low-budget, is an extremely powerful piece of technology for the price. More than that, it demonstrates our competence in systems and software engineering. It demonstrates the interests of each member of our team, and shows that we are capable of implementing the features we wished to learn more about at the beginning of Senior Design. If we are able to complete this final version of LookSee, we would all be very proud.

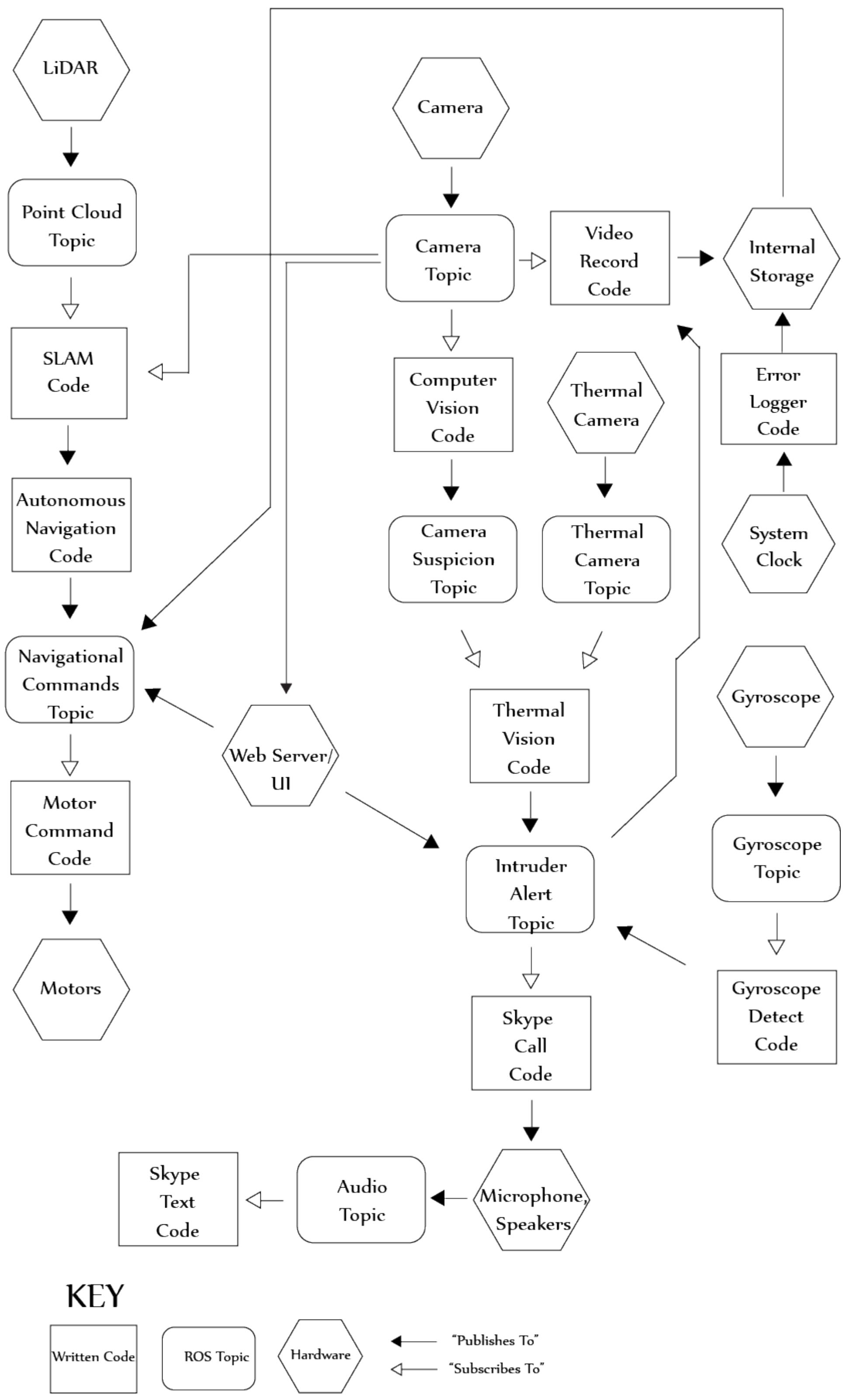


Figure 7.6.3: Stretch Requirements Software Diagram

8 Prototype Construction

Due to the fact that our team will be in low contact for the majority of the semester (due to COVID-19), we have carefully selected requirements and components that will involve as little physical construction or hardware modification as possible. For the majority of our team, the entirety of our responsibilities lie in programming software components and testing those components. However, balancing the construction of the robot and the development of the software will require advance planning and vigilance that the hardware is complete as early as possible and maintained throughout the semester. This section details our plan for construction of the entirety of LookSee.

8.1 PCB

For our printed circuit board we had to go through with a manufacturer to print our design since the equipment used is not something we have, nor does our university labs. Due to the current pandemic we are told that ordering any PCB's will be difficult. After reaching out to a few places JLCPCB was our best choice since they got us the PCB on time.

8.2 Chassis Modification

The chassis we have purchased is a model RC car with a plastic cover that is designed to look like the exterior of an automobile. To modify the chassis for our uses, we will first need to remove this plastic cover. Inside is a signal receiver, a motor controller, a motor, and a battery. We may decide to upgrade our battery in the future, but for now we will leave it as is. We will not be using the signal receiver, so we will remove it. However, the signal receiver originally passed signals to the motor controller. We will replace the signal receiver with a Teensy 3.2, which will now pass signals it receives from the Raspberry Pi to the motor controller. We will also need to attach all of the sensors to the chassis as well.

While the chassis can support the weight of these components, combined, they have quite a large footprint. We will likely need to arrange these components in layers using cut lexan or acrylic so that they can all fit on the robot. We will be designing these modifications once we receive the vehicle in the mail.

Once we have a design for the additional support structures that we will need to hold our sensors and processors, we will need to cut the acrylic or lexan so that we can use it. Our first resort is to use the laser cutter located in the UCF Innovation Lab to cut acrylic. However, we know that under the current conditions there is a possibility we will not have access to any of the university manufacturing resources. If this is the case, we will visit FamiLab, a makerspace located in Orlando, to use the laser cutter there. If FamiLab is also out of our reach, one of our members is in possession of a scroll saw, and will machine the lexan support structures by hand.

8.3 Coding Plan

The requirements of this project were carefully designed so as to facilitate four people doing four semi-discrete parts that can be later assembled to create a functional robot. This is because, due to the coronavirus, our team was unable to meet even semi-frequently for the duration of this project. The software setup and writing was done as follows:

Stavros Avdella possessed the chassis and nearly all of the physical components. He has been assigned to be our pseudo-electrical engineer, and as such, he designed the PCB. He was responsible for making all modifications to the existing chassis and assembling everything into a working prototype. Stavros also 3D printed the main body mount which held the Jetson Nano and the servo driver PCB. This design was publicly available as it was a piece from the donkey car project. He also designed and 3D printed a mount for the LIDAR. Stavros assembled the first initial prototype of the robot and passed it on to Tyler.

Tyler Wallace did the initial setup on the Jetson Nano which included installing Ubuntu on the Jetson Nano, setting up a ROS developmeent environment, and installing necessary packages. Tyler also worked to find a library for utilizing the servo driver PCB and wrote an interface so it would be easier for other to program the robots movement. He also wrote the user interface via a web dashboard and made sure all the necessary packages were installed and everything was working as intended. The main part of his contribution is utilizing a computer vision algorithm for person detection. After a semi-successful test setup with the Raspberry Pi, he implemented the NVIDIA's deep learning ROS nodes to be base of our object detection system. This would later be used by Jade's alert node to contact the user when a person is detected. Austin used the servo interface code when developing the find the gap algorithm.

Austin Pena was responsible for the thermal sensor and the LiDAR navigation. Austin, with the help of Jade, developed code that subscribes to the LiDAR point cloud topic and perform computations to calculate the robot's best next move. The algorithm he settled on after thorough testing is the 'find the gap' algorithm. Inspiration was taken from the UNC Chappel Hill code for the F1Tenth race competition. This code publishes steering and throttle commands to the servo interface node via the cmd_vel topic.

Jade Zsiros was responsible for helping Austin with the autonomous navigation code, and created the navigation logic visualizer. She also programmed the alert node which contacts the user in the event that a person is detected by the robot, and found, installed, and configured the necessary proprietary software to allow the robot to send SMS messages.

While we planned to have this happen separately, because many of the components relied on one another the robot was simply passed back and forth from team member to team member. When one of us needed help we were there to assist through an online call.

8.4 Prototype Construction Timeline

In Figure 8.4.1, we have outlined the timeline for completion of our Senior Design prototype. We have been careful to portion out the work so that each cell will take approximately a reasonable expectation of time in focused work, so that we will still have time for our other classes and for the rest of the work required for Senior Design 2. We have also done our best to design a timeline that will result in a completed product as early as possible, because we can be confident that things will happen that can push against the deadline. By making our own deadline early, we ensure the safety of the deadlines that we are actually graded on.

In the first week, we need to ensure we all have access to our tools and that those tools are in the condition we need them in. We will all install Linux and ROS onto our personal computers and ensure that the version we install is compatible with all of our hardware. A new version of ROS is released frequently, in fact, a new version was released during the Summer 2020 semester, so there is a very high probability that one or more of our sensors will be incompatible with a version of ROS we are considering. Once we have verified sensor output from each device, and have ensured we are all using the same version of ROS, we will be finished with our work for week 1.

In weeks two and three, we will be doing our respective most difficult tasks of the semester. We will have the most time during the beginning of the course, and so we have decided to put the tasks that will require the most attention and innovation here. The actual computer vision algorithm, the web page, the PCB, and the thermal detection code will be completed during this time. We will not attempt to integrate these components with ROS yet; the focus will be on writing and implementing reliable and well thought out solutions.

Once all of those building blocks are complete, in week 4 we will modify them so that they integrate with the rest of the robot. The computer vision code will stop outputting to the screen and start outputting to a ROS topic, which will instruct the thermal code on where to look. The thermal code will stop receiving instructions from the user and begin receiving instructions from the aforementioned ROS topic, and will output alert flags to the new (intruder_detect) topic. The user interface will publish its button presses to ROS topics as well. Any manufacturing that needs to be done to facilitate the attachment of our hardware components will be done during this time.

In weeks five and six, we will begin to diverge a bit. Hardware and Computer Vision will work together to integrate the team's progress in software onto the Raspberry Pi that will serve as the brains of LookSee. This will need to be done carefully, as the components will have never interacted in the past. The navigation algorithm, find-the-gap, will also be implemented during this time, but there is a possibility that the PCB will not have been received by then, which would preclude the ability to test this code throughout the process of its writing. Testing navigation code is of the utmost importance to its fidelity, so if the robot is not yet available, the code will be tested on a custom map in the Gazebo simulation software. This will ensure constant progress in all departments regardless of any outside factors. Finally, User Interface will implement the Skype call code during this time, and ensure that it is triggered by a flag raised in the appropriate ROS topic. The actual user interface will be used to test this, which will serve as a test for the UI as well.

Week seven will be dedicated to testing individual and combined components. User Interface and Computer Vision will work together to test that the computer vision algorithm correctly identifies humans, uses the thermal sensor to pass a flag to the Skype call code, and that a call is actually made. Hardware and Navigation will work together to stress test the navigation software, tune it, and ensure that all of the hardware is working as expected. Additionally, the last software component of the core configuration, the video stream on the UI, will be implemented.

In week eight, all four of us will meet as a group for the first time. We will spend a full day testing each component in as many configurations and orders as we can think of to ensure that no malicious interactions occur in the system. We will mark any issues or areas for improvement, and discuss how we would like to proceed for the remainder of the semester. If the navigation algorithm is not working particularly well, we may decide to focus on implementing SLAM to improve that area. If all functions of the robot seem to be working extremely well, we will focus on the advanced and stretch goals that impact no other functionality to preserve the progress we have made.

By this part of the semester, we will all have a better understanding of each of the components, and better-formulated opinions on how to improve our design. From week nine forward, we will focus on improving LookSee so that it is to our liking, but by this time we anticipate having completed the core goals we set for ourselves at the beginning of Senior Design 1. It is very likely that by this time, we will need to focus more on the website and final paper, as well as our other classes. This timeline should ensure that no matter what difficulties we face during the semester, the result is a complete product that functions as anticipated and is ready to be shown to our mentors and peers.

Week	Computer Vision	Navigation/Thermal	User Interface	Hardware
1	All: Install Linux and ROS on personal computer, verify functionality. Verify all parts have been received and are functional.			
2-3	Write a program that can identify a human in an image and output a confidence value.	Write a program that can check the temperature at a location in an "image" received from the thermal array.	Create a user-facing page for a web server with nine buttons, space for a camera feed, and appropriate graphics.	Design PCB and place order. Pay for express shipping.
4	Modify above program so that the location of the suspected person is published to a ROS topic.	Modify the above program so that it checks the location published by the computer vision program, and publishes a flag to a new ROS topic if 60% confidence is exceeded.	Modify code for buttons so that all buttons publish to their respective ROS topics.	Create additional layers on chassis to securely hold sensors and other electronics.
5-6	Work with Hardware to integrate all code the team has completed thus far into ROS on the Raspberry Pi that will be used on LookSee.	Begin implementing find-the-gap navigation and testing in Gazebo if hardware has not yet been received.	Write Skype call software.	Work with Computer Vision to integrate all code the team has completed thus far into ROS on the Raspberry Pi that will be used on LookSee.
7	Work with User Interface to begin testing intruder detection and alert features.	Work with Hardware to test and fine-tune autonomous navigation.	Work with Computer Vision to begin testing intruder detection and alert features. Implement UI video stream.	Assemble the robot. Work with Navigation to test and fine-tune autonomous navigation, test hardware.
8	All: Meet to perform all tests again, identify problem areas, strategize for the remainder of the semester, and decide which stretch goals to pursue.			
9-End	All: Fix issues that may have been identified, pursue advanced and stretch goals, perform tests.			

Figure 8.4.1: Stretch Requirements Software Diagram

8.5 Alternatives and Back-Up Plans

With our team's current situation, being separated and most of the manufacturing resources we would have previously had access to out of reach, there are a variety of things that could impede our progress. In case we are unable to contact each other or get our hands on any particular equipment, we have come up with the following alternatives.

Gazebo Gazebo is a robotics simulation software compatible with ROS. If there is a problem with the physical robot, or other members of the team cannot access it, we will construct a URDF (Unified Robot Description Format) file that can emulate our robot, download or construct a map in Gazebo, and run the code on this simulated version of LookSee. Obviously, the goal will always be to have a fully functioning, physical robot, but if PCBs take too long to manufacture, or an accident occurs with the hardware, Gazebo is a route we can take to ensure continued progress on the software while the hardware is brought up to peak performance.

Backup Car The majority of our components are very cheap, and most of us have access to Raspberry Pi. Because we will be separate for most of the semester, it is in our best interest to have a second vehicle with the cheapest components available to test code on and pass from team member to team member. Additionally, if anything were to happen to our primary vehicle, we would be able to switch to the backup vehicle relatively easily.

Orlando Maker Spaces During the current world situation, we are not certain that we will have access to all of the university resources we may have been used to, including the Innovation Lab and the machine shop. Should we lose access to any of these areas, we are prepared to pitch in for a membership to one of Orlando's four maker spaces, each of which is more than adequate to provide the tools we need to complete our project.

Camera-Only Navigation The most expensive and delicate component of our robot, and therefore the most difficult to replace, is the LiDAR. We also purchased the cheapest model we could find. If for some reason we were unable to use our current LiDAR unit - water damage, software incompatibility, or manufacturing defect, for example - we would not be able to simply acquire another identical component on our current budget, especially if it turns out there is a problem with the actual sensor and another model would need to be selected. However, without LiDAR, all hope is not lost. There are a variety of simple algorithms that can be used to navigate with only a camera or other cheap, easy-to-find components.

- **Line following** - Using an additional camera pointed near the ground, we could use painter's tape or something similar to delineate a path for the robot to travel. Simple code could identify where the robot is in relation to the line and keep the wheels centered over it.

- **Navigation by icon recognition** - With clear, easily identifiable symbols such as QR codes, we could mark turns in the path that the robot needs to take. The robot would use the camera to identify these icons.

- **Wall following** - While this hypothetical assumes we cannot get another LiDAR unit, he cost of individual single-point ultrasonic sensors is in the single digit dollars. With three ultrasonic sensors, one pointing ahead, one pointing to the right, and one pointing to the left, we can have the robot follow the left and right walls at random, and avoid running into things in front of it. While this isn't the most sophisticated algorithm in the world, it is incredibly cheap to implement, and serves as an excellent alternative if we lose use of the LiDAR.

8.6 Block Diagram

Below is the block diagram that details the connectivity of the robot elements and the responsibilities of each team member.

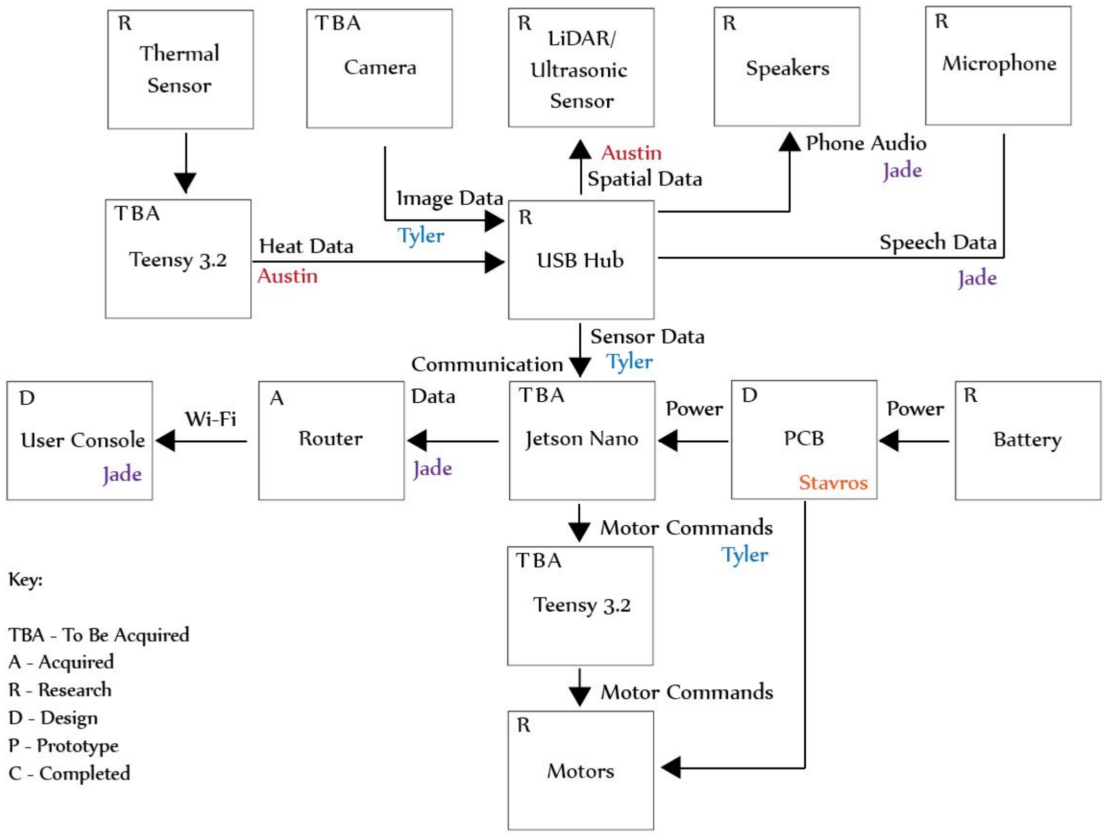


Figure 8.6.1: Team Responsibility Block Diagram
(Graphic created by Jade Zsiros)

9 Prototype Testing

Once we got our components and have started to program, we began testing. We have outlined specific requirements to verify the completion and effectiveness of each of the components. The majority of our tests were completed as soon as a component has been completed, and repeated until the component passes, but a small number of these tests required multiple parts relying on each other's completion. For our tests, we had to wait until nearly the end of the semester to run, as the four members of our team are not in frequent contact due to the current remote education situation.

9.1 PCB

Once the PCB design has been designed and made, we ran test to make sure it can do what it is intended. We did this by wiring it up to our Jeston and motors and running a command in our prompt to detect for our servo driver and its values. Everything checked out perfectly however we ran into a PCB power issue. The PCB was only able to give enough power to one motor at a time. We troubleshooted and discovered we needed an external 5v power coming into the PCB. Because of the delay in manufacturing we were unable to remake and reorder a new PCB.

9.2 Hardware Testing

The majority of our efforts will be put towards the completion and effectiveness of our software design and autonomous robot functionality. However, it is still vitally important that our robot's hardware, including that which we purchased off the shelf and did not modify, is entirely functional and achieves the goals which we have set. Below is listed our procedures for testing our hardware.

9.2.1 Chassis

We ordered a 1/16th scale four wheel drive RC Car from Exceed-RC. We tested that the battery lasts as long as is advertised, and checked that the motor controller is accessible to us.

We knew right away we would not be using the stock frame that is made out of plastic to make the chassis look like a truck. Upon removing that we were left with 4 coilovers stick out that we used as mounting points. Due to the situation we don't have access to a 3D printer, so we ordered a 3D printed frame from the Donkey Car website. We also drilled out holes to allow our WiFi antenna to go through.

After doing this we tested to make sure the robot still ran with no issues and nothing in the way of the wheels and wiring. Since this frame was built for our RC car we knew coming in we would have no issues with it.

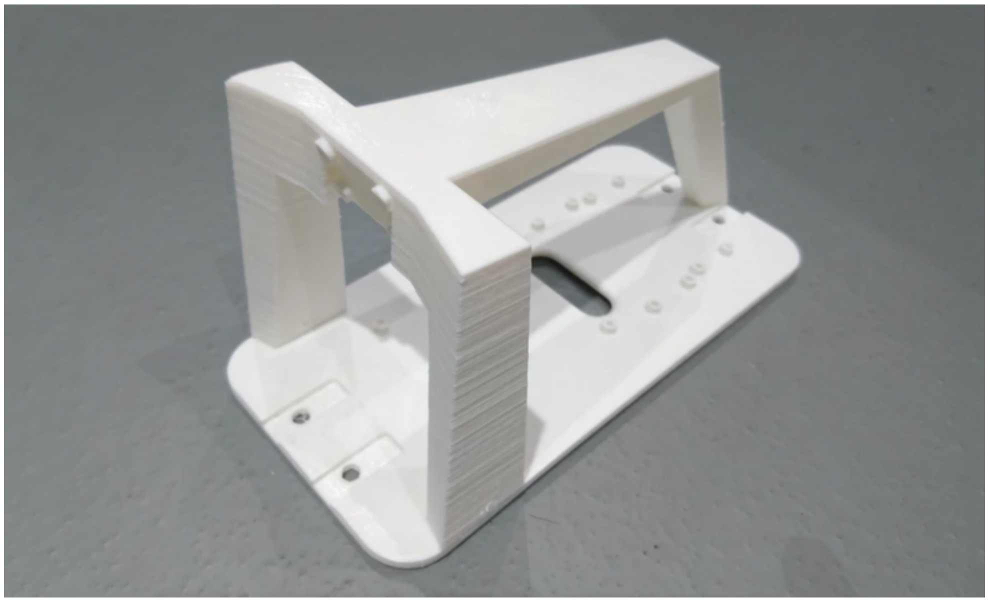


Figure 9.2.1: Donkey Car Chassis

9.2.2 Wheels

We used the wheels that come with our RC car that we purchase, They are made of rubber and do not require any air as some other models do. We tested them by putting a ten pound weight on the car to make sure it can still move under load and not slip. Although ten pounds is a very extreme weight we know that any load under ten pounds will be perfectly okay. Of course the less weight the better as we will save power and wear on the tires.

We also tested the robot's driving on both carpet and tile. As we do not know where we will be demonstrating our project, it is important that it performs predictably on all indoor flooring surfaces.

9.2.3 Power Distribution

A serious risk with developing robots is that the power draw at high speeds can cause the processors to lose power and the robot to restart. This would be a massive problem for our robot. To test this, we drove the robot at high speeds for as long as possible and ensure that the robot remains functional past our expected time limit.

After a few tests we noticed a huge power draw whenever the robot was in motion causes the Jeston to shut off due to the lack of power. We ended up going with our second plan, which was to implement a portable battery to run the Raspberry Pi, while the RC car battery only runs the RC car. We did this by mounting the portable battery under the base plate. It is held up by zip ties that we fed through holes we drilled into the base plate. We made sure that we placed the battery in the position it is as seen in the image below to sure the power button is accessible, as well as all the ports being accessible from the side.

The RC car's battery will be using is a NiMH 5000mAh running at 7.2 V. We were able to drive the car on full charge for over 40 minutes with no power loss. Even after multiple charge cycles we noticed the battery did not degrade, and we were fine to continue using this battery.

Once installed we made sure there was enough clearance between the portable battery and the RC cars electronics. We did notice a significant drop in the RC car due to the weight, but we knew this would happen and the RC car is still functional with no restrictions.

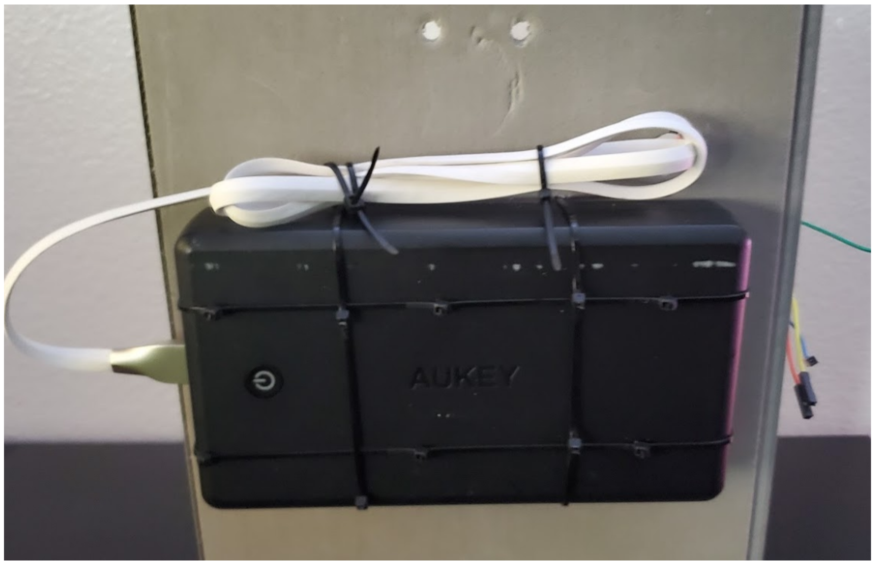


Figure 9.2.2: Portable battery mounted under base plate

9.2.4 Sensors

Once we receive the sensors that we ordered, we immediately tested them to ensure that they output data at the rate and quality that was advertised. Next, as soon as we have ensured that the sensors were correct and working, we integrated them within ROS, as a test to ensure that their open source ROS modules were functional and compatible with the rest of the software versions we were using.

9.2.5 RPLidar

Due to the environmental processing being depending on Lidar it's integral to the functionality of LookSees autonomous navigation that the output of data matches the rates advertises within the accuracy detailed in the data sheets to ensure that our model isn't defective and were ready to be assembled and used for information collection to be processed by our Jetson Nano. We ensured that the data being collected by the RPLidar was accurately reflecting the environment of what LookSee observed matched the present environment which LookSee was in.

9.2.6 PCA9685

After setting up our Raspberry Pi and installing all the packages we needed, we started to wire up our PCA9685 Servo Driver following the diagram block diagram below.

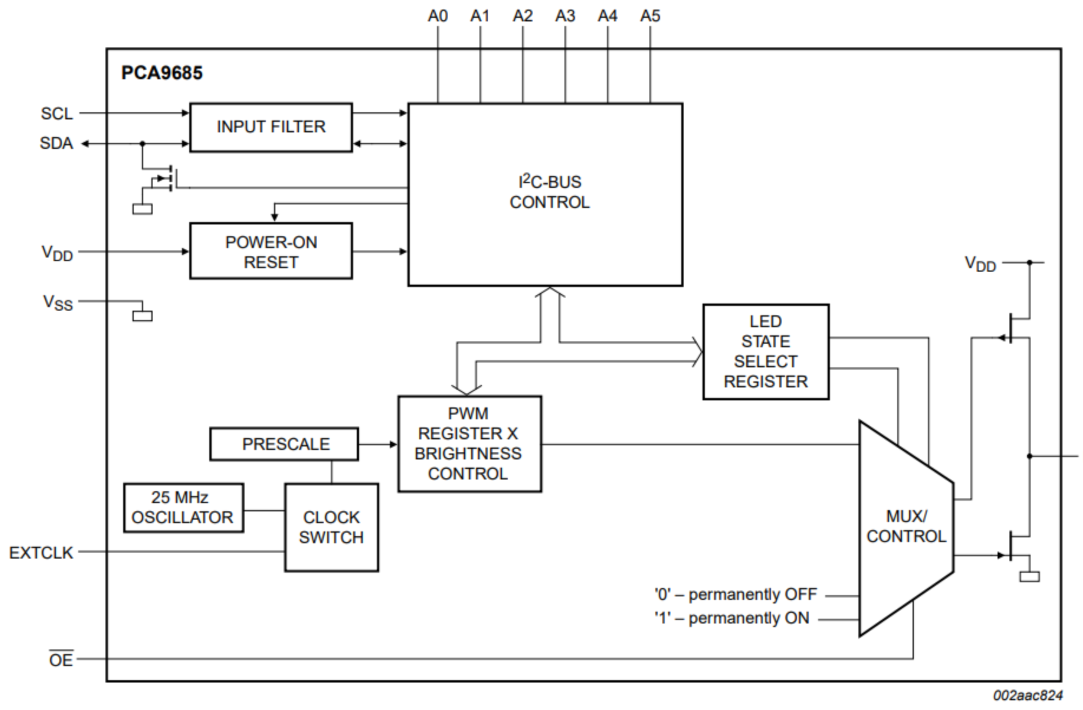
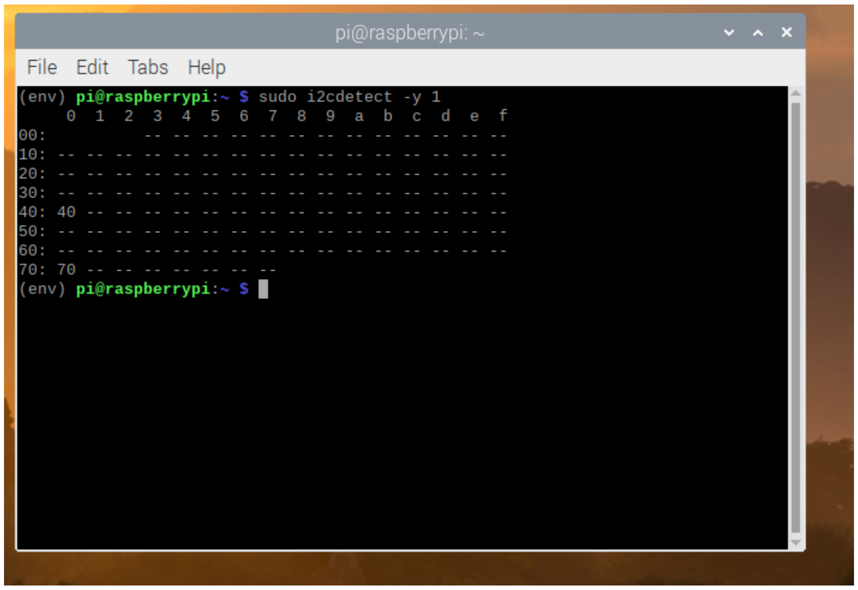


Figure 9.2.3: PCA9685 Block Diagram

To insure that the Raspberry PI is detecting it we ran a line of code in the command prompt and it should give back a grid of addresses with the number 40 as the address of our PCA9685 board. We ran a simple command to get servo and motor to run, and as expected it worked. We were then able to edit this to be applied to the Jetson Nano and it worked as described above for the Pi.



```
(env) pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
(env) pi@raspberrypi:~ $
```

Figure 9.2.4: Testing if Servo Driver is detected

After our test, we wanted to be able to mount our servo driver on our RC car and be able to run wires for the motors to the controller. We did this by drilling a hole through the base plate next to spot we plan to place our servo driver. We drilled 4 holes for the mounting screws and then feed the wires through the hole we drilled as seen in the image below.

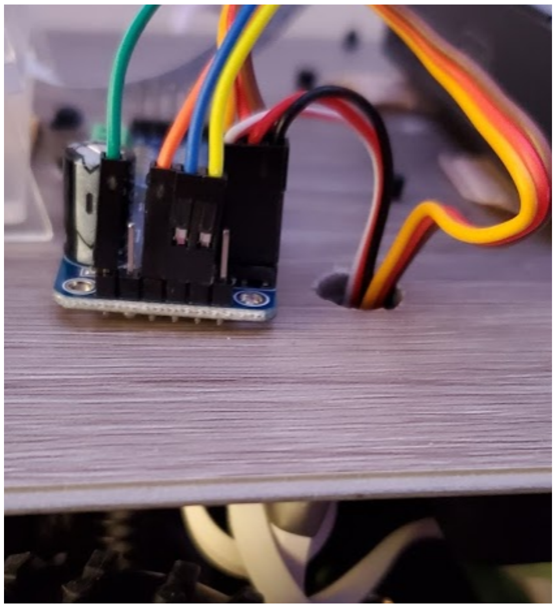


Figure 9.2.5: Wiring of Servo Driver

9.2.7 Microcontroller

Our Teensy 3.2 will need to forward all of the motor commands for the robot's movement. Once it is programmed, we will test it by running the autonomous code on the robot. If the directions are forwarded correctly, the robot will move.

9.2.8 Microprocessor

The failure mode for the Raspberry Pi would be if there are too many computations running on it and the Pi begins to freeze. Once all of our code has been written, we will test running all of the code at the same time. If the robot does not start to lag, we will know that we have stressed it no more than necessary.

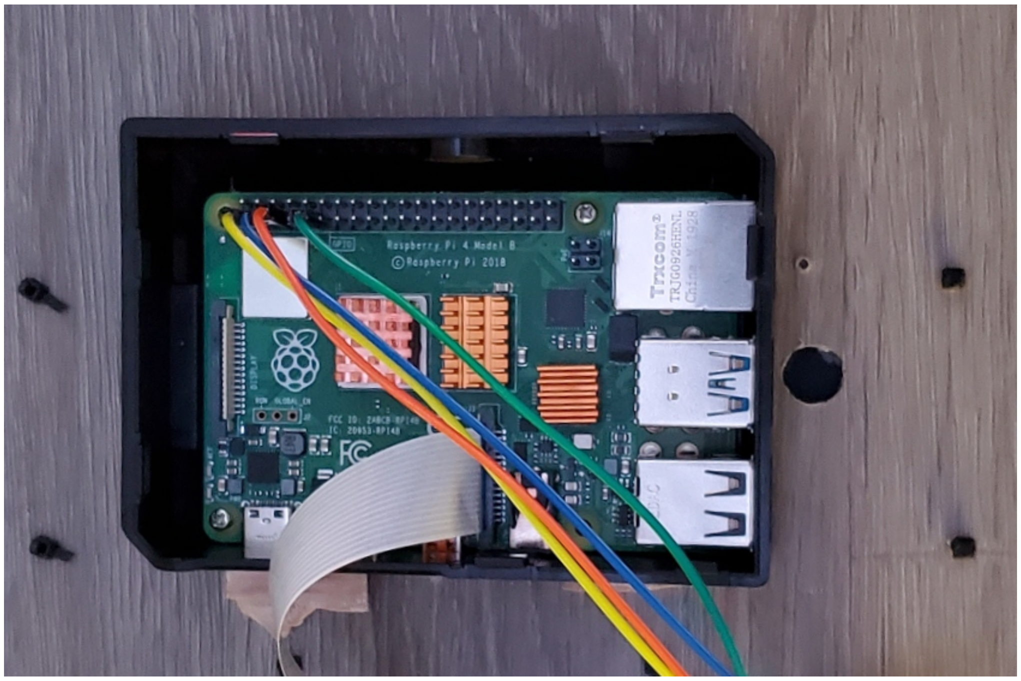


Figure 9.2.6: Raspberry Pi 4 mounted on base plate

9.2.9 Hardware Integration Testing

As the most important integration of hardware to software is the forwarding of the motor commands from the Raspberry Pi to the Teensy 3.2 to the motor controller, the "Microcontroller" section above describes what will serve as our hardware integration testing.

9.3 Software Testing

As we are all computer engineering students, we decided early on that the software will be the focus of our efforts on this project. It is of the utmost importance to us that the software is functional and demonstrates the abilities we have developed in programming since the start of the pursuit of our Computer Engineering degrees here at UCF. The below sections detail how we will test our software.

9.3.1 Graphic User Interface

The user interface must be able to publish messages to the navigation and intruder detection topics when the joystick is manipulated on the dashboard. This can be verified with the 'rostopic echo cmd_vel' command, which, if run on an open and separate Linux terminal window on the robot while a user manipulates the joystick, this will show the publishing of the directional commands to ROS from the web server. Verification that the correct values appear in the (cmd_vel) topic when the joystick is moved in all directions and that the proper sections of the message, linear for throttle and angular for steering, are corresponding to the intended directions.

9.3.2 Servo Interface

The servo interface is the node that allows any node to publish a Twist message to it containing linear and angular values in the range of -1 and 1 and converts those messages into direct PWM signals that are then sent to the servo driver. It also contains a timeout method in case the controlling node gets disconnected in order to prevent a run-away robot.

To verify this functionality at the start, manual Twist messages can be published to the cmd_vel node via the command line. This will be good for finding out the actual limits of the servos. Once we are confident that they are working as intended we can connect the UI joystick to the servo interface node and make sure the values received from the joystick also are being processed as intended.

9.3.3 Phone Call Software

The Twilio Developer API allows programmers to write code that can send text messages and make phone calls. LookSee's software, if an intruder is detected, attempts to start a video call with the emergency contact by texting them a meeting invite link. LookSee's intruder detection software is put to sleep for two minutes, and if the intruder is detected again after that, the meeting invite link is sent repeatedly until the user joins the meeting, at a maximum of once per two minutes.

9.3.4 Camera Human Detection

The camera must be able to detect whether a human is in a photo at least 60% of the time. We do not anticipate needing this to be higher precision, as the suspicion will be verified by the thermal sensor. (In the final iteration, we ended up disregarding the thermal sensor in favor of better traditional detection through the camera. Our detection rate was roughly close to 90% which justified this decision.) This process will be accomplished through computer vision-based object detection, which is composed of three primary goals: picking objects from the background images, identifying the object using a probability score, and defining the boundaries of the object and the origin, including the object's dimensions, and its distance relative to the camera in the environment.

The identification of the objects is done through a systematic division of a picture, using algorithms to identify potential objects of interest. The algorithm then proposes the obect it has discerned to be an object of a particular class (eg. a person, a fire hydrant, a crosswalk). The final steps of the process are then to return the class of the object as well as the location of the object in the frame or image provided to the machine learning algorithm. In more detail, the identification of an object, in our case humans, would be the class we're looking to identify and the algorithms in the visual field providing frame data would first need to be fed processing blocks.

Processing blocks are trained by looking at thousands of images of a particular object to develop a well rounded idea of how the object looks from nearly every angle and in a vast number of differing background environments. These processing blocks are called models and can be trained to identify nearly any particular object. This idea of object identification is primarily based around classifying an object using a model, which then takes in a raw image, and based on its training returns a proposed object class.

An important aspect we mentioned earlier was the probability of detecting a human presence in the photo or frame, and these models return a probability value depending on the model's confidence of identification of a particular object within a frame. It extends from this idea that the accuracy of the identification is built on the data set and level of training the model has received, in addition to the type of analysis being performed on a frame of video, and the quality of the frame of the video (with a poor quality image, it will be harder to distinguish objects within the frame.

In addition, a commonly overlooked quality important to the identification of objects within an image is the size of the object relative to the total image size. (eg. If a person is to be identified, but they are miles away and microscopic in the image, it will be incredibly difficult to discern them from the environment regardless of how well trained the model is.)

The idea of recognition is a multifaceted concept with several approaches, and through a use of a conjunction of tools we're better able to utilize the information readily available from computer vision. If we were to want to detect humans as they are in a store, object detection would model both the environment as the individual objects within a designated class (like item shelf, or bathroom door, or front counter) and the object class of people; from this you could see which portions of the store are utilized by people. Similarly it follows in our use case if we were to develop (as a stretch goal) the ability to discern the aspects on the environment, potentially in a break LookSee could ascertain that the front window was broken into as it appears amiss with a very low probabilty to the model.

9.3.5 LIDAR Environment Processing

The LiDAR unit we purchased has a a ROS node built and supplied by the developer that published the data measured to a ROS topic. The 2D point cloud (sometimes referred to as a point line) is composed of stream of coordinates composed for degree and radius from the most recent laser scan. The software would then use simple geometry to evaluate the environment LookSee is currently traversing. With an abundance of time perhaps we could have implemented the SLAM algorithms as a relatively far stretch goal. This, if possible would have required some form of telemetry to increase the quality of mappings and traversal within it. Depending on the implementation chosen in this stretch goal case where we perform simultaneous location and mapping the telemetry data may need to be provided via accelerometer, or a similar sensor. Potentially telemetry could be derived from the scan data and the way it's changing based off scan to scan, though the system is entirely observable then perhaps looking at the subsystem responsible for motor controls could provide telemetry. Ultimately, with a more reasonable approach being Find The Gap navigation; or even in the most simple case where LookSee performing some Follow The Wall algorithms to navigate autonomously whilst performing Camera and Thermal Human Detection. The Lidars data is instrumental in all forms of behavioral navigation, but in order for those to have the data to operate on the environmental data must be correct as well. Testing of the device will ensure it's accurate to the specified data of whichever Lidar device is purchased.

9.3.6 Behavioral Navigation

We verified the behavioral navigation code by allowing LookSee to navigate a garage where in we would place obstacles in the path, or walls for the robot to follow. At first we had to have motor commands implemented before graduating to behavioral navigation to ensure that the sequence of control is in place and we could effectively eliminate certain errors from suspicion by process of elimination and knowing each prior step is implemented effectively. After the method of behavioral navigation was implemented the test space was be composed of an empty garage at first, and gradually increasing the difficulty of navigation required by LookSee to see the widest possible range of use cases emulating all possible environments that the robot may encounter through various courses developed within the garage through use of cardboard/particleboard structures to obfuscate and force autonomous traversal and navigation.

9.3.7 User Controlled Navigation

We will verify user controlled navigation by pressing buttons on the user interface and ensuring that the robot performs the correct movement for each. This will include bug testing and constant use of the navigational buttons throughout the development of the entire project as this will be the initial steps in movement after motor commands are in functioning order. This way, in the case of failure or resetting the robot, the team won't need to traverse an obstacle course in order to retrieve LookSee, and, it ensures that these obstacle courses are designed within reason. If a human driver can't traverse the course, we can hardly expect behavioral navigation to stand a chance.

9.3.8 Motor Commands

Integral to the testing of LookSee is functioning Motor Commands, and will likely be controlled through servo driver PCB. In order for any movement to occur, autonomous, or user controlled the development of motor commands must occur. The functional requirements of several other components hinge on the reliability of our motor commands effectively carrying out their instruction and will be tested throughout the life cycle of the product through the various other stages above ensuring functionality across all states. The original PWM signals sent by the RC car were measured and recorded via a oscilloscope. Then when sending those same signals from the Nano, we made sure the max and min values of the PWM signal matched the originals. These commands were sent by a simple library that was not integrated into ROS at all and instead was just a Python script with the sole purpose of outputting to the servo driver.

9.3.9 Software Integration Testing

We will verify that all software is properly integrated by placing the robot in a loop and allowing it to drive autonomously. We will then check that the user interface is operating correctly, and that the teleoperated controls work as intended. Obstacles can be placed in front of the robot to make sure it can successfully navigate its environment. We then placed a person in front of the robot and verify that the robot recognizes the threat and calls the emergency contact. If all of these functions work, the robot software will have been verified.

9.4 Anticipated Failure Conditions

We have chosen algorithms and electronic products that we believe to be able to perform in a wide variety of situations, since we do not know what our eventual testing environment will be. However, there are a few situations we can anticipate being probable failure conditions for LookSee, and we will need to either compensate for them or request accommodation during our project's eventual demonstration.

Low Light and Busy Backgrounds Our vehicle relies on sight to identify human intruders. As we anticipate demonstrating our project in a new environment, it will be important that there is more than adequate lighting in the area where we will be demonstrating, and that the walls do not have any confusing details, such as a brick surface, paintings, or plaques.

Large Circular Areas The find-the-gap algorithm relies heavily on straight paths and turns that are relatively uniform in width. If a path opens up to a much wider area, with certain geometry, the algorithm may decide that the wisest choice is to simply turn around and go back. We may be able to write additional code to compensate for this, but in general, it is important that the path LookSee is asked to drive on is as similar to the path it was tested on as possible.

Outdoors LookSee relies on LiDAR technology in order to navigate, but inexpensive LiDAR does not work reliably outdoors. The sunlight that is pervasive even in shadows would interfere with the relatively weak lasers that are used by our particular model of LiDAR. Solving this issue with a better LiDAR sensor would at least double the final cost of the project, so it is not a problem we intend to attempt to solve.

Internet Dead Zones Although we have no doubts our robot will continue to operate in the event of a internet outage. However, if this was to occur the robot would be unable to send an alert message to the user if there is an intruder. It it very crucial that the entire room of surveillance has a decent and stable connection in the internet at all times. We have the error logging function as a stretch goal for exactly this reason, but for the purposes of demonstrating our product, simple error logging would not replace the emergency contact function this robot's concept is built on.

Stairs Potentially LookSee's biggest weakness, our project has a massive vulnerability to stairs and any other sudden drops in the floor. Because our LiDAR can only scan in two dimensions, returning a slice of the map at the robot's eye level, there is absolutely no way for it to detect that floor may not be available directly in front of it. A future iteration of LookSee could implement an additional ultrasonic sensor placed at the front of the robot, but for now, we will simply have to ask to be able to demonstrate our robot in first-floor hallways or cover openings to stairs with a large piece of cardboard.

Bodies of Water As an extension of the problem LookSee will face with stairs, our robot will be entirely unable to detect water features that are sunk into the ground level of its patrol area. If there is any bodies of water in a room that are not gated off such as in ground pool, hot tubs, water features, or even ponds, our robot will not be able to run its surveillance course. Any of these bodies of water will not be detected and will cause our robot to drive straight into them and completely ruin all of our electronics.

10 Project Operation

In this section, we will detail how to start up the robot from shutdown and begin all nodes. The following instructions need to be completed for both autonomous and manual navigation. While this process is not ideal we do believe that in future iterations we could minimize this process to a simple switch.

Start by plugging in either the robot's power supply or battery, and plugging a mouse, keyboard, and monitor in for your own use.

Once the robot has booted into Ubuntu, start ROS with the 'roscore' terminal command.

From here, follow the instructions for either autonomous or manual navigation.

10.1 Autonomous Navigation

Start by plugging in either the robot's power supply via a battery. After some time you can access the Jetson via your favorite SSH terminal. The static IP address used for the Jetson is 192.168.0.168. The password to the Jetson is 'jetson12345'.

Once connected to the robot you start ROS with the 'roscore' terminal command. This will start the base ROS program that all nodes connect to.

To give the LIDAR write permissions, run the following terminal commands in a new window: 'ls -l /dev —grep ttyUSB', 'sudo chmod 666 /dev/ttyUSB0'.

Then, to start the robot bringup, run 'cd catkin_ws', 'source devel/setup.bash', 'roslaunch robot_bringup robot_bringup.launch' in a new terminal window. All of the robot's functions start from this.

To begin autonomous navigation, run 'cd catkin_ws', 'source devel/setup.bash', 'roslaunch laser_values laser.launch' in a new terminal window.

10.2 Manual Navigation

Start by plugging in either the robot's power supply via a battery. After some time you can access the Jetson via your favorite SSH terminal. The static IP address used for the Jetson is 192.168.0.168. The password to the Jetson is 'jetson12345'.

Once connected to the robot you start ROS with the 'roscore' terminal command. This will start the base ROS program that all nodes connect to.

Once 'roscore' has completed, open a new terminal and enter the commands 'cd webpage_ws', 'python -m SimpleHTTPServer 7000'. This will start the web server.

To navigate to the web page from another computer, type the robot's IP address into the URL bar followed by ':7000'. The web page will load, and you can use the controls there to control the robot after robot bringup.

To start the robot bringup, run 'cd catkin_ws', 'source devel/setup.bash', 'roslaunch robot_bringup robot_bringup.launch' in a new terminal window.

11 Administrative

The schedule of the project and budget changed dramatically compared to what was proposed in Senior Design 1. This was due to shipping issues with certain parts, the ESC getting burnt out in the middle of testing, and COVID restrictions.

11.1 Project Milestones

Below is a time line of completion of each of our major project milestones and the date they were completed.

Senior Design I	Started	Completed	Status
Familiarize ourselves with project	05/18/2020	05/29/2020	Completed
Role Assignments	05/18/2020	05/29/2020	Completed
Identify Parts	05/29/2020	06/12/2020	Completed
Divide and Conquer	05/18/2020	05/29/2020	Completed
Updated Divide and Conquer	05/29/2020	06/05/2020	Completed
New Assignment on Standards	06/05/2020	06/26/2020	Completed
60 page Draft Documentation	06/26/2020	07/03/2020	Completed
100 page Final Report Documentation	07/03/2020	07/17/2020	Completed
120 page Final Report Documentation	07/03/2020	07/28/2020	Completed
Senior Design II			
Critical Design Review	08/31/2020	09/17/2020	Completed
Middle Term Demo	08/31/2020	10/10/2020	Completed
First Entire Prototype Built	10/12/2020	10/16/2020	Completed
8 Page Paper	11/06/2020	11/20/2020	Completed
Final Presentation	11/06/2020	11/29/2020	Completed
Project Website	11/13/2020	11/29/2020	Completed
Project Showcase	11/29/2020	12/02/2020	Completed

Table 20: Project Milestones

11.2 Project Budget

Rather than having one member of the team pay for the majority of the project and take it home, we deconstructed the project at the end of Senior Design II.

We all purchased components that we intend to take home at the end of the project, with each member's contribution being approximately equal. For components that cannot reasonably be salvaged at the end of the project, we ended up splitting the cost equally among all four members. Members who donate their own personal hardware and supplies took back their donation at the end of the Senior Design 2, after we have presented our final project.

Our budget primarily came from the following sources, but not limited to:

- Donated items
- UCF Robotics Club
- Our own money

We asked the manufacturers of the components that we are using to sponsor us or provide us with a discount, but we were unsuccessful and had to pay for the components out of pocket with sponsorship or discount.

Our aim is for the project was to cost no more than 200 dollars to each member of the team. As our objective is to make this a low cost robot. We were able and willing to spend any extra if needed due to the possibility of something breaking or not working, but we managed to stay under out goal which was a huge success.

As we noticed and were told, sponsorship and other form of funding for our project will be hard to get due to the current situation with the pandemic. We were intended to meet with companies to request funding but all of those had to be canceled. We had also experienced some difficulties with the electronics market as another result from the pandemic. Many companies have been closed and a lot of parts and products that we need are unavailable or have been selling at a much higher price than usually.

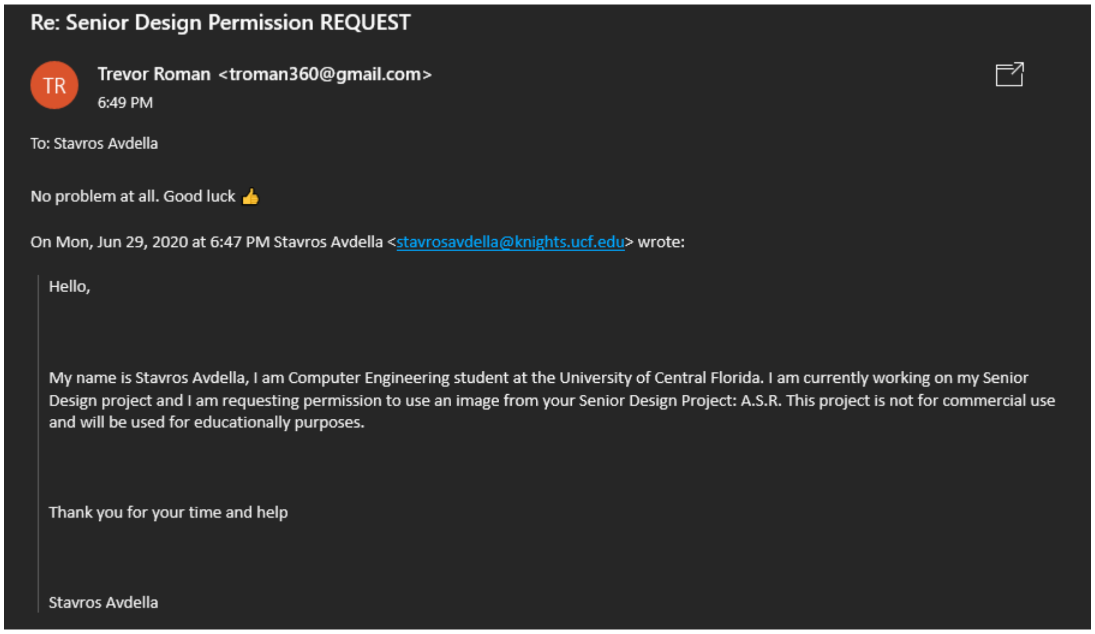
Item	Quanity	Vendor	Donated	Cost with Qty.
RC Car	1	Exceed-RC		$119.95
Donkey Car Frame	1	Donkey Car		$49.99
RC Car Screws	1	Exceed-RC		$9.99
RC Battery	1	Exceed-RC		$0.00
RC Battery Charger	1	Exceed-RC		$0.00
Power Bank	1	Anker		$49.99
Nvidia Jetson Nano	1	Nvidia		$99.99
USB to Barrel	1	SIOCEN		$7.99
Servo Driver	1	SunFounder		$9.99
Logitech Web Cam	1	Logitech	Yes	$49.99
Screws	1	Kuman		$8.00
USB Sound Card	1	Roccat		$19.99
LiDAR	1	RPLIDAR		$99.00
Twilio Phone Number	1	Twilio		$21.00
PCB	1	JLCPCB		$47.95
Micro SD Card	1	SanDisk		$9.99
RC Battery	1	Venom		$49.99
Wiring/Sleeving	1			$19.99
		Total		$673.79
		Donated Parts		$49.99
		Net Total		$623.80

Table 21: Cost of Project

A Appendices

A.1 Copyright Permissions

Figure 3.1.1: A.S.R Permission Obtained



Re: Senior Design Permission REQUEST

Trevor Roman <troman360@gmail.com>
6:49 PM

To: Stavros Avdella

No problem at all. Good luck 👍

On Mon, Jun 29, 2020 at 6:47 PM Stavros Avdella <stavrosavdella@knights.ucf.edu> wrote:

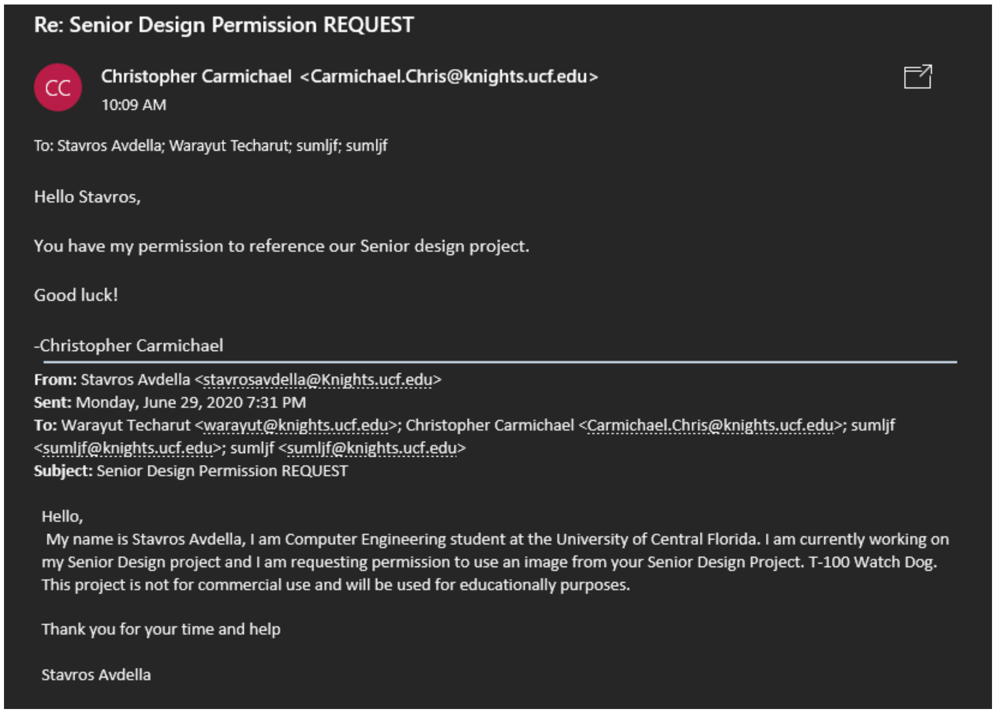
Hello,

My name is Stavros Avdella, I am Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use an image from your Senior Design Project: A.S.R. This project is not for commercial use and will be used for educationally purposes.

Thank you for your time and help

Stavros Avdella

Figure 3.1.2: T-100 Watch Dog Permission Obtained



Re: Senior Design Permission REQUEST

Christopher Carmichael <Carmichael.Chris@knights.ucf.edu>
10:09 AM

To: Stavros Avdella; Warayut Techarut; sumljf; sumljf

Hello Stavros,

You have my permission to reference our Senior design project.

Good luck!

-Christopher Carmichael

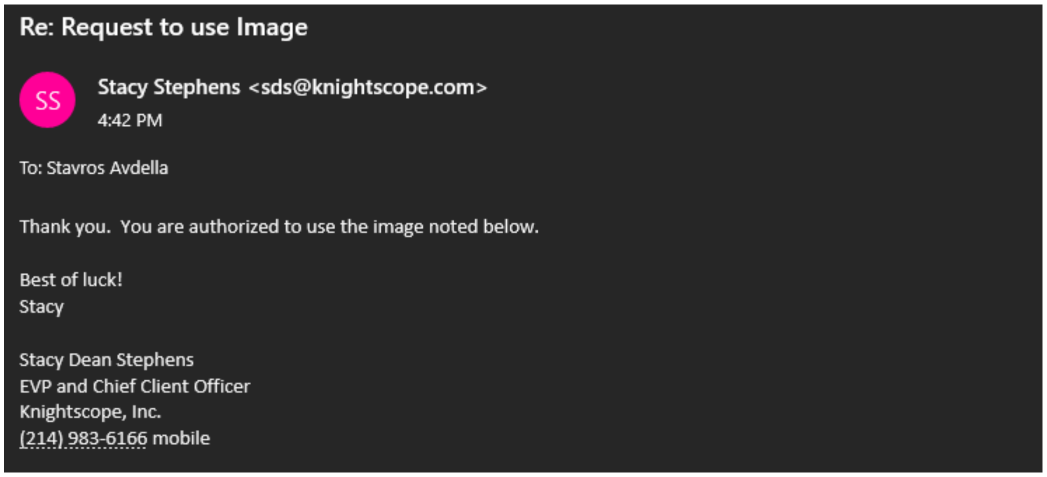
From: Stavros Avdella <stavrosavdella@Knights.ucf.edu>
Sent: Monday, June 29, 2020 7:31 PM
To: Warayut Techarut <warayut@knights.ucf.edu>; Christopher Carmichael <Carmichael.Chris@knights.ucf.edu>; sumljf <sumljf@knights.ucf.edu>; sumljf <sumljf@knights.ucf.edu>
Subject: Senior Design Permission REQUEST

Hello,
My name is Stavros Avdella, I am Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use an image from your Senior Design Project. T-100 Watch Dog. This project is not for commercial use and will be used for educationally purposes.

Thank you for your time and help

Stavros Avdella

Figure 3.1.3: Knightscope K3 ASR Permission Obtained



Re: Request to use Image

Stacy Stephens <sds@knightscope.com>
4:42 PM

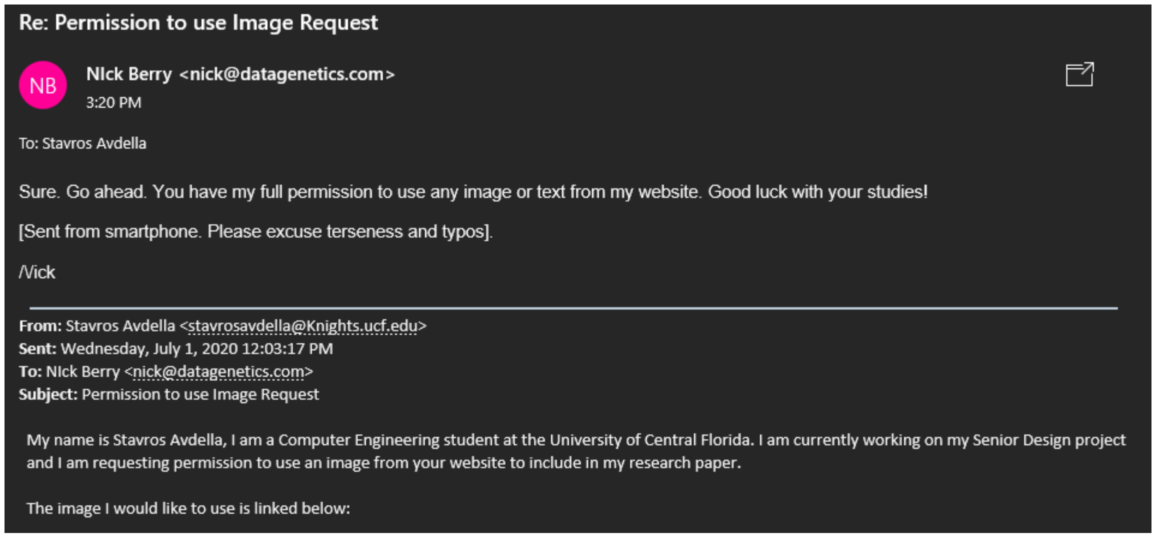
To: Stavros Avdella

Thank you. You are authorized to use the image noted below.

Best of luck!
Stacy

Stacy Dean Stephens
EVP and Chief Client Officer
Knightscope, Inc.
(214) 983-6166 mobile

Figure 7.2.2: Differential Drive Permission Obtained



Re: Permission to use Image Request

NIck Berry <nick@datagenetics.com>
3:20 PM

To: Stavros Avdella

Sure. Go ahead. You have my full permission to use any image or text from my website. Good luck with your studies!

[Sent from smartphone. Please excuse terseness and typos].

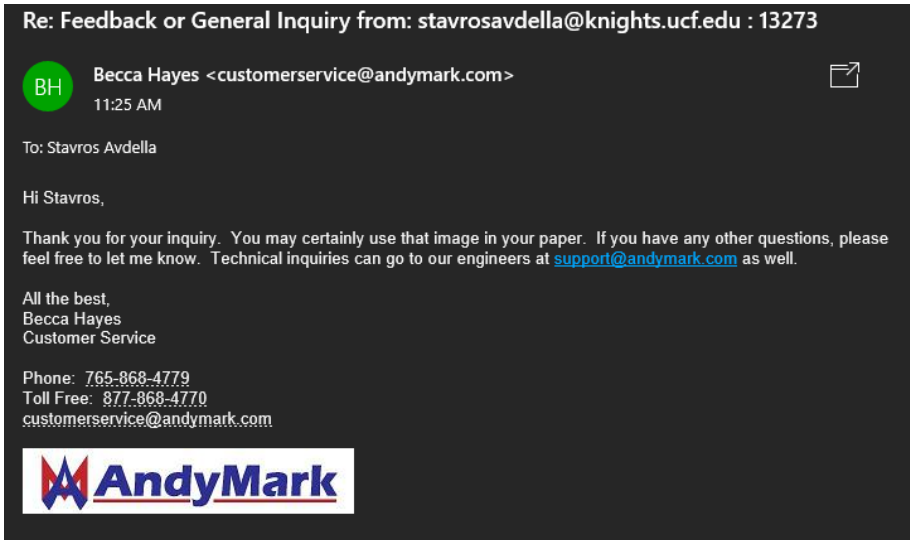
/Nick

From: Stavros Avdella <stavrosavdella@Knights.ucf.edu>
Sent: Wednesday, July 1, 2020 12:03:17 PM
To: NIck Berry <nick@datagenetics.com>
Subject: Permission to use Image Request

My name is Stavros Avdella, I am a Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use an image from your website to include in my research paper.

The image I would like to use is linked below:

Figure 3.13.2: Controllable Caster Wheel (Permission Obtained)


Re: Feedback or General Inquiry from: stavrosavdella@knights.ucf.edu : 13273

Becca Hayes <customerservice@andymark.com>
11:25 AM

To: Stavros Avdella

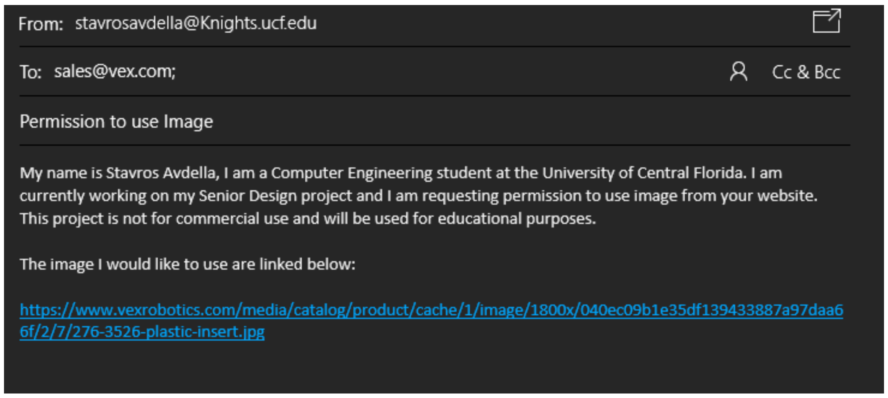
Hi Stavros,

Thank you for your inquiry. You may certainly use that image in your paper. If you have any other questions, please feel free to let me know. Technical inquiries can go to our engineers at support@andymark.com as well.

All the best,
Becca Hayes
Customer Service

Phone: 765-868-4779
Toll Free: 877-868-4770
customerservice@andymark.com

Figure 3.13.3: VEX OMNI-Wheel (Permission Pending)


From: stavrosavdella@Knights.ucf.edu

To: sales@vex.com;

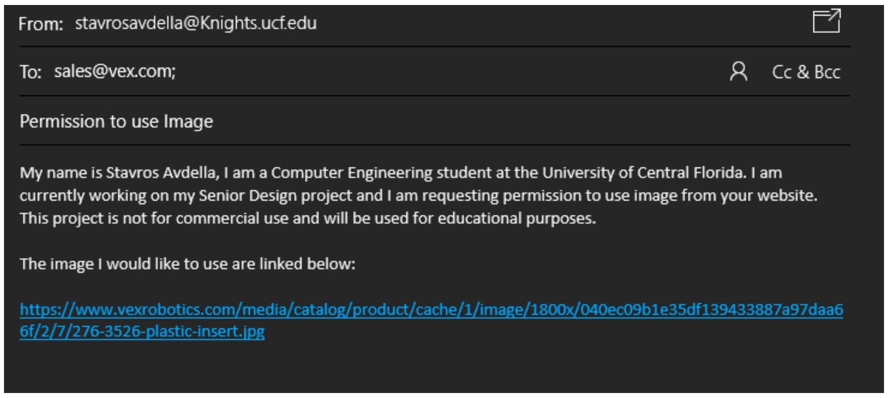
Permission to use Image

My name is Stavros Avdella, I am a Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use image from your website. This project is not for commercial use and will be used for educational purposes.

The image I would like to use are linked below:

https://www.vexrobotics.com/media/catalog/product/cache/1/image/1800x/040ec09b1e35df139433887a97daa66f/2/7/276-3526-plastic-insert.jpg

Figure 3.13.4: Mecanum Wheel (Permission Pending)


From: stavrosavdella@Knights.ucf.edu

To: sales@vex.com;

Permission to use Image

My name is Stavros Avdella, I am a Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use image from your website. This project is not for commercial use and will be used for educational purposes.

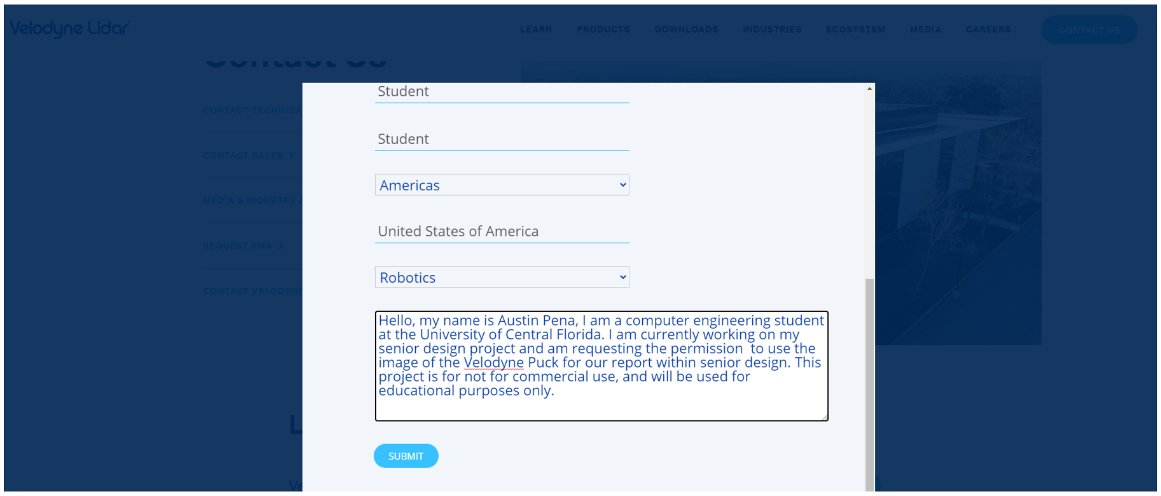
The image I would like to use are linked below:

https://www.vexrobotics.com/media/catalog/product/cache/1/image/1800x/040ec09b1e35df139433887a97daa66f/2/7/276-3526-plastic-insert.jpg

Figure 3.13.5: Tricycle Drive (Permission Pending)

Date: Wednesday, July 1, 2020, 4:49 pm
Thank You : Stavros Avdella (stavrosavdella@knights.ucf.edu)
Feedback as: Query
Message:
My name is Stavros Avdella, I am a Computer Engineering student at the University of Central Florida. I am currently working on my Senior Design project and I am requesting permission to use an image from your website. This project is not for commercial use and will be used for educational purposes.
The Image I would like to use is linked below:
http://www.robotplatform.com/knowledge/Classification_of_Robots/tricycle_drive.png

Figure 3.8.4: Velodyne Puck (Permission Pending)



Student

Student

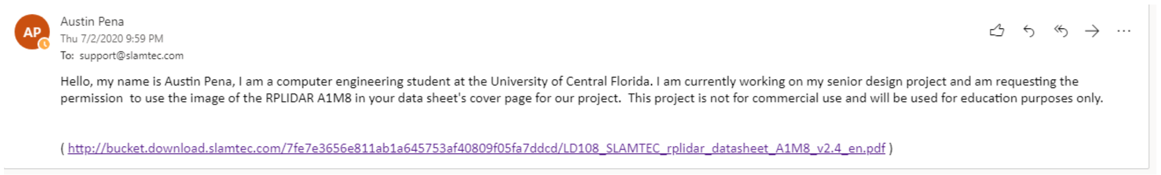
Americas

United States of America

Robotics

Hello, my name is Austin Pena, I am a computer engineering student at the University of Central Florida. I am currently working on my senior design project and am requesting the permission to use the image of the Velodyne Puck for our report within senior design. This project is for not for commercial use, and will be used for educational purposes only.

Figure 3.8.2: RPLidar (Permission Pending)

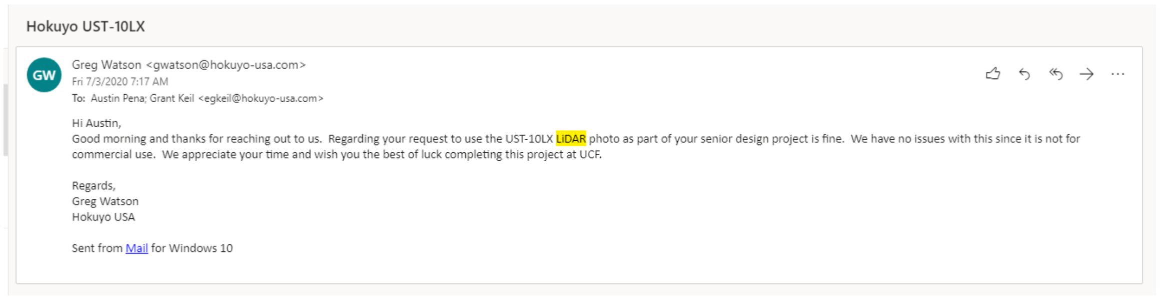


Austin Pena
Thu 7/2/2020 9:59 PM
To: support@slamtec.com

Hello, my name is Austin Pena, I am a computer engineering student at the University of Central Florida. I am currently working on my senior design project and am requesting the permission to use the image of the RPLIDAR A1M8 in your data sheet's cover page for our project. This project is not for commercial use and will be used for education purposes only.

(http://bucket.download.slamtec.com/7fe7e3656e811ab1a645753af40809f05fa7ddcd/LD108_SLAMTEC_rplidar_datasheet_A1M8_v2.4_en.pdf)

Figure 3.8.3: Hokuyo (Permission Granted)



Hokuyo UST-10LX

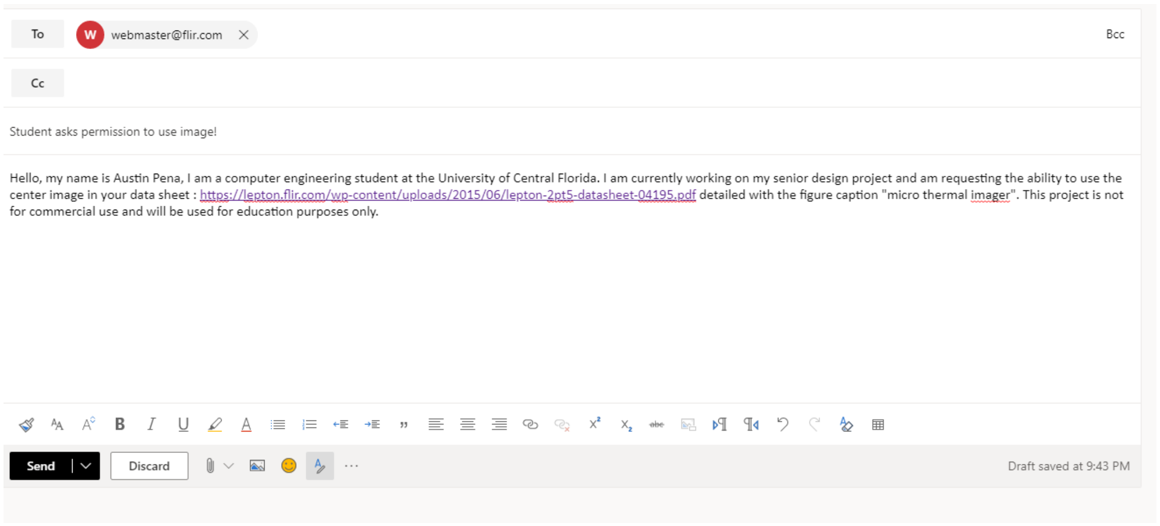
GW Greg Watson <gwatson@hokuyo-usa.com>
Fri 7/3/2020 7:17 AM
To: Austin Pena; Grant Keil <egkeil@hokuyo-usa.com>

Hi Austin,
Good morning and thanks for reaching out to us. Regarding your request to use the UST-10LX LiDAR photo as part of your senior design project is fine. We have no issues with this since it is not for commercial use. We appreciate your time and wish you the best of luck completing this project at UCF.

Regards,
Greg Watson
Hokuyo USA

Sent from Mail for Windows 10

Figure 3.8.6: Flir (Permission Pending)



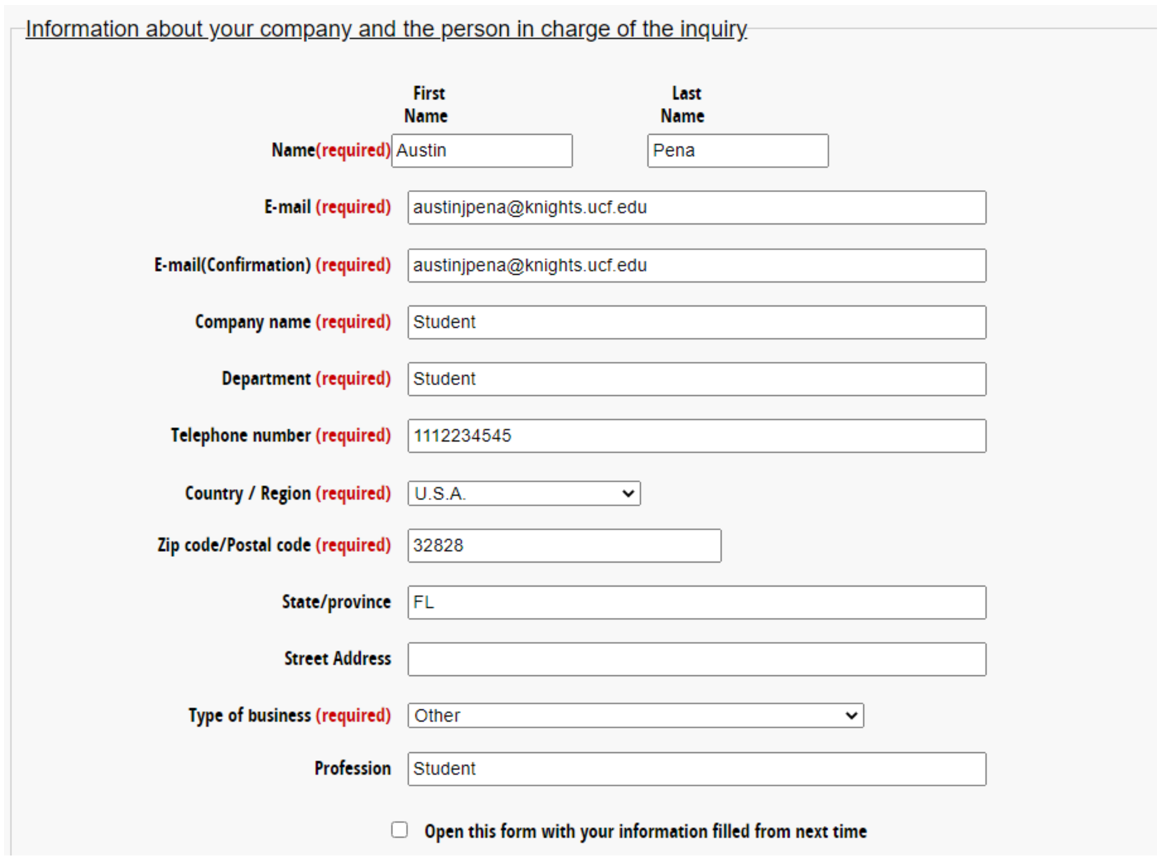
To: webmaster@flir.com

Cc

Student asks permission to use image!

Hello, my name is Austin Pena, I am a computer engineering student at the University of Central Florida. I am currently working on my senior design project and am requesting the ability to use the center image in your data sheet : https://lepton.flir.com/wp-content/uploads/2015/06/lepton-2pt5-datasheet-04195.pdf detailed with the figure caption "micro thermal imager". This project is not for commercial use and will be used for education purposes only.

Figure 3.8.5: GRIDEYE (Permission Pending)



Information about your company and the person in charge of the inquiry

	First Name	Last Name
Name(required)	Austin	Pena
E-mail (required)	austinjpena@knights.ucf.edu	
E-mail(Confirmation) (required)	austinjpena@knights.ucf.edu	
Company name (required)	Student	
Department (required)	Student	
Telephone number (required)	1112234545	
Country / Region (required)	U.S.A.	
Zip code/Postal code (required)	32828	
State/province	FL	
Street Address		
Type of business (required)	Other	
Profession	Student	

☐ Open this form with your information filled from next time

A.2 Works Cited

References

[1] University of Central Florida Electrical and Computer Engineering Department. *ECE Senior Design*. June 2020. URL: `http://www.eecs.ucf.edu/seniordesign/index.php`.

[2] Trevor Roman Brian Dodge Nicholas Musco. *The Autonomous Sentry Robot*. 2015. URL: `http://www.eecs.ucf.edu/seniordesign/sp2015su2015/g09/Archive/ASRSD1FinalReport.pdf`.

[3] Ismael Rivera Journey Sumlar Chris Carmichael Warayut Techarutchatano. *T-100 Watch Dog (Autonomous Security Vehicle)*. June 2020. URL: `http://www.eecs.ucf.edu/seniordesign/sp2014su2014/g04/documentsandfiles.html`.

[4] VEX Robotics. *VEX Robotics Mecanum Wheels*. June 2020. URL: `https://www.vexrobotics.com/mecanum-wheels.html`.

[5] KnightScope. *KnightScope K3 ASR Indoor Autonomous Security Robot*. June 2020. URL: `https://www.knightscope.com/knightscope-k3`.

[6] OpenMV Camera. *OpenMV H7 Camera*. June 2020. URL: `https://cdn.sparkfun.com/assets/a/3/f/d/9/OpenMV-H7_Datasheet.pdf`.

[7] Logitech. *Logitech C922*. June 2020. URL: `https://www.logitech.com/en-us/product/c922-pro-stream-webcam#specification-tabular`.

[8] Logitech. *Logitech StreamCam*. June 2020. URL: `https://www.logitech.com/en-us/product/streamcam#specification-tabular`.

[9] Canon. *Canon EOS Rebel T8i EF-S 18-55mm IS STM Lens Kit*. June 2020. URL: `https://www.usa.canon.com/internet/portal/us/home/products/details/cameras/eos-dslr-and-mirrorless-cameras/dslr/eos-rebel-t8i-ef-s-18-55mm-is-stm-lens-kit`.

[10] Antion Audio. *ModMic UCB*. June 2020. URL: `https://antlionaudio.com/collections/microphones/products/modmic-usb`.

[11] Audio-Technica. *ATR4750-USB Omnidirectional Condenser Gooseneck Microphone*. June 2020. URL: `https://www.audio-technica.com/cms/wired_mics/d9bb02096e4e4622/index.html`.

[12] SHURE. *CVL Centraverse Lavalier Condenser Microphone*. June 2020. URL: `https://d24z4d3zypmncx.cloudfront.net/Pubs/CVL/cvl-specification-sheet-english.pdf`.

[13] Battery University. *What's the Best Battery?* June 2020. URL: `https://battery_university.com/learn/archive/whats_the_best_battery`.

[14] MOOG. *How a Car Steering System Works: Easy Guide*. June 2020. URL: `https://www.moogparts.eu/blog/how-a-steering-system-works.html`.

[15] Parallax. *Boe Bot Robot*. June 2020. URL: `https://www.parallax.com/product/boe-bot-robot`.

[16] Robot Platform. *Robot Platform — Knowledge — Wheel Control Theory— Tricycle Drive*. June 2020. URL: `http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html`.

[17] Autodesk. *Autodesk Eagle: PCB Design*. June 2020. URL: https://www.autodesk.com/products/eagle/overview?plc=F360%5C&term=1-YEAR%5C&support=ADVANCED%5C&quantity=1.

[18] SunFounder. *16-Channel 12-Bit PWM Servo Driver*. June 2020. URL: https://www.sunfounder.com/pca9685-16-channel-12-bit-pwm-servo-driver.html.