Autonomous Solar Lawn Cutter (eGOAT)

Steven Cheney, Jordan Germinal, Eduardo Guevara, Davis Rollman, Jonathan Smith

University of Central Florida School of Electrical Engineering and Computer Science,

Orlando, FL

Abstract — A sponsored project by Orlando Utility Commission and Duke Energy, the eGOAT (electronically Guided Omni-Applicable Trimmer) is an autonomous lawn mower that effectively reduces expenditures and carbon footprints on the planet. It features technology such as Lidar and ROS. The eGOAT is really efficient in terms of power, as efficiency was our highest priority in terms of functionality. As this project is very large and required many different challenging aspects to be adhered to, numerous fields of engineering teamed up together, including: Electrical-Computer, Mechanical and Computer Science. The project requirements stated by the sponsors have a number of constraints that the engineers had to work with. While we did manage to achieve the majority of the restraints given by the sponsors, we were unfortunately impacted by the corona virus. This forced limited to no meeting times post March 9th, 2020. This made it difficult to test out the last couple constraints and integrate the project.

I. INTRODUCTION

Solar panels "harvest" energy from the sun using photovoltaic panels by allowing photons, or particles of light, to detach electrons from atoms. This in turn generates a flow of electricity. These solar panels are usually installed and set into an array on a large grassy field in order to produce more energy, avoiding obstruction from large objects such as buildings, trees, etc. in order to have the highest yield of renewable energy. Although this source of energy becomes a "free" source of energy over time, they are still severely expensive due to upkeep costs. For example, most of these farms are built over grass, as the grass neither absorbs nor reflects heat as severely as a flat concrete pad. However, as all grass does in the world, it grows every day.

The maintenance on such fields can be a major expense to the companies supplying the services and impacts the cost of energy users in the end. Our solution is the "e-GOAT": an AI-assisted autonomous solar powered rover-based robot that can cut the grass and reduce the cost of maintenance of the solar farm. The "e-GOAT" robot will be a high-functioning autonomous lawn mower that will move through terrain in order to cut the grass. The "e-GOAT" will be able to autonomously identify areas of grass that need attention, avoid obstacles, and provide motion and navigation to the land mower bot.

The low-cost bot will also be friendlier to the environment than traditional lawn service equipment; the bot will be electrically powered, unlike most regular conventional mowers, which in turn will allow it to reduce the carbon footprint to almost, if not completely, zero. As an added benefit, it will be able to operate at day or night, alleviating the need for extra lighting around the plant.

II. SYSTEM COMPONENTS

The eGOAT can be sectioned into ten key components with individual purposes used to assist in the completion of this project. These ten sections were divided up into the groups as described in this section.

A. Microcontroller

The microcontroller chosen for our project is the Atmega2560. This microcontroller was chosen because it operates on the lower spectrum of computational computing power. This microcontroller creates PWM signals for motor controllers, interfaces with digital and analog signal input and output, and supports serial communication between the Single Board Computer through serial communication. The Atmega2560 also provides a simple environment through the use of the Arduino IDE to program the chip and allow for faster implementation of software. Through the use of the simplistic Arduino IDE, we will be able to develop the solution we need through the plethora of C/C++ libraries available to us through the community. The microcontroller is also capable of several communication protocols, such as I2C, UART, and SPI on various physical ports.

B. RPLIDAR A2 Lidar

The Slamtech RPLIDAR A2 Lidar is being used for navigation. It enables the eGOAT to easily get around and detect obstacles. It is a very expensive and top of the line lidar with very good resolution and customization. Lidar is very simple in concept. The head of the lidar will rotate at a certain cycle, and one of the lidars ports will transmit a laser. The second port acts a receiver and catches all of the reflected light. With the combination of knowing its rpm,

when the light was sent, and when it was received, the ODROID can begin to build a 2D map of the robot's surroundings. This is very useful for things like obstacle avoidance, and navigation. We will use the lidar in addition with the camera to be able to more accurately know where the robot is and be able to better adjust our position to avoid obstacles.

C. ODROID XU4 main Computer

The ODROID-XU4 is powerful Single Board Computer and an energy efficient piece of hardware. It is one of the fastest of the three Single Board computers that is on the market. The ODROID-XU4 supports open source, which allows us to run various versions of Linux, including Ubuntu 16.4, Android 4.4, KitKat, and 5.0 Lollipop. The ODROID-XU4 has extremely amazing data transfer speed that complements its already powerful processing power of eight ARM CPU cores. This is a very important aspect for this project, as it allows a lot of computing and data transfer for the navigation and wireless communications.

The ODROID-XU4 comes out on top when comparing it with the other eight core Single Board Computers out on the market. It is one of the mature Single Board Computers, and the ODROID-XU4 is able to run the mainline Linux kernel. Documentations and supports are widely available for the ODROID-XU4, as it has a really big community of people that are willing to help and get beginners started on their project.

We had very minor difficulties with the ODROID XU4, due to a small yet vital issue. This issue is further explained in the next section, which also includes the reasoning on why we ended up having to utilize the Raspberry Pi 4 instead.

D. Raspberry Pi 4

The Raspberry Pi 4 runs at 1.4 GHz with 4GB of memory. The cost of a Raspberry Pi is relatively inexpensive at a price of only $35. It has the capability to support the camera as an input, as well as enough inputs and outputs to support the different sensors that are required for this complex project. The Raspberry Pi 4 also operates at 5V and contains 4 USB 2.0 ports.

The Raspberry Pi 4 supports Wi-Fi and BLE protocols. The Raspberry Pi 4 has a Quad Core Broadcom BCM2837 64-bit ARMv8 processor. It also has an integrated GP-Video Core IV that runs at 400 MHz. These product specifications are really good processing speeds that meet the high video requirement constraints that this robot demands for computer vision architecture in order to make important navigational decisions. The power consumption is relatively good as it does not consume too much power for the task that it would handle.

As you would expect from a Single Board Computer, it can be programmed in numerous different programming languages, including C and C++. It comes with its own operating system called Raspbian OS.

We ended up switching to the Raspberry PI 4 from the ODROID-XU4, because we were having issue with the ODROID we had ordered. This issue was that the ODROID-XU4 was unable to start the services required to start all of the ROS components and our program. When we tried the same process for the Raspberry PI 4, we were able to get all of the required services and nodes to launch.

E. IMU

The IMU module for this project is the FXOS8700 6-Axis sensor Accelerometer & Magnetometer and the FXAS21002C 3-Axis Digital Angular Rate Gyroscope. The IMU sensor consists of three distinct chip modules that can be interfaced using the IC2 standard. When working with the sensors, it is important to correctly wire the devices in order to enable proper communication through IC2 to the microcontroller. The FXOS8700 sensor package is a small, low power three-axis accelerometer and Magnetometer that is combined to be a small CBD package. The package is supplied with voltages between 1.62 and 3.6v with a low current draw of 240μA while only one sensor is active, and a current draw of 80μA when both sensors are active. The FXAS21002C sensor package is a small yaw, pitch and roll rate gyroscope that is packaged as a CBD surface mount component with good accuracy.

F. Motor Controller

H-bridge design is used to control motor movement Series of On-Off pulses (PWM). This controls the amount of power delivered without wasting power. Pulse bits are used to control the direction of the wheelsclockwise versus anti-clockwise. This enables us to control the speed and direction of the motors.

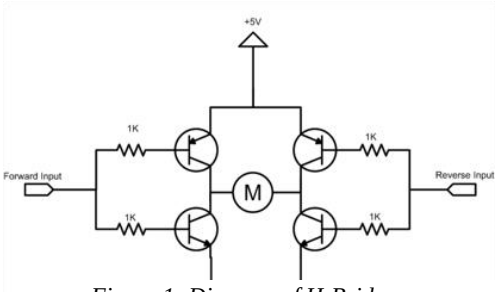

Figure 1: Diagram of H-Bridge

G. *GPS Module*

The MTK3339 GPS module is a SMD GPS chipset with a high sensitivity level and low power consumption for precise GPS signal processing to get precise location in sub-optimal conditions. The module comes with pin input for external antenna I/O and comes with an automatic antenna switching function with short circuit protection. The GPS module is capable of a high update rate of up to 10Hz with proprietary self-generated orbit prediction for instant positioning fixing. The modules have a cold startup time of 34 seconds. This GPS utilizes the NMAE 0183 communication standard help by the National Marine Electronics Association.

The module has been developed with the Adafruit Ultimate GPS Breakout V3 board. The board has been useful for testing and implementation purposes. The breakout board contains an external antenna support with a uFL connector. This has been converted to an SMA attachable antenna using an SMA to uFL adapter. The development board comes with 9-pin out connections, two digital I/O for serial communication, and two VIN pins for redundancy. This has been used for communication with the microcontroller to gain positional coordinates

H. *Trimmer Motors*

The Mechanical team has concluded with the proper trimmer head to be equipped with the rover. The MaxPower Pivot Trim is a universal trimmer head that is powerful enough to accept both 0.080” and 0.095” line thickness for the trimmer string. There are three pivoting lines to prevent breakage and is easily able to use.

I. *Drive Motors*

For the wheel motors, we as a group are less interested in speed, and more interested in torque, because the mower is only going 2-4mph in order to achieve the cleanest cut, a restraint given by the sponsor, yet still has to pull around 30lbs. The wheel motors will be attached to the frame of the mower and will be driven by the motor driver which will in turn be driven by the microcontroller.

J. *Perimeter Wire / Tank Circuit*

Boundary wire offers a simple solution to creating an invisible barrier around a perimeter of land. It is used in almost all commercially available autonomous lawn mowers today. It is also used for other applications such as invisible fences for pets and other commercial guidance cases. Boundary wires use two different subsystems that comprise of the larger solution: a perimeter wire with a function generator attached in a circuit, and a receiver with an electromagnetic field (EMF) sensor.

III. SYSTEM CONCEPT

The main goal behind the eGOAT is to cut the costs and potential risks involved with human ground maintenance. The AI-guided task planning software of the eGOAT will be able to detect obstacles in the environment, identify locations and objects in need of trimming, and construct a sequence of tasks to accomplish mission objectives and avoids any potential damage to important infrastructure. The system will be using a variety of components such as ODROID XU4(Switched to Raspberry PI 4), RPLIDAR, camera, and voltage step down modules in order to aid the eGOAT to operate autonomously.

A. *System Hardware Concept*

The first figure below shows how the hardware will be connected via power and data lines. All the components have either power or data connected to the printed circuit board for the eGOAT.

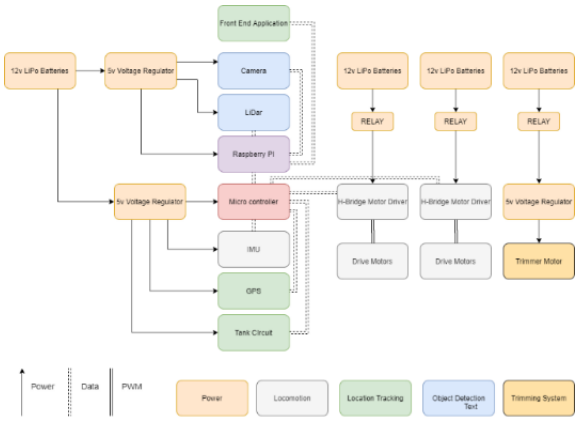

Figure 2: Block diagram presenting major system components

The second figure below shows a high-level mock-up of the rover with the system components within the autonomous robot. This is an additional piece of information to display the communication system between components and to show the different protocols each one uses.

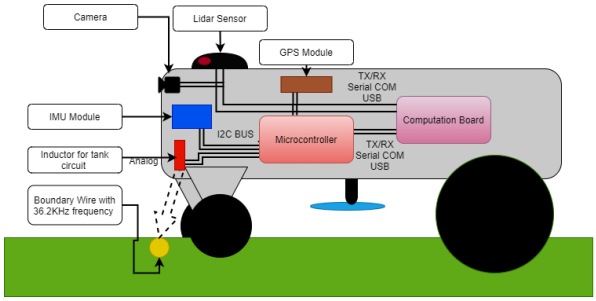

Figure 3: eGOAT Hardware Placement Diagram

IV. HARDWARE DETAIL

A. *Function Generator Design*

For the first part of the perimeter, we needed to create a function generator to produce a 36.2Khz frequency signal over 150ft of 20-gauge copper wire. Using the parts selected for the function generator, an initial prototyping sketch was gained from the NE555 manual to understand the connections for the NE555 Timer Circuit.

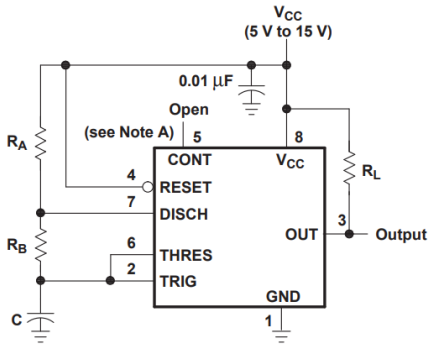

Figure 4: Function Generator Design

To set the duty cycle and frequency of the square wave, we use the following equation:

$$f = \frac{1.44}{(Ra + 2 * Rb) * C}$$

Where f is the frequency desired, Ra is the resistor in series with Rb and C. We would like a frequency between 32KHz and 44KHz. As experienced in other projects, these frequencies should not interfere with our GPS, Wi-Fi, or other modules onboard the rover. We also use a potentiometer to change the frequency to match the resonance frequency of the receiver circuit. The potentiometer has a resistance of 12Kohms with an additional choice of + 4.7Kohms. Using equation (1) we calculate our upper bound and lower bound frequency fL and fU using the max and min resistance of the potentiometer:

$$fL = \frac{1.44}{(3.3 + 2 * (12 + 4.7)) * 1.2e10\text{^}(-9)} \approx 32.698KHz$$

$$fU = \frac{1.44}{(3.3 + 2 * (12 + 0)) * 1.2e10\text{^}(-9)} \approx 43.956KHz$$

When we increase resistance of Rb, we reduce the frequency of the timer while keeping the gain constant. When we decrease the resistance of Rb, we increase the frequency of the timer while also keeping the gain constant.

B. *EMF Sensor*

To detect our square wave produced by the function generator in the previous section, we needed a circuit that picks out the created frequency and determine if we have tripped the wire or not. This way, it can act as an electrical resonator to store the energy of the frequency being emitted and we can measure this as an analog signal into the microcontroller. This kind of filter needed would be an LC filter or so called a resonant filter. A resonant filter consists of an inductor and a capacitor in parallel or series that through Faraday's Law and the drop of magnetic field, the capacitor is charged up at the resonance frequency of the circuit.

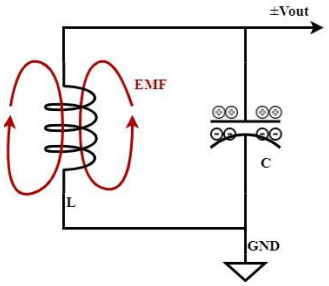

Figure 5: EMF Sensor

Calculating the resonance frequency based on the frequency outputted by the function generator for parallel LC circuit, we get:

$$f0 = \frac{1}{2 * \pi * \sqrt{L * C}}$$

where L is the inductance value of the coil in Henry (H) and C is the capacitance value measured in Farads (F). Fixing the capacitance for a small capacitor at 22nF, we get an inductance value of L = 1mH at the resonance frequency of 33.932KHz. The voltage amplitude of the capacitor is fairly small due to the change in the magnetic field. With such a small change, the microcontroller chosen is not capable to detect a small voltage amplitude.

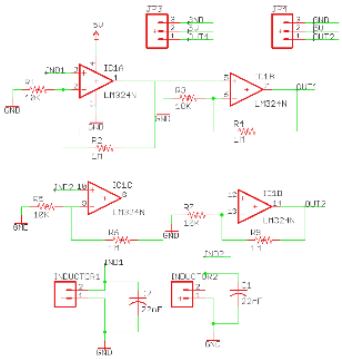

Figure 6: EMF Sensor Design

In order to increase the amplitude, we also use an Op-Amp to increase the signal that can be read to the microcontroller. The chosen Op-Amp, the LM324, contains four Op-Amps but we only make use of two. To achieve a non-inverting gain of 100, we connect two resistors in series with the Op-Amp to specify the desired gain. Using the data sheet provided we chose $R1 = 10KOhms$ and $R2 = 1MOhms$ to achieve a resonance frequency around 34Khz. The gain achieved is then able to be read from the analog inputs in the Atmega2560 Analog inputs via PF0 and PF1 pins. The reading will then allow to determine if the rover has crossed over the wire connected to the function generator and determine what navigation action to take.

C. GPS Module

The GPS module chosen for this is the MTK3339 GPS module. This module needed its own serial communication pins connected to the microcontroller board in order to use the NMAE packet communication protocol. The GPS module utilizes one of the several serial protocol ports available on the Atmega2560. These pins are PD3 for TX and PD2 for RX. We also needed a Vin of 3.3v regulated from either the microcontroller itself or an external regulated power supply (for example from the on board LiPo batteries regulated through another device) to increase the signal coverage that the GPS module can receive. The module is also extended with a large antenna. The antenna is mounted on the back of the rover outside of the main body to reduce interference within the electronic chamber and reduces interference from the shielding of the rover.

D. IMU Module

The IMU modules will communicate via the IC2 standard through both the FMX015700 and the FXAS21002C. Both modules contain a unique 8-bit address that will be used to address the modules from the controlling master unit. There are several slave addresses that can be assigned to the modules through the SA1, SA0, GA1, or GA0 ports assigned on the modules by raising them high or low.

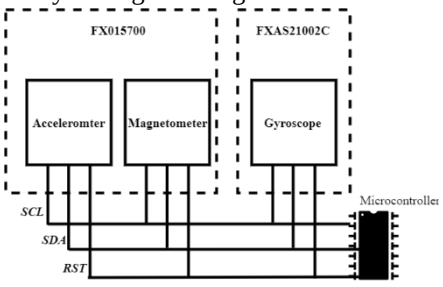

Figure 7: IMU Wire Diagram

In order to obtain the most accurate measure of direction and orientation, the IMU is calibrated using open source software developed that calculates the soft and hard iron error in the magnetometer. Once the components on the rover have been installed onto the rover and the rover is in its environment of operation, the rover is calibrated by rotating the body in several directions for a time of three minutes. This way, the correct soft and hard iron error can be calculated. This is then taken and placed inside the firmware of the microcontroller.

```
// Offsets applied to raw x/y/z mag values
float mag_offsets[3]            = { 99.23F, 177.02F, 120.28F };

// Soft iron error compensation matrix
float mag_softiron_matrix[3][3] = { {  0.943,  -0.065,  0.022 },
                                    {  -0.065,  0.966, 0.005 },
                                    {  0.021, 0.005,  1.101 } };

float mag_field_strength        = 40.96F;

// Offsets applied to compensate for gyro zero-drift error for x/y/z
float gyro_zero_offsets[3]      = { 0.0F, 0.0F, 0.0F };
```

Figure 8: Soft and Hard Iron compensation Values Calculated Used for Sensory Fusion Algorithms

V. SOFTWARE DETAILS

Part of having an autonomous lawn mower means that it must be "autonomous." This means that the robot being constructed must be able to move, or navigate, around by itself, without the aid of any remote control, or human interaction. In order to achieve this feat, we are planning to use a sensor that will be able to detect objects around it. Although several sensors were looked at in comparison with the Lidar sensor, we found that there was ultimately no better sensor out there on the public consumer market that could compete, especially at the same price point we were able to acquire this Lidar sensor at. The following section will delve deeper into the reasoning behind why we chose the Lidar sensor.

A. Lidar

The Lidar is the primary range-finding sensor that the eGOAT will use to construct its occupancy map. It allows for highly accurate range-finding capabilities in all directions around eGOAT with a resolution measured in centimeters. However, the Lidar is only capable of collecting range data

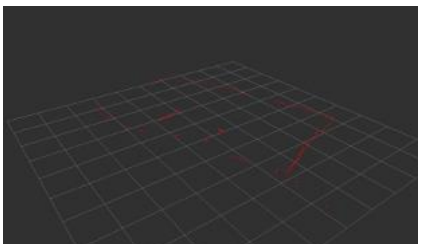

Figure 9: Lidar Scanning its Environment

in a 2D plane around the eGOAT, meaning that it will need help from other sensors to detect difficult terrain or obstacles that sit on the ground below the Lidar's vertical range.

B. Simultaneous Localization and Mapping (SLAM)

SLAM, or Simultaneous Localization and Mapping, is a field of algorithms for generating a map of the world based on sensor input over time. Two of the most common algorithms are LIDAR based or camera-based SLAM. We are using a LIDAR based solution, as provided by the open source software OpenVSLAM. OpenVSLAM is one of the main commonly used SLAM libraries and has been used on similar low powered hardware by others. OpenVSLAM also has a ROS package available, which will be handy because we decided to use ROS to fuse all parts of this project togehter. While we may be restricted in the processing of the LIDAR input, our LIDAR sensor is not a particularly advanced sensor anyway. In conjunction with the Lidar sensor, we can build an estimate of the world with a point cloud.

One of the techniques of map building that we are implementing is the Simultaneous Localization and Mapping (SLAM). This technique can be used in partnership with a camera. SLAM is available to us using the ROS Navigation stack. It would enable us to use the lidar to create a map of the environment and be able to use that map to localize itself on the map and know where it has navigated or not. 3D reconstruction is the ability to create a 3D map of the environment. A camera that have that feature can help the robot understand and interact with the world.3D reconstruction is very important when it comes to collision avoidance, motion planning, and realistic integration of the real and virtual world as seen in the figure below.

C. Computational Hardware

After we had carefully evaluated the different Single Board that we were considering and that would fit our project, we were going to go with the ODROID-XU4. At the time of consideration, the ODROID was a clear favorite, as it had all the required specifications. Despite it being a slightly higher price than the Raspberry Pi 4, it had all the features we needed and has better processing power while keeping its power consumption as low as the Raspberry Pi. Overall, the ODROID-XU4 was originally the clear winner in our preliminary decision round.

The ODROID-XU4 was overall the best option at the time. We rated the comparisons of all the boards previously mentioned on a scale of 1 to 5, with 1 being the lowest and 5 being the highest. These scores were then multiplied by a factor of 3 in order to give us a closer comparison on the effectiveness of each potential product. The ODROID has the ability to handle the required load that we need for this particular project. With this rating scale, we went ahead and ordered the ODROID and began testing. However, unfortunately, it did not quite pan out to being the option that worked with our program when implementing with ROS. Because of this, we ended up having to change the board we chose. The next best choice on our list was the Raspberry Pi 4, and so we placed an order on this board.

As discussed in Section II, Subsection D, we chose the Raspberry Pi 4 as our board, as it had better integration with the components we had chosen and purchased, as well as booted up completely with the program we wrote.

Using the simultaneous localization and mapping (SLAM) algorithm, as well as the algorithms to handle navigation; sensor filtering and synthesizing; and short-term and long-term task planning just to name a few. The relatively large processing power and multitasking required by these algorithms will require a reasonably powerful computer with an operating system and graphics processing unit in order to coordinate the various functionalities and subsystems of the eGOAT as well as interpret the data feed from the sensor suite and servos in the amount of time available.

D. Software Middleware

In order to communicate with multiple peripheral devices such as sensors and controllers, a communication form or protocol must be used in order to handle various streams of data. To do this, we use the software middleware ROS to create channels of information that receive information and give commands. Sensors and control hardware are set as topics that can be accessed to receive information. Through use of a specific library called ROS-Serial library, we are able to create topics to sensors and equipment controlled

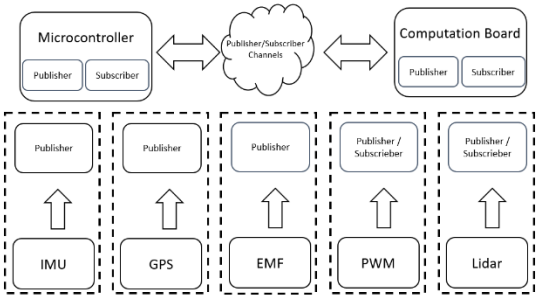

Figure 10: Visualization Diagram of ROS Topics

through the microcontroller by feeding information through serial communication to the high level single board computer, in this case, the Raspberry Pi 4. Serial communication is passed though USB from the Raspberry PI to the microcontroller. Topics can either publish information or subscribe to other publishing topics.

As depicted in the visualization, all sensory and control I/O such as the IMU, GPS, EMF, PWM, and Lidar sensors are associated with a topic to publish information or subscribe to information streams. ROS topics are created with ROS-Serial Arduino library using C++ library and compiled and uploaded with the Arduino CMake toolchain.

```
ros::Publisher orientation("orientation", &geometry_msg);
ros::Publisher orientation_chatter("orientation_chatter", &std_msg);
ros::Publisher positionP("position", &fix_msg);
ros::Publisher timeP("time", &time_msg);
ros::Publisher tankCircuit("tank_circuit", &tank_msg);
```

```
//Create Subscriber after node listener is created.
ros::Subscriber<geometry_msgs::Twist> motorCmd("cmd_vel", &messageMotors );
```

Figure 11: C++ Publisher and Subscriber Declarations

E. PWM Control in Firmware

To control the H-Bridge drivers from the microcontroller, two dedicated PWM ports and timing registers are used to create variable duty cycles depending on commanded input. This command input is given through the "cmd_vel" ROS topic as a linear velocity in m/s and an angular velocity in rad/s. The subscriber on the microcontroller receives the command from the Raspberry PI via the serial communication port. When the message is received, an interrupt is raised to convert the high-level command into a PWM duty cycle and directional output. Timer3 on the Atmega2560 is used to set variable duty cycles. With the help from the PJRC Timer Three Library, it is easy to set a PWM signal through Atmega microcontrollers. Using telemetry calculations of the rover body and wheel diameter, we are able to roughly approximate the PWM duty cycle needed to achieve the speed.

F. Reduction of Problem

We reduced the problem of finding a shortest path with the max coverage we can do, where we define coverage as the total area the robot moves through. The robot is not just moving in a line but takes up additional area (the grass that the blades will cut if the robot is standing still). There are numerous papers available to solve this common problem. One interesting paper uses a modified A-star algorithm to solve this problem. We transformed our input into a useable occupancy grid map, we implemented this algorithm as is to solve our path planning issue.

The output of the algorithm is a list of waypoints for the robot to follow. It is still necessary, however, to have a good localization method, so we can know where we are in relation to each waypoint. The motion planning then becomes trivial

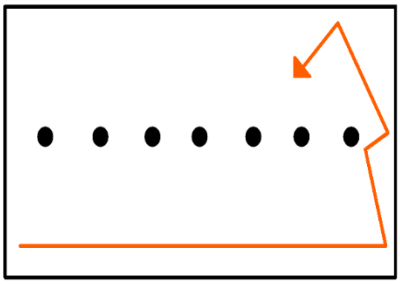

Figure 12: Illustration of Robot Motion Path

robot motion from waypoint to waypoint. The following image is from the paper, illustrating a potential map and solution as in Figure 11. This could work very well with a higher-level design using RTK-GPS, or any method with an accurate Localization technique. Some other motion planning techniques we've discussed would not work well with this, such as a pure computer vision approach, avoiding obstacles with something like a camera, combined with Object Detection or Edge Detection and obstacle avoidance, because we would not have a full map of the world along with a precise position of where we are in that world at the current time.

In order to deliver a working prototype as fast as possible (which itself is a large improvement from all of last year's groups, which functionally did not do any autonomous grass cutting, even though they planned it), we have come up with a simple method for cutting. The method will be expanded after we have it working. This method is very similar to how an indoor iRobot Roomba works and does not have any special planning or mapping. We used three main sensory

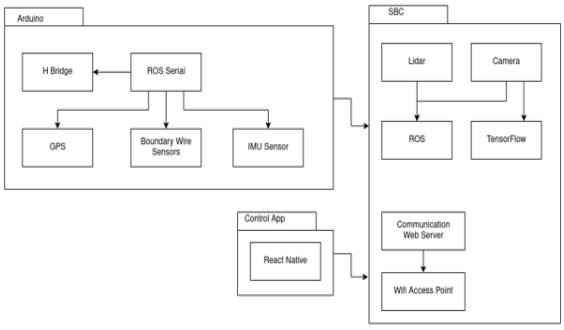

Figure 13: Navigation Stack

components for this method; boundary wire, lidar, and the camera.

With these three components, we adequately created a minimal prototype that fulfills all the project requirements. The system will work on a random turn when it runs into an obstacle, just like a Roomba. The boundary wire detection and Lidar obstacle detection components will be the two signals for the robot to back up and rotate, before proceeding ahead.

The robot could be controlled by a mobile app, created using React Native technologies. The app communicates to an integrated web server on the rover, over the Wi-Fi access point on the board. The app allows basic manual directional control, as well as retrieving GPS coordinates of the rover. However, due to restrictions, we could not get this integrated in time. Instead, the rover's manual control uses a wireless game controller, which was simple and easy to configure with ROS.

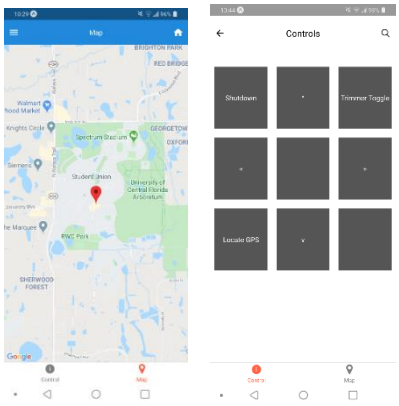

Figure 14: Application UI

G. TensorFlow API For Object Detection

The camera will be used exclusively for person detection in order to meet the requirement of shutting down the trimmer motors when a person is detected in front. We used a pre-trained neural network and very common such as the TensorFlow API and conjunction with OpenCV to perform object recognition. Because we are not using an NVIDIA Jetson, and our chosen Raspberry board is weaker, it is able to run at a few frames per second. Early testing has confirmed this to be true, as it is very. This is perfectly acceptable to us.

The model has been trained on the MS COCO dataset (Common Objects in Context). It was trained on a dataset of 300k images of 90 most found commonly objects, which includes dogs, cats, Laptops, Teddy bear, chair, phones, and others.

Our choices for object detection are quite sophisticated and will be able to accurately detect people and make the right decision on almost any object it sees.

VI. THE ENGINEERS

Steven Cheney is one of the three Computer Engineers on this EGOAT team. He is a jack of all trades, having experience ranging from hardware assembly and design, to having programming experience.

Jordan Germinal is one of the three Computer Engineers working on this project. He is primarily working on the autonomous lawn mower's robot vision with the Computer Science major on this team, Davis Rollman.

Eduardo Guevara is one of the three Computer Engineers within the project that will be helping with several subsystems towards the final solution. He focused in working on the Navigation and Localization subsystems closely related to several parts of the hardware stack. He will be taking on a position at L3Harris as a digital design engineer.

Davis Rollman is the only computer science major on the team, but not the only coder. He worked with Jordan Germinal on Computer Vision aspects of the project. We will be using the LIDAR sensor to gather data about the world around the robot.

Jonathan Smith is the only electrical engineer on the team, his main focus was to be on power and PCBs. This project needed quite a few PCBs including the microcontroller, motor drivers, voltage regulators, remote relays, and emf sensor.

VII. ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Dr. Samuel Richie, Solar lawn cutter Green Team: Mechanical and Computer Science Teams and the University of Central Florida.