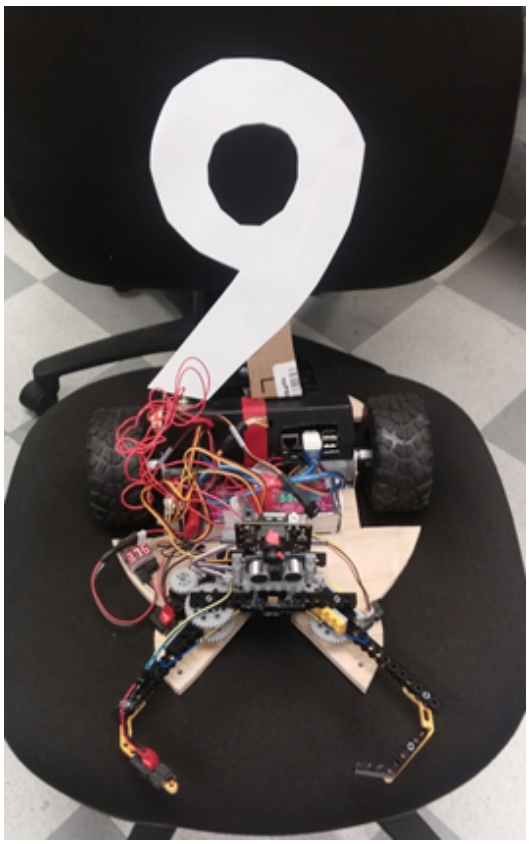
Object Gathering and Delivering Robot

Using Autonomous Techniques and Inter-Robot Collaboration



ENEE408I: Autonomous Robotics
Section 0102 - Group 9

Zachary Lawrence ● Paul “Nick” Laurenzano ● David Lim

Abstract

The goal of ENEE408I is to work in a group of 3 students to produce a fully autonomous robot capable of completing a small task and communicating with other robots. We constructed the frame of our robot using wood, plastic parts and legos. We then outfitted the frame with 2 motors, a servo, 3 IR sensors, a ping sensor and a pixy camera. We communicated with all of these sensors though an Arduino Uno and Raspberry Pi 2. Inserting a WiFi dongle into the Raspberry Pi enabled us to utilize wireless communication between robots to coordinate actions. The WiFi dongle also allowed us a host a web page on the Raspberry Pi that would allow for remote control of the robot through a browser.

Project Overview

Initial Objective

In the beginning of the course, the goal of the autonomous robot was broadly defined to allow the task to be ultimately defined by the capabilities of the final robots. From the start, we knew that our robot had to collaborate with other robots in order to collect different colored cones (fig. 1) and deliver them to an end zone (fig. 2). Of course, the robots had to do this all autonomously.

Final Objective

By the end of the course, the task for the robots was well defined. Teams of 2-3 robots were placed in an arena about 8 feet by 6 feet. The bottom of the arena was made out of foam, and there were 1 foot high walls around the perimeter of the arena. Each robot in the arena would take turns delivering a cone to the end zone, which was a large green square placed vertically in the corner. The robots had to deliver the colored cone in order, i.e. no robot could deliver a cone before the others took their turn. Additionally, the color of the delivered cones had to follow a predefined pattern; for instance, orange then blue alternating. When a cone is delivered to the end zone, it is removed to avoid cluttering.

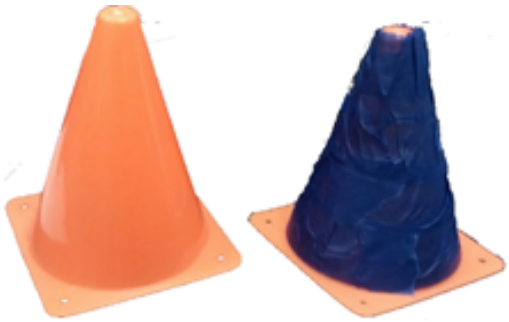


Figure 1: Orange and Blue Collection Cones

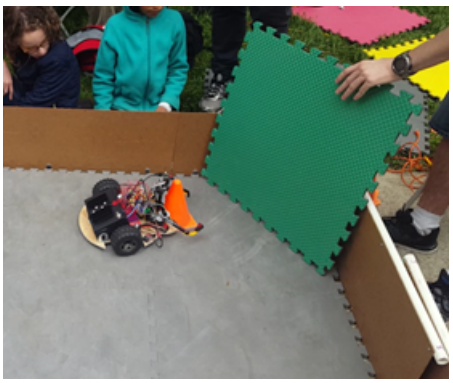


Figure 2: End Zone

Requirement Analysis

To meet our project goals, we set out to create a robot that fulfilled the following demands. We then determined what hardware and software components were necessary to achieve them.

1. Our robot shall be autonomous.
    - It needs to have a processing center and sensors.
    - It needs to have algorithms to interpret its sensor data in the context of the challenge.
    - It needs to be fairly easy to add, change, and remove portions of code as development progresses.
2. Our robot shall be wireless.
    - It needs to have an onboard power system capable of supporting all devices for a reasonable amount of time (at least 3 hours, the length of the lab time).
    - It needs to have wireless communications for software development and high-level controlling of the robot (running software or not, etc.).
3. Our robot shall be able to achieve basic obstacle avoidance.
4. Our robot shall be able to collect orange and blue cones.
5. Our robot shall be able to deliver cones to a designated area.
6. Our robot shall be able to communicate and collaborate with other robots.

Implementation

Hardware

Structure, Drive, and Mechanisms

The main body of our robot is a flat, round piece of wood with cutouts for each of the wheels and a cone wedge. We installed some brackets and Velcro attachment points to help hold the various parts stable and in a consistent arrangement. This allowed us to build a fairly lightweight, low-to-the-ground robot that performed very consistently.

The drive system is comprised of two large rubber and plastic wheels and a metal bolt under the front center to act as a leg to maintain balance. The leg had a LEGO® glued to the bottom of it to decrease friction as it was dragged across the tile floor, foam floor mats, and other surfaces.

At the front of our robot is a LEGO® claw system that allowed our robot to firmly capture and hold onto the plastic cones, allowing it to maneuver freely around the arena without worrying about losing the cone.

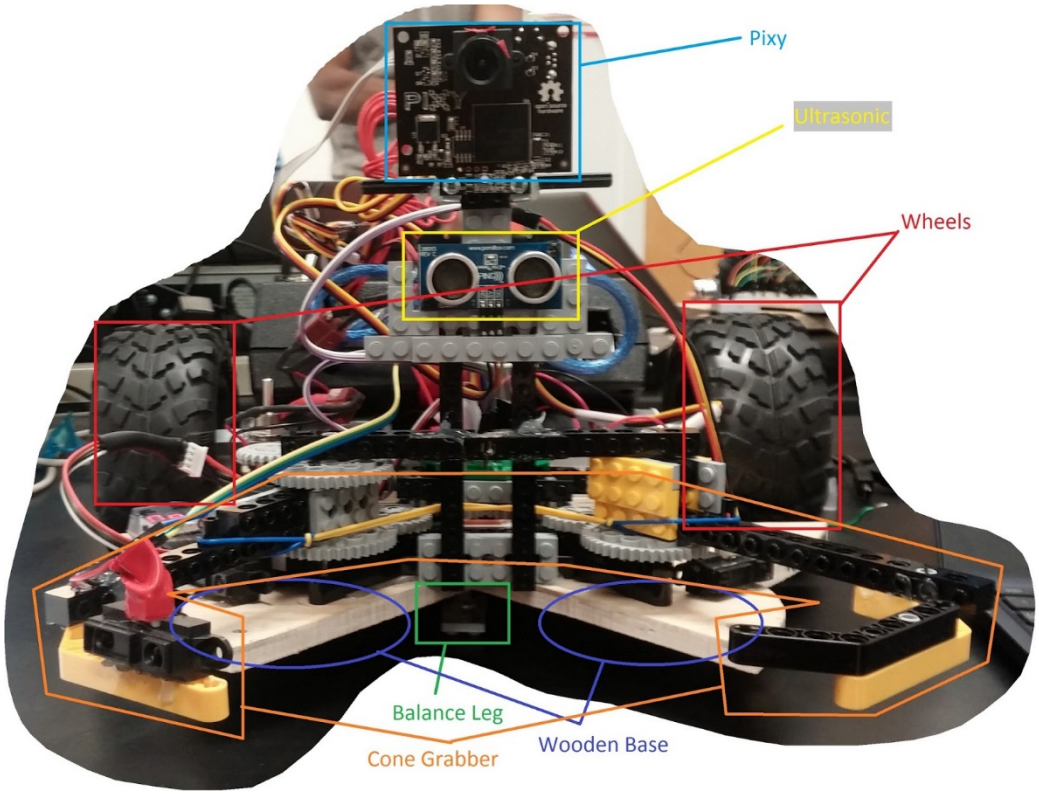


Figure 3: Hardware Components

Electrical and Power

Power was a major concern of ours from the outset, and we fared well through the semester because we invested a large amount of much time and effort into making our power system robust and flexible from the beginning. All of our electrical systems were run from a single 5.1 amp-hour battery, which was connected to two variable-voltage MOSFETs (configured for 5-volt output, sharing a single large heat sink without directly electrically conducting) and a motor controller. The battery was connected to these devices through a master power switch, making it easy to shut down power to the entire robot and store it for long periods of time with the battery still attached. The battery also had a device that could monitor and display the voltage of each of its cells and make a loud wailing noise if the voltage dropped too low, indicating that it needed to charge.

The first MOSFET powered our Raspberry Pi 2 computer via a pluggable USB port, which was the core control for our entire system. It had a Wi-Fi dongle which allowed us to remotely access the robot, and also connected to the Arduino Uno via USB to provide it with power and to facilitate communications.

The Arduino Uno was the sensor and actuator hub of our robot, and was essentially slaved to the Raspberry Pi. It was connected to an ultrasonic sensor, 3 infrared (IR) sensors, a servo, the

motor controller, and a Pixy camera. It was effectively a mediator between these devices and the Raspberry Pi.

The second MOSFET powered the ultrasonic sensor, all 3 IR sensors, and the servo. This was because running all of them directly from the Arduino Uno could have caused enough current draw to ruin part or all of the board. We elected to play it safe, and allow possible room to expand our sensor array in the future.

The 3 IR sensors and the ultrasonic sensor were used for obstacle avoidance and detecting when the delivery zone was reached. The ultrasonic sensor was also used to determine when a cone was close enough to be acquired. One IR sensor was mounted to one of the cone-grabbing arms so that when the ultrasonic sensor was covered by an acquired cone, we could still allow the robot to effectively see obstacles in front of itself.

The servo was used to open and close the cone-grabbing mechanism. A LEGO® gear was mounted to the top of the servo and the servo was mounted to allow that gear to interface with the gears already built into the grabbing mechanism.

The motor controller was powered directly from the battery since it had on-board voltage regulation already, but we added an extra switch into the power circuit to allow us to turn off the motors while running the rest of the code. This was very valuable in testing the rest of our system without worrying about our robot running into walls or leaping fearlessly off tables. Wires from the Arduino lead to the other inputs of the motor controller to help determine the direction and speed of each of the motors independently, giving our robot full freedom of direction.

The Pixy camera allowed our robot to identify objects based on their color and size in its field of view. We placed it high enough that even after the robot acquired a cone it could see over it, so that the robot could still find the delivery area (which was designated by a color).

Many of our electronic systems were mounted inside of a black plastic box (fig. 5-6), to make the robot more aesthetically pleasing and to protect them from hazards such as other violent robots and wire-yanking small children. All of the external components had some kind of easily-detachable connector to allow easy removal and swapping of parts.

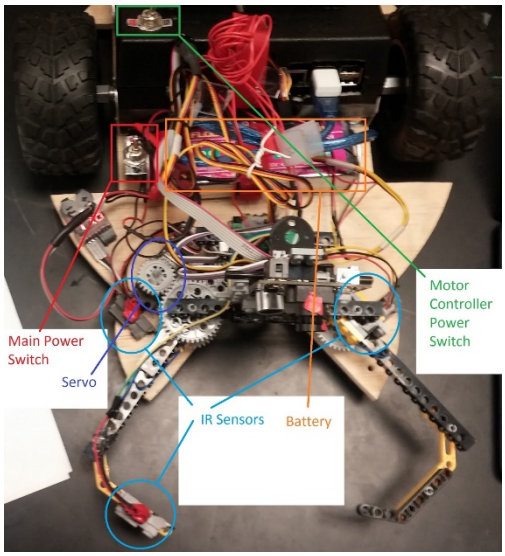


Figure 4: Basic Electronics

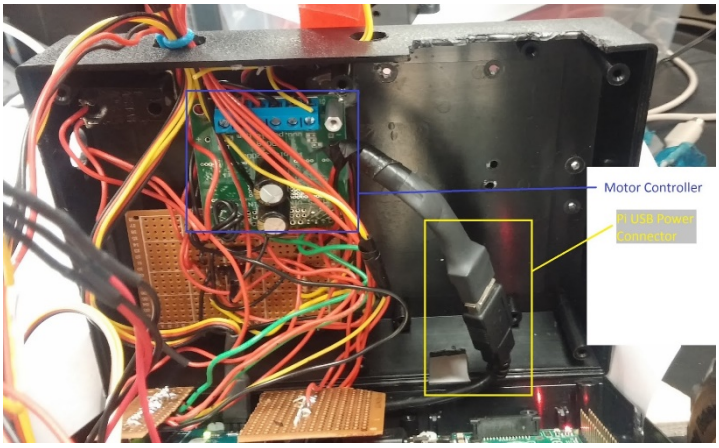


Figure 5: Electronics Black Box, Top Half

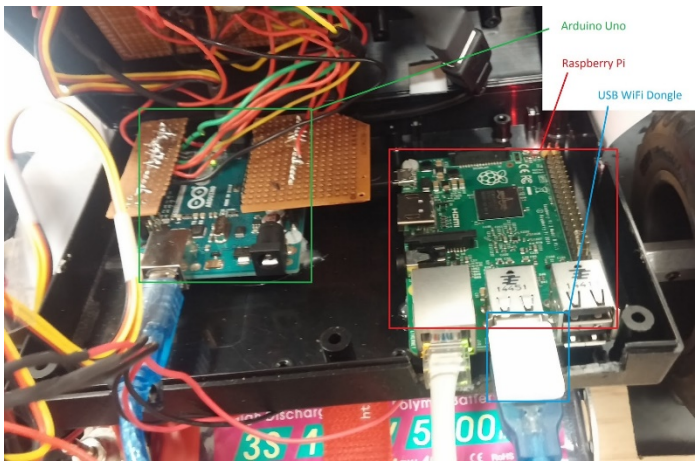


Figure 6: Electronics Black Box, Bottom Half

Software

Framework

As described above (see Hardware), the brains of our robot existed across both an Arduino Uno microcontroller board as well as a Raspberry Pi 2 mini computer. The division of labor between the two is pretty natural: the Arduino interfaces with all of the hardware, handling lower-level controls, while the Pi handles higher-level logic as well as the inter-robot communication. The main decision we had to make was how the Arduino and Pi would interact with one another.

Our initial mindset was that the Arduino would contain all of the functions for sensor data gathering and motor control. These functions would be invoked by the Pi, who would first gather sensor information from the Arduino, process it, then move the robot, again by using functions contained by the Arduino. All of this communication would be done via serial communication, using an original protocol that we designed. However, this quickly became a nitpicky and complicated task, so we decided to look for other solutions.

We came across an already established protocol for serial communication between an Arduino and Raspberry Pi called Firmata. While you can use Firmata solely as a means of serial communication, the more common use is to preload the Arduino with a standardized sketch, then use a developed library to setup and use the Arduino. This essentially turns the Arduino into a slave, and any control that the Arduino had over the hardware is shifted to the Pi. We found this setup very appealing. Not only would it give us an easy way for the Arduino and Pi to communicate, it would also simplify software development because all the code could now be contained in the Pi.

With the Arduino setup as a slave, we were able to connect all of the hardware (motor driver, sensors, pixy camera) to the Arduino and channel them all to the Pi. We then imported a Python library called Pymata3 (dependent on Python v. 3.5) so we could interface with the slaved Arduino. One thing to note is that the Firmata sketch that is loaded onto the Arduino, along with the Firmata libraries, all must have added support for specific devices. For specialized devices, such as the Pixy camera, an extended library has to be used. Namely, we needed to use the FirmataPlusRBPixy sketch found within Pymata3's documentation.

Class Definitions

For development speed and ease of use, we decided to write all the code running on the Raspberry Pi in Python. We broke the code up into the following classes:

main.py - As named, this is the main script that is run on the Pi when it's booted. It's main job is threading two loops, the Flask server (explained later), and the robot driver.

driver.py - This class contains the logic for the state machine (explained later).

arduino.py - Initializes the Arduino pins for ping sensor, IR sensors, motor controller, and Pixy. Defines functions for interacting with this hardware.

consants.py - Declares constants used throughout the code.

navigation.py - Contains functions for physically moving the robot, including turning and wandering.

pixy.py - Implements functions for using the pixy, such as setting signatures and getting blocks from the pixy.

State Machine

Since the task that our robot was designed to complete was relatively simple and required few transitions between actions, we decided to contain the logic for our robot within a state machine running on the Raspberry Pi. This state machine was in charge of handling transitions between different algorithms designed to complete subsections of the overall task. Our final state machine can be seen in fig. 8.

Inter-Robot Communications

Coordination between robots was made possible by inserting a Wi-Fi dongle into each Robot's Raspberry Pi. Each robot then connected to the same network and hosted a web server on a designated IP address. We used Flask, a Python web development micro-framework, to host a communication endpoint. When other robots wanted to inform our robot about a completed action, they would send an HTTP GET request to our endpoint. This would then allow our Python code to trigger entrance into a new state.

Manual Web Page Control

As an additional feature, we designed our robot to respond to a custom built web page. This web page could swap our robot between autonomous mode and manual mode. Autonomous mode would force our robot to follow the above state machine, and manual mode would allow the keyboard up/down/left/right keys to drive. This web page was hosted on the Raspberry Pi using apache. A screenshot of the web page can be seen in fig. 7.

Released Code

All our code for this project has been released open source at: https://github.com/zacharylawrence/ENEE408I-Team-9.

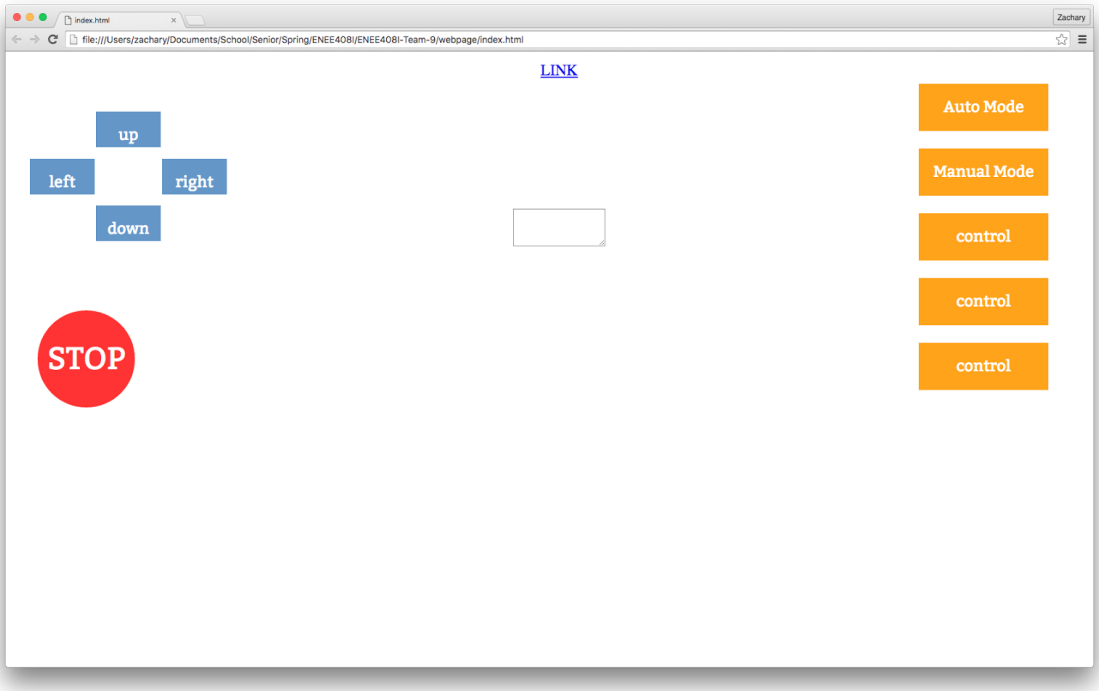


Figure 7: Manual Control Webpage

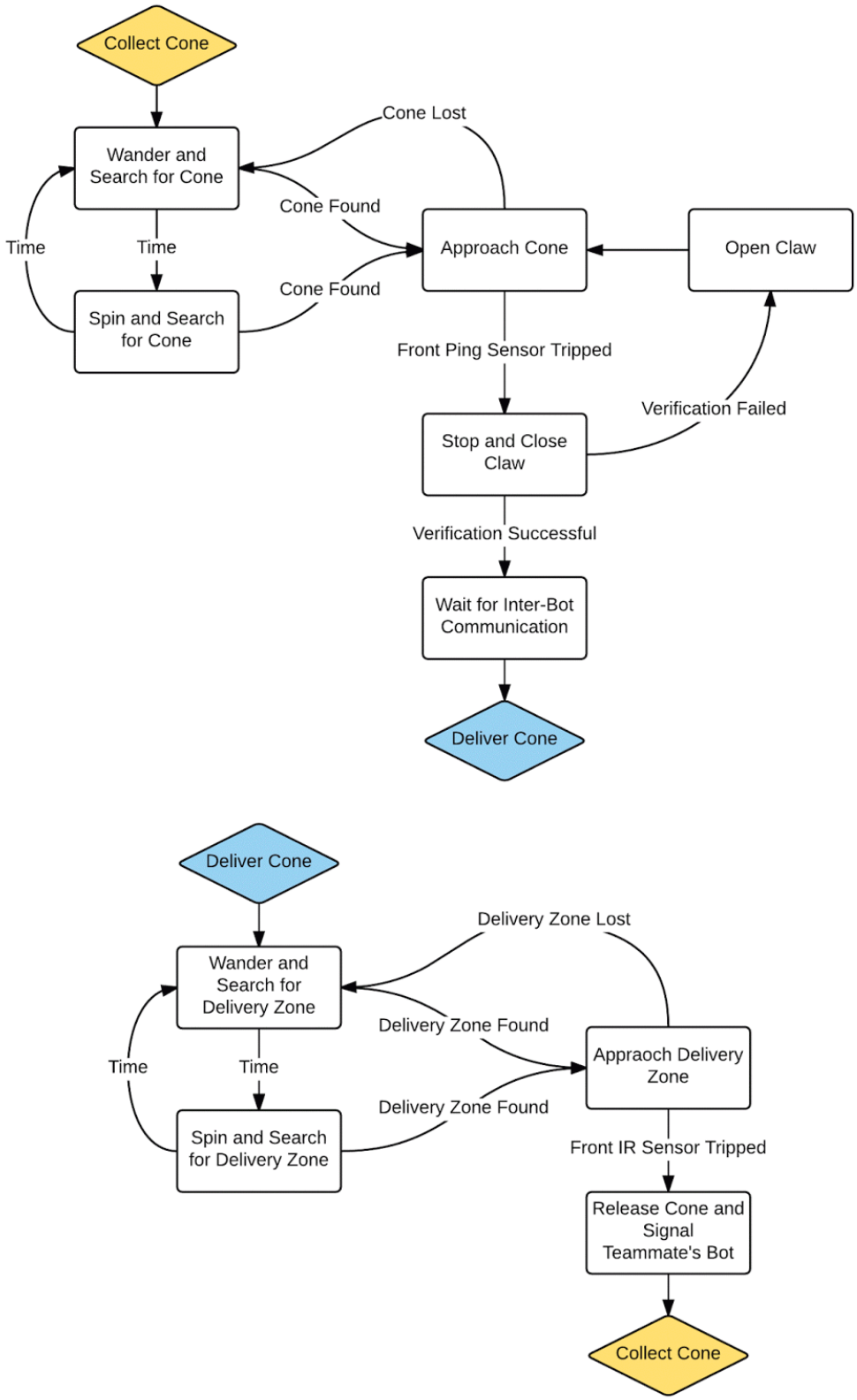


Figure 8: State Diagram

Performance Results

After completing all of the required tasks listed above, we had built a fully functional autonomous robot capable of completing a small task and communicating with other robots. We were able to work with two other robots (Group 8 and Group 10) to complete an entire run-through of the final objective. A video of all 3 robots working together can be seen here: https://youtu.be/JHeIsZMsnAo.

Although our robot would occasionally lose track of different objectives or run into other robots, we demonstrated our ability to analyze a problem, design a solution and implement a solution from scratch. Our robot was capable of collecting and dropping off a unique pattern of cone colors while talking to other robots doing the same. If we had more time during the semester, we would have focused on improving the reliability and robustness of our system.

Conclusions

We began this class by working in a group of 3 students design a fully autonomous robot capable of completing a small task and communicating with other robots. Using a variety of materials, sensors and electronics, we were able to build our designed robot from scratch. After constructing a fully functional autonomous robot, we were able to successfully compete in the competition, collect and deliver a unique pattern of cone colors and collaborate with 2 other robots.

We were also able to take away many lessons from this project. One of the largest lessons we learned from this class was the importance of designing your project before you begin to implement it. Although you invest time into design at the beginning of your project, it ultimately allows for you to save time by correctly implementing the ideal solution the first time. We also learned of keeping your project clean and organized. Both the electronics and codebase for this project could have easily become unwieldy; however, we focused on spending time to organize our work which allowed for us to later spend more time on fixing new problems rather than troubleshooting old ones.

Recommendations for Course

When looking forward to future iterations of this class, we recommend that the format of the class remains the same. Although a more in depth and comprehensive robot could have been constructed within our timeframe by starting with a base robot, this project helped teach that a robot can be built from parts without requiring a large amount of preliminary knowledge or money. However, we also believe that having a more focused competition outline at the beginning of the semester would allow for students to more quickly dive into designing a system for more advanced inter-bot communication.