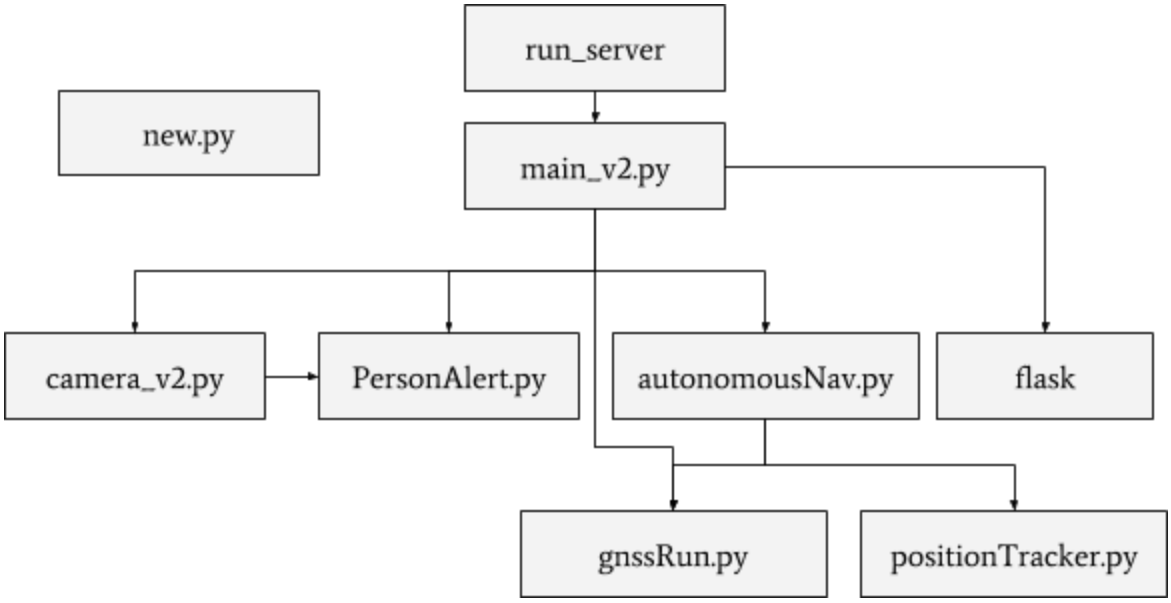
Software Report

1. Raspberry Pi (Flask, Python)

The code on the Raspberry Pi is all located in the UPR_Main folder on the desktop (the same folder is present in the GitHub repository). The bash script, run_server, shows what commands need to be run to start the program. The main_v2.py file is the central file used to run the project: it will call functions from other files to perform functions required by the application. It works closely with the camera_v2.py file to process video inputs from the RPi camera. To send alerts to the web server, the PersonAlert.py file is called. The gnssRun.py file contains convenience functions for working with the GPS and determining the position of the robot. The positionTracker.py file is responsible for most of the heavy lifting when it comes to finding position, and currently contains code for differential GPS corrections. The new.py file is an alternative to the main_v2.py that only implements the autonomous navigation portion of the robot. This file is good for testing the autonomous navigation functionality, because it does not have the complications of the other functions of the main_v2.py file. All other files in the UPR_Main folder are legacy code that is not being used.

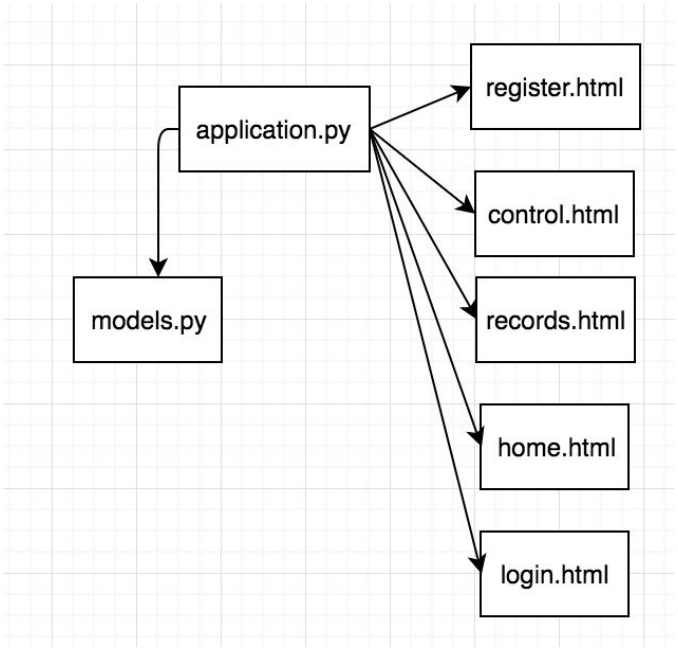


2. Arduino (Arduino-C)

The arduino code is all centralized within a single file, as is required by Arduino. This file is broken down into various functions, but the most important two functions are setup and loop. These two functions call every other aspect of the program and can be used to gain an understanding of how the program works.

3. Web Application (Flask, Python)

The Web App is build in the FLASK Stack, using primarily Python, HTML/CSS, and Javascript to build out its functionality. The file containing most of the functionality of the Web App is the application.py file, which holds all of the routes to the different pages, as well as an API to remotely receive Alerts from the Raspberry Pi. Additionally, there is a models.py file which creates all of the object types for each alert item, which allows the API to create a new object for each incoming alert. Finally, there are the HTML files which are used to create all of the displays for each of our pages. There is the control.html, home.html, login.html, records.html, and finally the register.html which is only used as a proof of concept to demonstrate adding login credentials for a new user.



4. Amazon Web Services

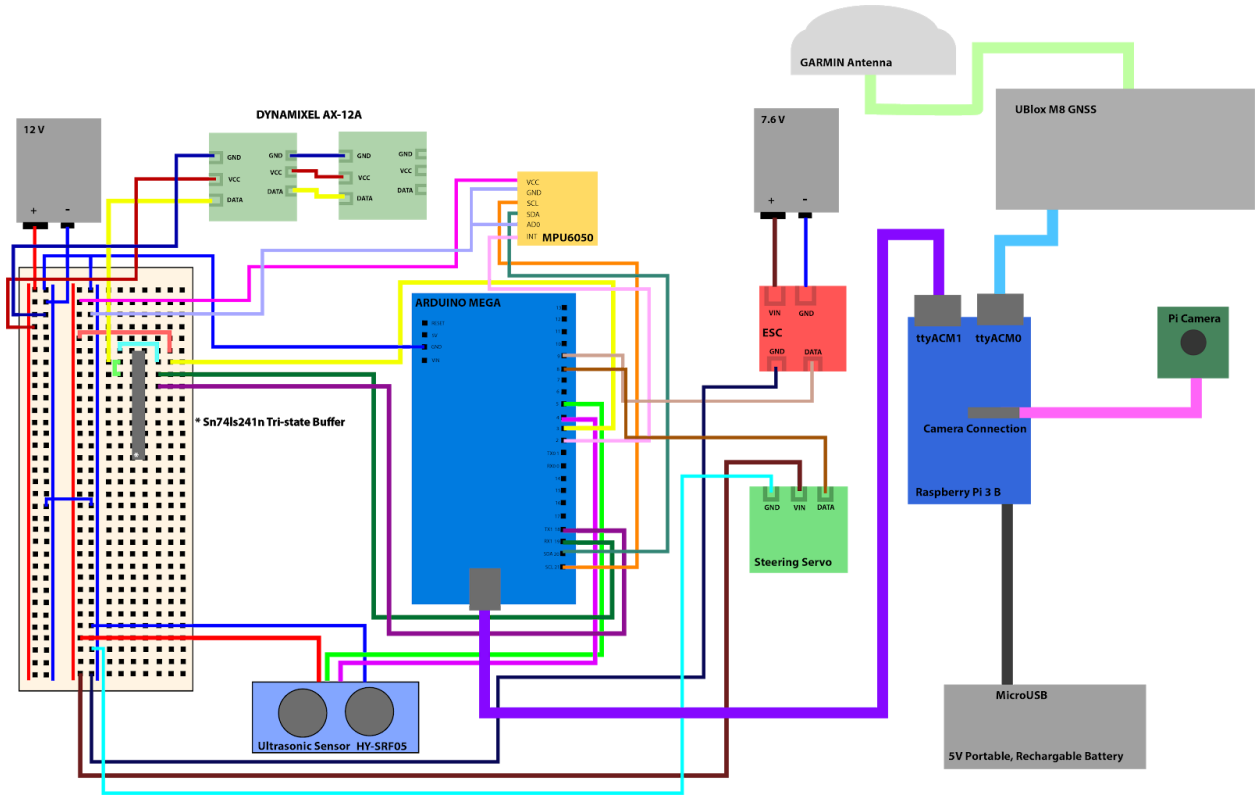
Amazon Web Services is used to host all services related to the web application, including the web app itself, video storage, and the relational database service. To run the web application on Amazon Web Services, an AWS account with the appropriate IAM security permissions is required. Once the IAM permissions are created, access keys will need to be generated on the Raspberry Pi, in the PersonAlert.py file. For the web application, the Elastic Beanstalk application (created using the AWS console and deployed to with either the AWS CLI or EB CLI) will need to be explicitly given permission to access S3 so that it can serve the patrol videos on the server. A Bucket will need to be created in S3 to host the video files created from patrols. There is a python script included with the web application code that will automatically create this Bucket when run. Amazon RDS, the relational database

service, will need to be used to host the SQL server that Flask will be using. To set up the AWS architecture, follow these steps:

1. Go to aws.amazon.com and sign in to the console (or create an account if needed).
2. Go to the following URL and follow the instructions to install the Elastic Beanstalk Command Line Interface: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html
3. Use the following instruction to create your EB instance (run these commands from the WebApp/ folder of the repository to ensure the project is created succesfully): https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-configuration.html
4. Use the following link for any further EB CLI references: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3.html
5. To set up the RDS, go to Services > RDS in the AWS Console (aws.amazon.com).
6. Select “Launch Instance” and choose MySQL as the type. Select “Only enable options for Free Usage Tier” and follow the rest of the setup prompt.

Hardware Report

1. HARDWARE SCHEMATIC



2. DEVELOPMENTAL TOOLS

2.1 Arduino:

To program the Arduino, we needed the Arduino Software (IDE), an open-source application made by Arduino. This was the easiest application to download, all one needs to do is follow the main Arduino page to the Software section where one can download the software, then install it onto their machine. Link is: https://www.arduino.cc/en/Main/Software. The more difficult parts was downloading the required libraries for our programs. Mostly if a library can be downloaded online in a .zip file, the IDE allows for a user to easily include the .zip file. All one needs to do is go to Sketch -> Include Library -> Add .zip Library and the IDE places it in the Arduino/Libraries folder.

2.2 Python 3 and Virtual Environment

On the Raspberry Pi 3 B, we ran the following commands on terminal to get python 3 and use a virtual environment when running OpenCV

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
$ export WORKON_HOME=$HOME/.virtualenvs
$ source /usr/local/bin/virtualenvwrapper.sh
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
$ mkvirtualenv cv -p python3
```

To get on the virtual environment run these commands:

```
$ source ~/.profile
$ workon cv
```

2.3 OpenCV

On the Raspberry Pi 3 B, we ran these commands on the terminal to get the OpenCV libraries for python.

```
$ wget -O opencv-3.1.0.zip https://github.com/opencv/opencv/archive/3.1.0.zip
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/3.1.0.zip
$ unzip opencv_contrib.zip
$ unzip opencv-3.1.0.zip
$ cd opencv-3.1.0
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D INSTALL_C_EXAMPLES=OFF -D
BUILD_EXAMPLES=ON -D WITH_GTK=ON -D WITH_FFMPEG=OFF  -D
```

OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules
-DENABLE_PRECOMPILED_HEADERS=OFF ..
$ make
$ sudo make install

3. VENDOR INFORMATION/BILL OF MATERIALS

3.1 Vendors

3.1.1 Micro Center

730 Memorial Drive, Cambridge, MA 02139 (617) 234-6400

Micro Center is a chain of stores that provide various electronic parts from soldering wire to GPUs to TVs. We purchased the Arduino Mega, Raspberry Pi 3 B, HY-SRF05 Ultrasonic Sensor, and Raspberry Pi 3 Camera Module here.

3.1.2 Sebastijan Mrak

Mr. Mrak was able to graciously loan us his U-Blox M8 GNSS and Garmin Antenna.

3.1.3 Adafruit

Adafruit Industries is a manufacturer of various chips and modules that work with programmable microcontrollers such as Raspberry Pi and Arduino. We purchased the MPU6050 from here. It does not seem to be on their website anymore. We also purchased the 5V 2A Battery from there.

3.1.4 Professor Little

Professor Little was gracious enough to lend us the Exceed RC Rock Crawler Radio Truck which he uses for his class EC544: Networking of the Physical World.

3.1.5 SuperDroid Robots

SuperDroid robots designs, builds, assembles, programs, and tests all custom robots and parts to be sold. We purchased the tri-state buffer needed for the Dynamixel Servos from there.

Item	Description	Cost
1	Exceed RC Rock Crawler Radio Truck	$200
2	MPU6050	$6
3	HY-SRF05 Ultrasonic Sensor	$4
4	Raspberry Pi 3 Camera Module	$30
5	Sn74ls241n Tri-state Buffer	$2
6	Raspberry Pi 3 B	$30

7	Dynamixel AX-12A Smart Servo Motor x2	$90
8	U-Blox M8 GNSS	$179
9	Garmin Antenna	$57
10	Arduino Mega	$38
11	Battery	$40
	Total	**$676**

4. POWER REQUIREMENTS

We used three different batteries for this project because everything ran on different voltages and using one battery with voltage dividers would have killed our batteries very quickly. In the future we would get a better battery and just use one for everything.

Requirements:

Description	Voltage
MPU6050	5V
HY-SRF05 Ultrasonic Sensor	5V
Raspberry Pi 3 Camera Module	5V
Sn74ls241n Tri-state Buffer	5V
Raspberry Pi 3 B	5V
Dynamixel AX-12A Smart Servo Motor x2	9-12V
U-Blox M8 GNSS	3-3.6V
Garmin Antenna	4.5V
Arduino Mega	7-12V
Exceed RC Rock Crawler Servo	5V

We used a 12 V Battery to power the Arduino Mega and the Dynamixel Servos, each receiving 10-11 V. We used a portable, rechargeable battery to power the Raspberry Pi which then powers the Camera Module and the U-Blox M8. The Arduino powers the MPU6050,Ultrasonic

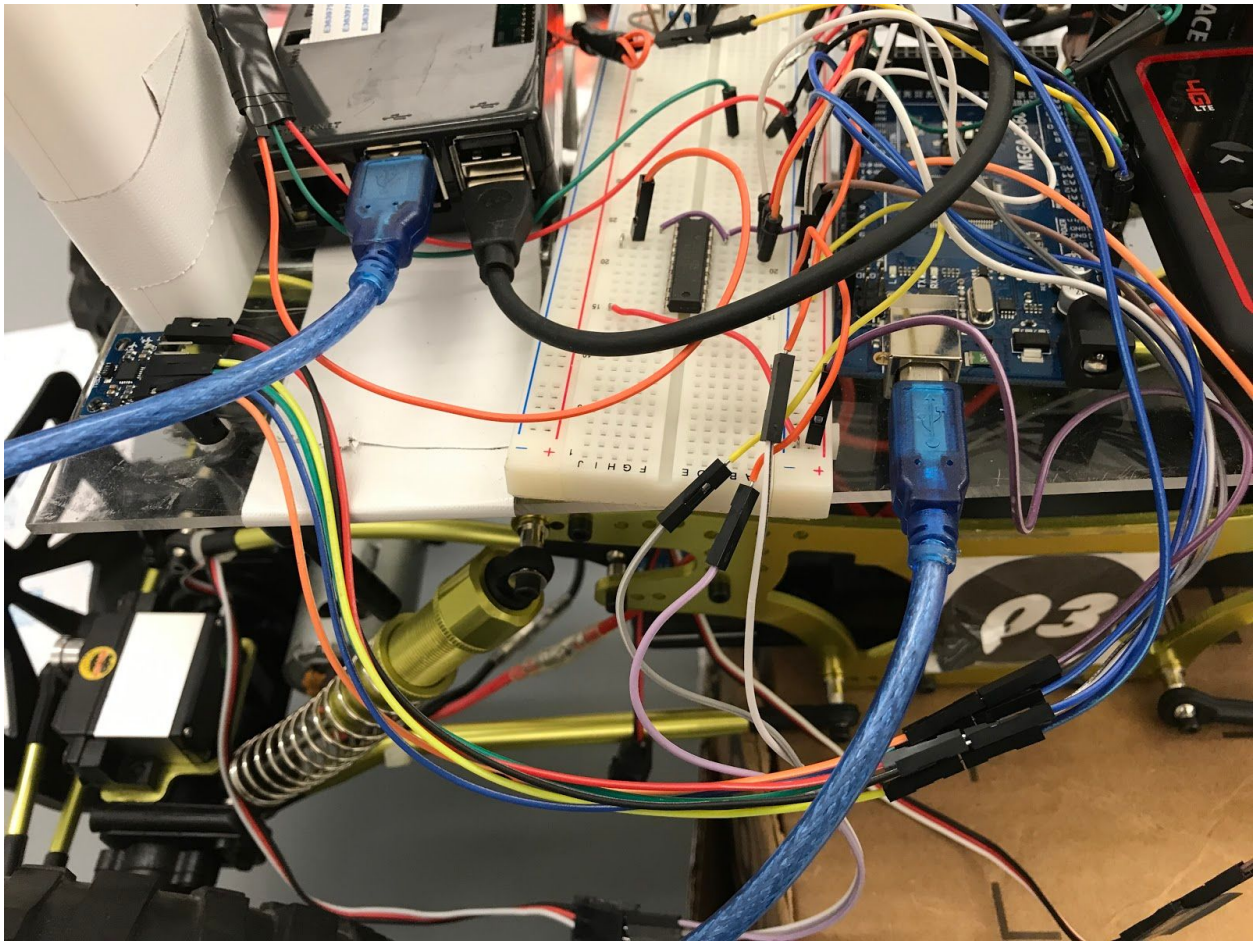
Sensor and Rock Crawler Servo. The ESC that controlled the DC motors on the Rock Crawler was powered by a 7.2V rechargeable battery.

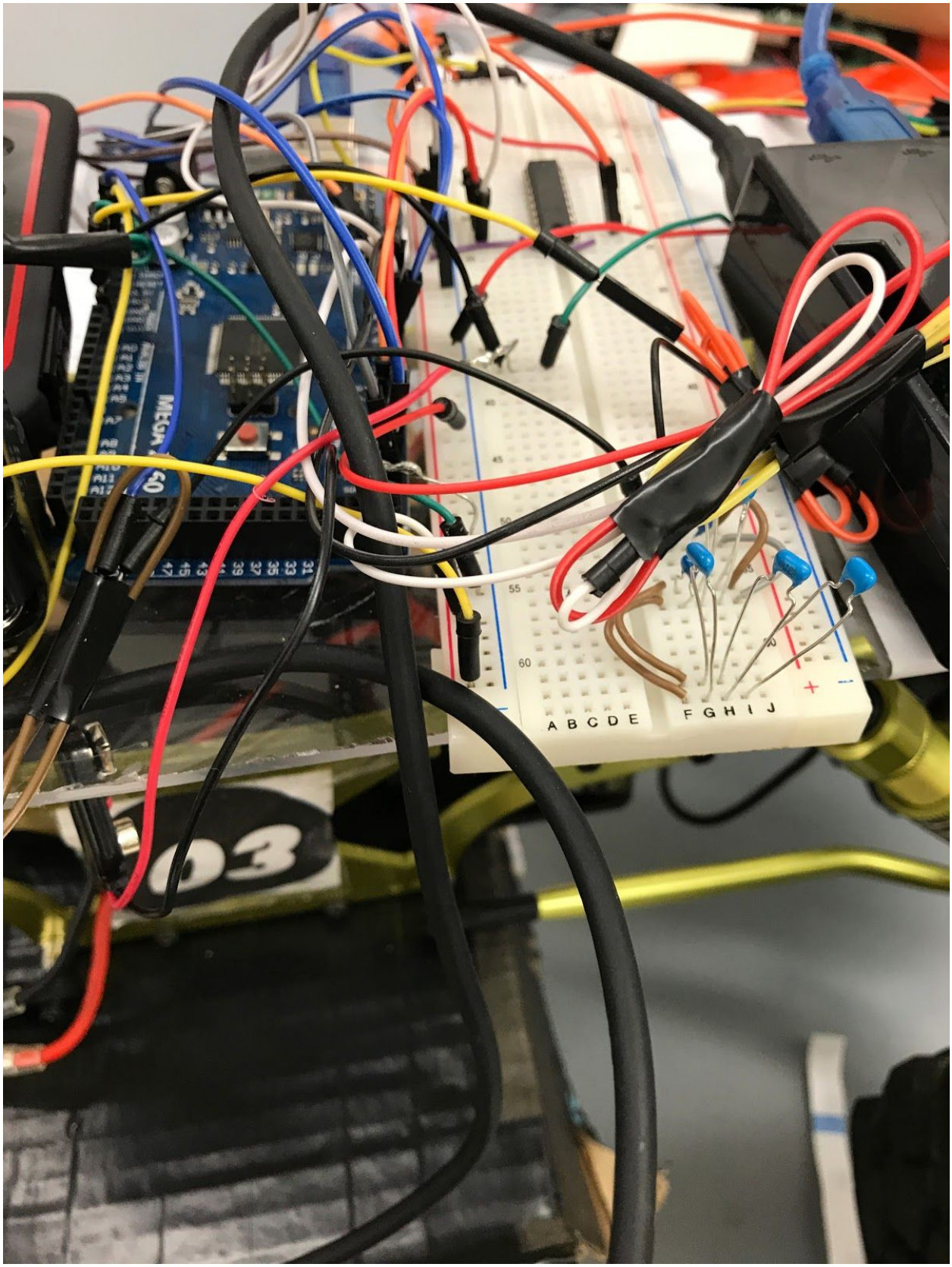
5. POWER SEQUENCE

To properly start up the patrol robot we follow these steps:

1. Power the ESC and allow it to calibrate without being connected to the Arduino
2. Power the Raspberry Pi and the connected components with the battery
3. Power the Arduino, the Dynamixel Servos, and the parts powered by the Arduino

6. PICTURE OF CONNECTED BOARD




A B C D E
F G H I J
55
60

7. DATA SHEETS AND RESOURCES

MPU6050

-https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

HY-SRF05 Ultrasonic Sensor

-https://www.robot-electronics.co.uk/htm/srf05tech.htm

Raspberry Pi 3 Camera Module

-https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

Sn74ls241n Tri-state Buffer

-http://www.ti.com/lit/ds/symlink/sn74auc1g126.pdf

Raspberry Pi 3 B

-https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

Dynamixel AX-12A Smart Servo Motor

-http://support.robotis.com/en/product/actuator/dynamixel/ax_series/dxl_ax_actuator.htm

-https://sourceforge.net/projects/dynamixelforarduino/files/?source=navbar

U-Blox M8 GNSS

-https://www.u-blox.com/sites/default/files/MAX-M8-FW3_DataSheet_%28UBX-15031506%29.pdf

Garmin Antenna

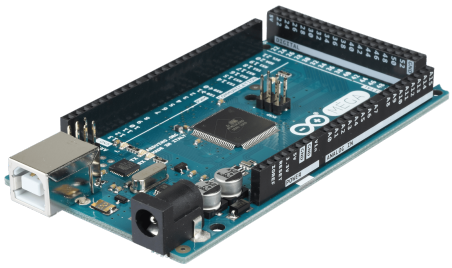
-http://static.garmin.com/pumac/GA35GPS_WAAS-antenna_AntennaInstallationInstructions.pdf

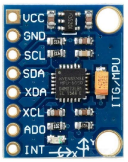
Exceed RC Rock Crawler Radio Truck

8. PHOTOGRAPHS OF SIGNIFICANT UNITS

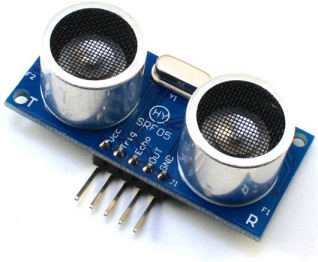
Full Project

Arduino Mega

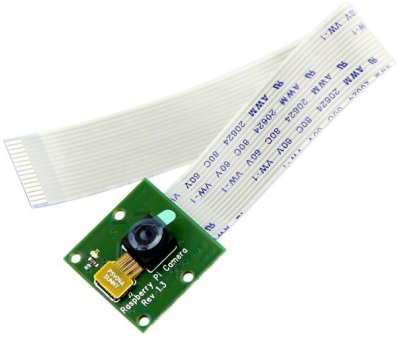




MPU6050



HY-SRF05 Ultrasonic Sensor



Camera Module

Tri-state Buffer

Raspberry Pi 3 B

Dynamixel Servo Motor

U-Blox M8 GNSS

Garmin Antenna