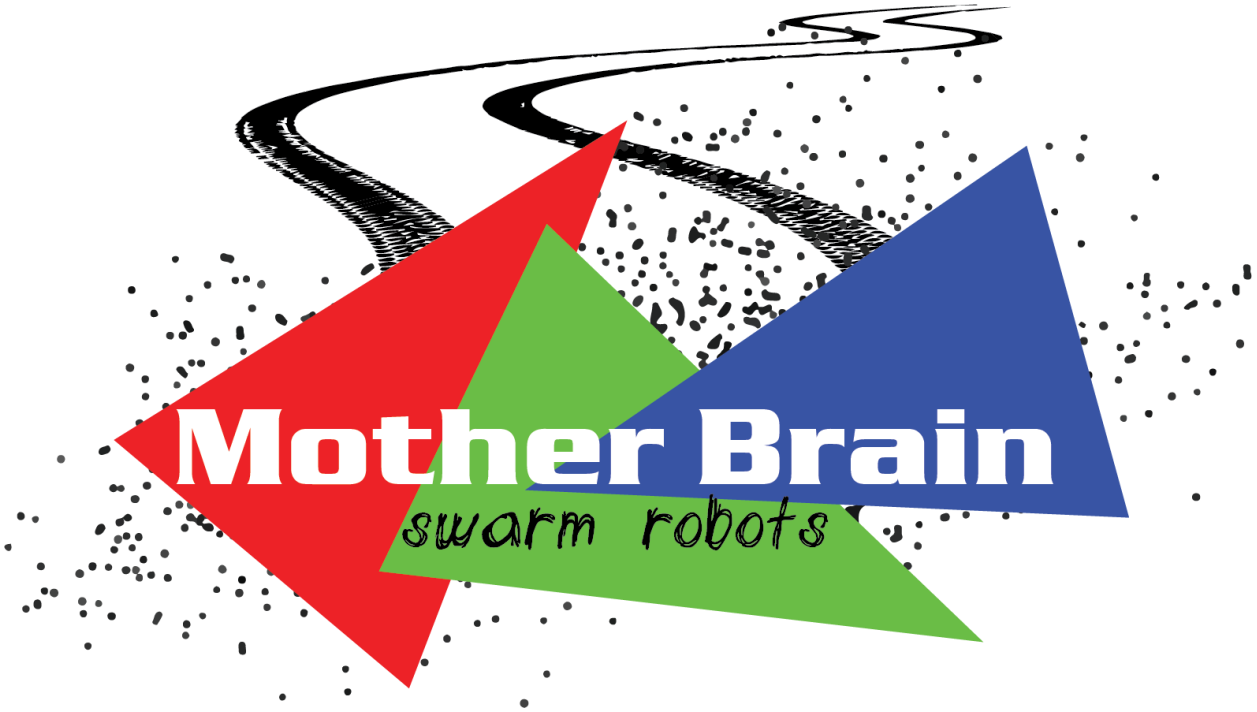


By

Mytch Johnson (EE)

Matthew Ng (CPE)

Darius Holmgren (CPE)

Tam Van (CPE)

Advisor

John Seng

Spring 2017

Table of Contents

Introduction

Since the Industrial Revolution, humans have been using technology and automation to ease their lives. Some notable examples include the assembly lines of the Ford Model-T in the 1920's and the production of the first drive direct arms in the 1980's. The growth of automation has transformed the manufacturing industry, making assembly quicker and cheaper. With assembly lines of robotic-driven automation replacing work that was previously done by humans, it continues to grow even today,

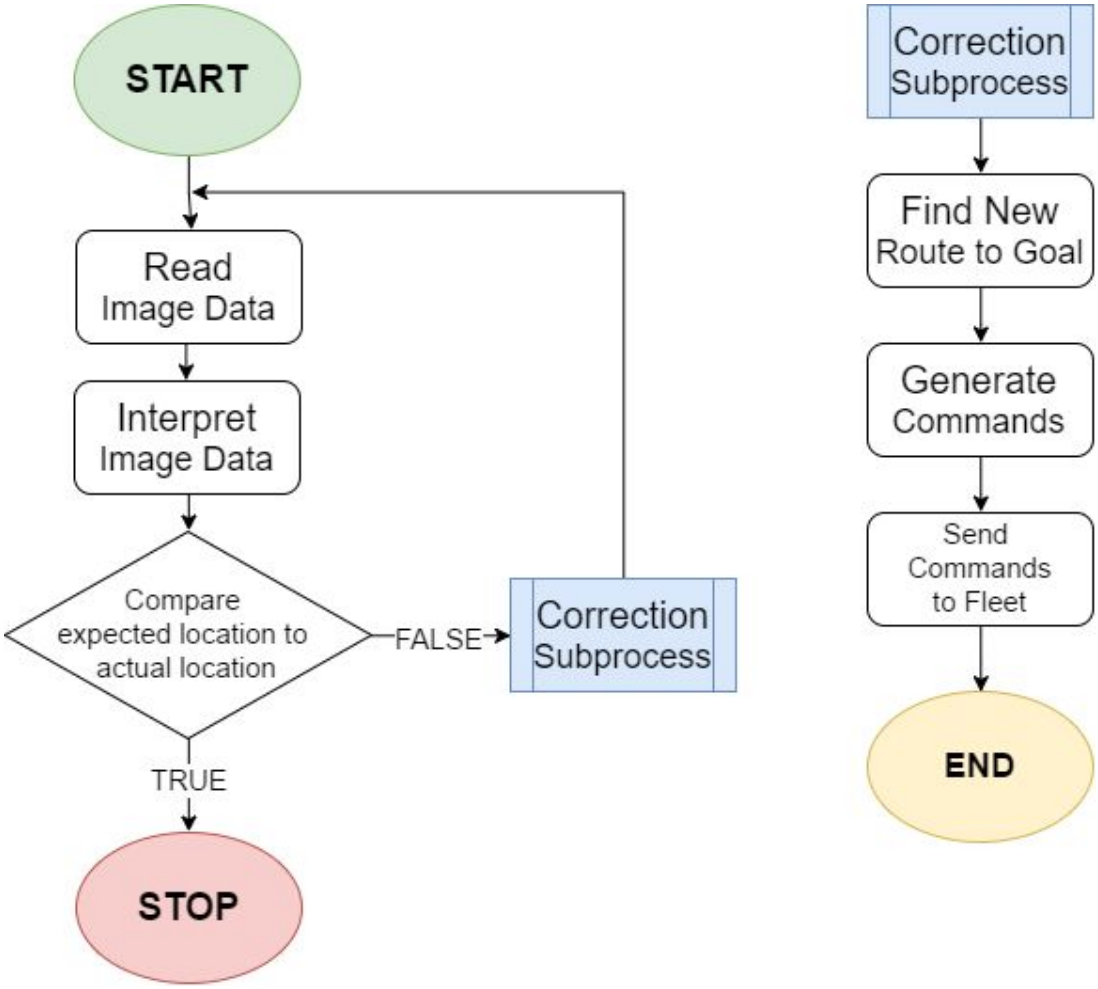
A modern example of problem solving using automation is Amazon's Kiva robots. Amazon, one of the world's largest e-commerce companies, uses a large system of automated packaging robots. Even though these robots have identical hardware, they each perform semi-unique tasks every day. Their primary tasks is to move racks of packages around the warehouse floor to the locations where it needs to be delivered. These robots are not reprogrammed each time a new request is received. Instead, they are told what the task is and they are preprogrammed to know how to complete it. This is an example of modern day problem solving using automation.

Problem Statement

Having to program the steps of a complex job over many robots individually is a cumbersome task. This obstacle is the next big problem facing automation today. While companies like Amazon have pioneered fleet robot type autonomy, there is still much to improve and a long way to go. Our senior project, MotherBrain, will be a simple case study looking into how a small-scale and simple centralized robot fleet might operate. The concept of MotherBrain is to create a simple way to program a fleet of semi-autonomous robots to complete a scalable task. This simple and scaleable task is simply moving to a location based on color and shape image data. Put simply, the robots in the fleet will automatically move to the color on a board that matches their own color. However, the robots do not sense the image data. Instead they receive commands from a central computer which reads and interprets the image data. This computer then decides what command to send the fleet in order to complete the task.

Software

Block Diagram



Flowchart 1: High Level software Process Chart

The MotherBrain computer was designed to perceive, interpret, and issue commands to the fleet. Because of this, the firmware on the robots is made to be as simple as possible. The transmitter module and all of the robots are programmed using C, the RF24.h library, and the Arduino library. The transmitter reads input from its standard input serial data stream compresses it, and then sends through SPI to the RF24 transceiver. The robots read in the data and interpret each piece of the data in the command. The RF packets are constructed into four byte chunks. The first chunk indicates which robot the command was meant for. If the robot number matches the sent command, the robot will actuate its motor speed and direction to the value sent in the second and third bytes. The fourth and final byte in the RF packet tells the robot whether this command should be held forever, or for only some fraction of a second.

If the packet was successfully transmitted to a receiving robot, an acknowledgement is sent back to the sender. If the sender does not receive an acknowledge after a certain amount of time, it tries to send the message again for a certain amount of retries. We configured the timeout to be 750us, and to retry up to 15 times. This gave us the best success rate in high noise environments and over the longest distances even in low power transmission modes. If an acknowledgement has still not been received after 15 attempts, a message is passed back to the user interface that the sending failed.

Before interfacing with the camera, it was necessary to manually set the webcam's configuration. These configuration setting involved disabling automatic color balancing and automatic white balancing. Once the camera is properly configured, reading its frames is fast and easy. Each frame is read in, split into 3 narrow color bands of red green and blue, then processed for contours. Those contours allow the AI to distinguish the robots as well as their targets.

UI Description

The user interface and computer vision work is all coded in Python3. The interface is created using the Python tkinter library. This library makes laying out the video stream, data stream, and buttons simple and structured. Packing the modules is as simple as calling the tkinter function to create a button and telling it what function to call when it is pressed.

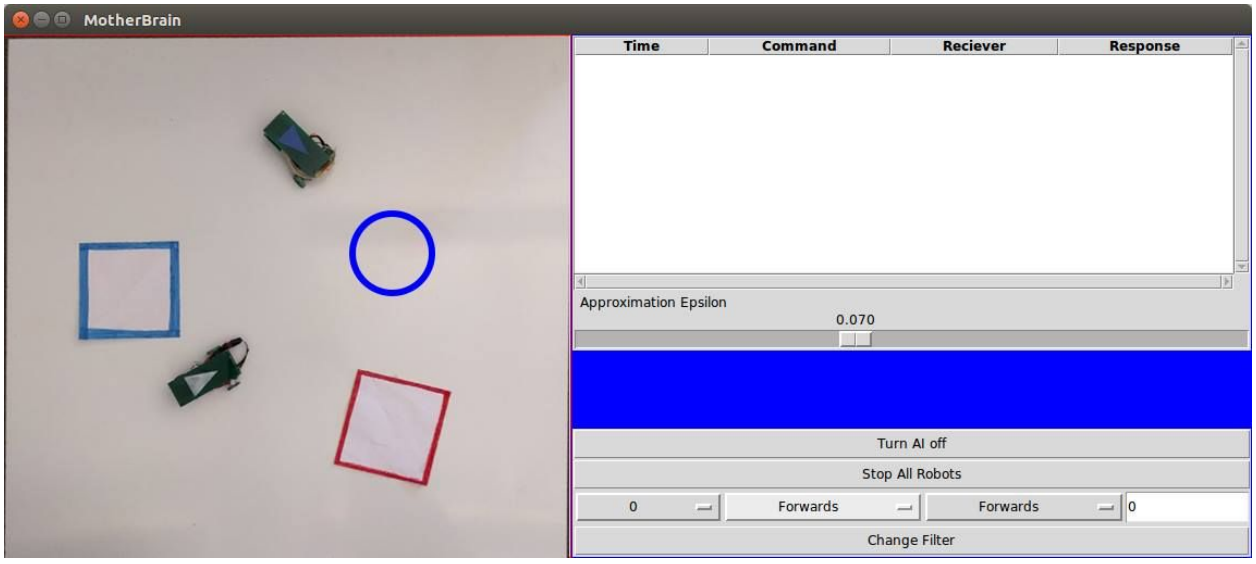


Image 1: UI Layout and Interface

The left side of the UI allows the user to see what the camera sees. This video stream can be toggled between the raw video with overlaid contours, and three preset color filters (red, green, and blue). In addition to the physical square targets that the robots home to, targets can be created for individual robots by left, right, or middle clicking on the screen. A blue target can be seen on image 1 above as a blue circle. Once these targets are reached, they disappear, and the robot while drive to the next target in its queue.

The buttons on the right side of the screen allow the filter to be changed, custom commands to be sent, and the AI to be toggled. The data block at the top right prints the commands being sent to the robots in real time as well as their response code in case of transmission failure. The approximation epsilon defaults to 0.70, but can be manually changed. Epsilon is an approximation which controls how much the computer vision algorithm is allowed to approximate the contours of a shape to minimize its edges. Therefore, a slightly jagged or aliased image of a triangle can be approximated to improve its detection accuracy.

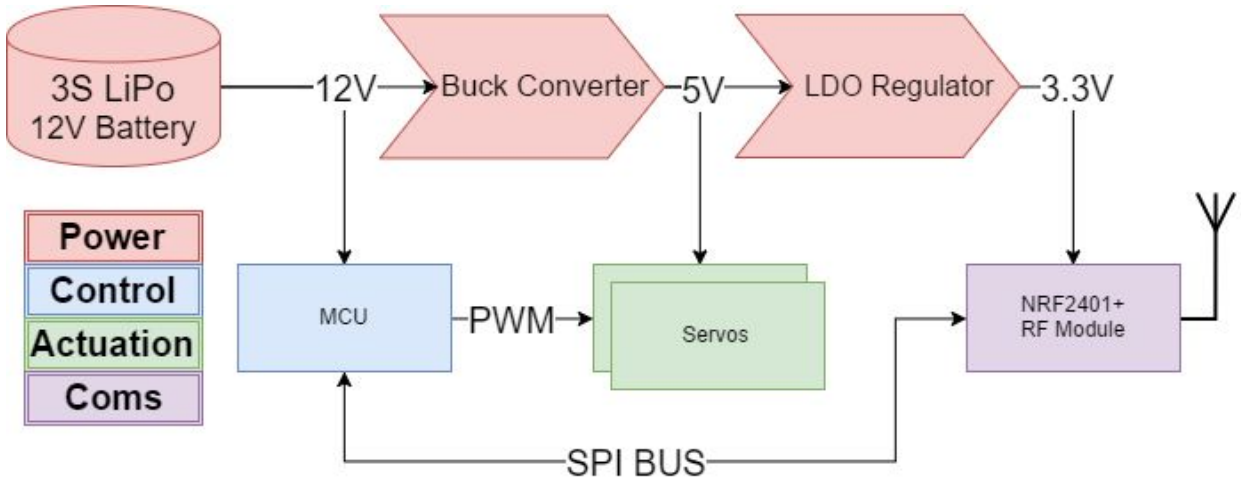
Vision Algorithm

We used OpenCV to capture the images from the webcam and used that image to find contours

1. Take a single frame with the camera
2. Blur the image to reduce the amount of noise while capturing
3. Isolate the image based on a color
4. Convert that isolated image to grayscale
5. Search for contours using cv2.findContours
6. Find contours with three sides
7. Exclude triangles that are too small
8. Calculate the lengths of the sides to determine the shortest side of the isosceles triangle
9. Determine robot angle from isosceles triangle sides

Hardware

Block Diagrams



Block Diagram 2: MotherBrain Robot Hardware System

Power

The power system for the robots consists of one 3S LiPo battery source and a three stages power regulator framework. Power regulation is necessary because LiPo batteries are an unregulated power source. They have a nominal charge of 11.1V, but can charge to as high as 12.6V and discharge to as low as 9V. The first stage feeds the unregulated LiPo battery into the microcontroller. The mcu regulates the voltage down to 5V (low power) with its internal LDO voltage regulator. The second stage bucks the unregulated LiPo to 5V through a highly efficient 3 amp buck converter. This high power 5V source is used to power the servo wheels and RF module. The third and final stage feeds the high power 5V source to an LDO which regulates it into a 3.3V 1 amp source. A high power 3.3V source is necessary to power the RF module with sufficient power for high signal-to-noise ratio communication channels.

Control

The microcontroller controlling the robots is a ATmega328 based Arduino nano. This microcontroller was chosen for its small form factor and easy programmability. Its main purpose is to interface with the NRF2401+ RF module which communicate bi-directionally with the mcu over SPI. The signals sent through the RF module is interpreted as a move command. The microcontroller interprets the signal and executes that move command with a PWM signal(s) sent to the two servo wheels.

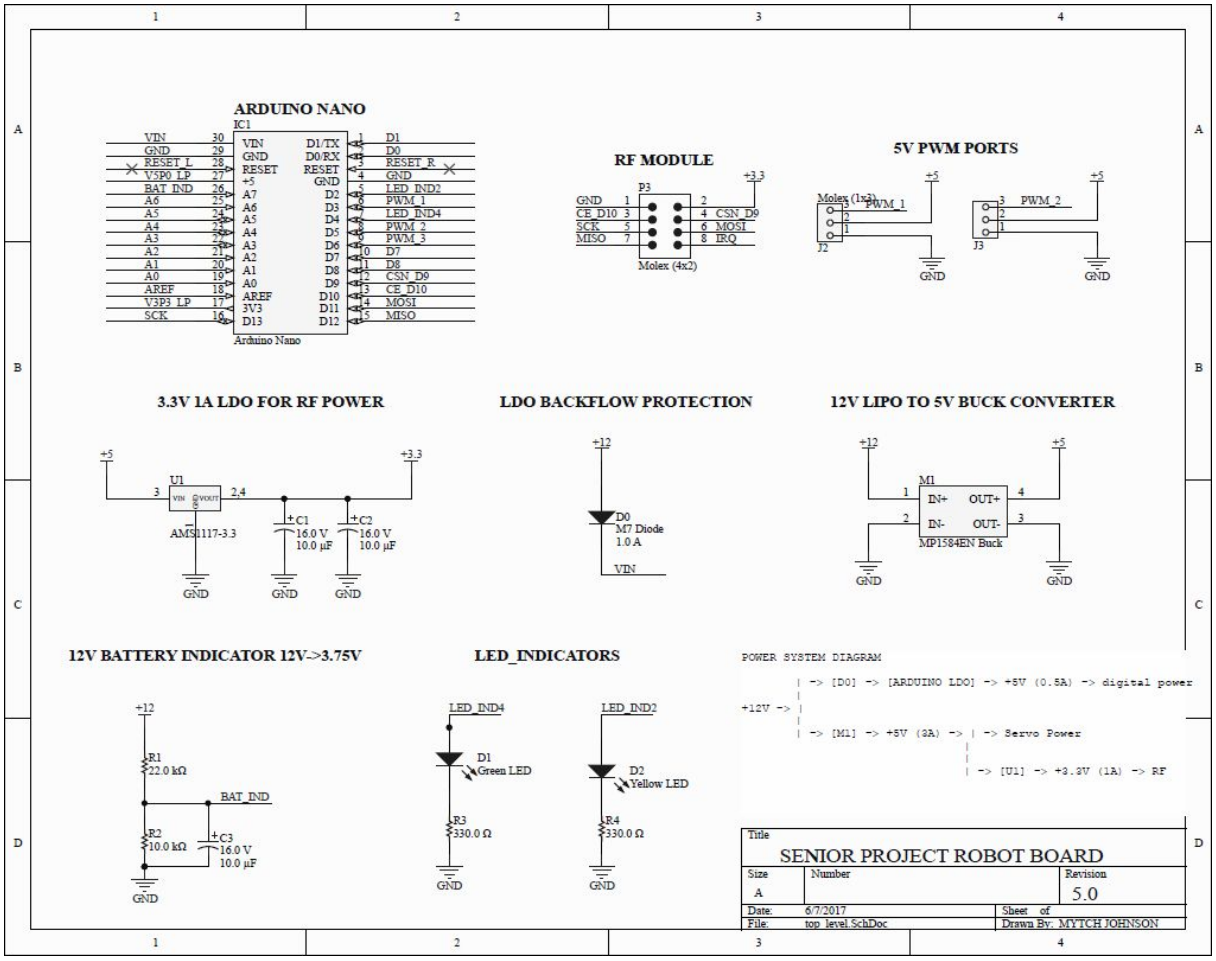
Actuation

There are only two actuators used by the robots. Both actuators are continuous rotation servos which move the robot. Because of their relatively high power usage, these servos are powered from a high power 5V source and not the low power microcontroller source. Each wheel is controlled with a PWM signal sent by the microcontroller which dictates its speed and direction.

Communication

Communication with the robots and the MotherBrain is done over a 2.4GHz RF channel. This channel is interfaced with by the NRF2401+ module on the robots and the MotherBrain computer. Each robot in the fleet has a unique channel number which is sent at the top is the RF packet. Once the signal is decompressed, the robot check's if the command was sent to its own channel. Channel 0x00 is reserved as a flooding every robot in the fleet. The packets are structured as such: [8 bit channel number | 16 bit speed command number | 8 bit time argument].

Schematic



Schematic 1: Robot Schematic Rev 5

Mechanical

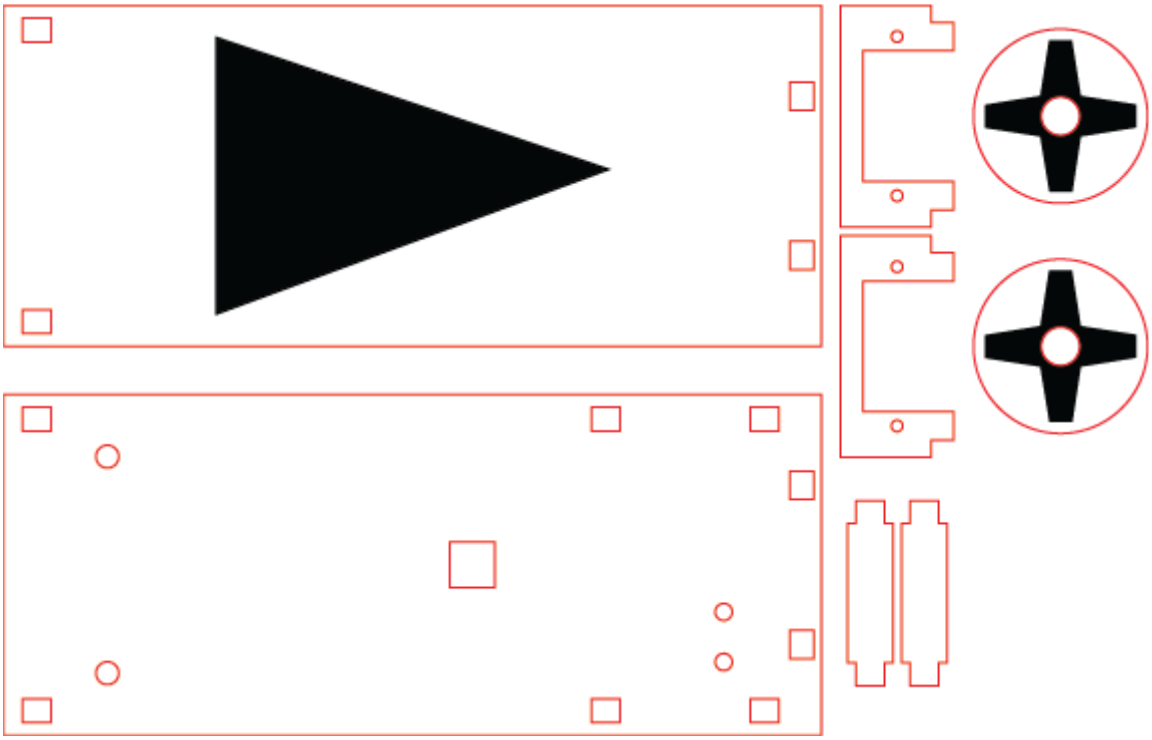


Image 2: Adobe Illustrator Chassis Design

Chassis

This is the final revision of the mechanical chassis which gives clearance under the robot for the small servo and wheels to fit as well as the ball bearing on the other side. The etching on the wheels allow for the servo to have a “+” attachment and to fit and have grip on the wheels.

The isosceles triangle etched out on the top of the robot is used to fill with a certain color paint which allows the computer vision to detect the robot as well as the direction the robot is driving in. The isosceles triangle helps the computer vision detect which direction by determining the shortest side of the triangle and finding the perpendicular centerpoint.

Because we needed to have the triangle visible on top of the robot for the computer vision to detect, the robot chassis had to be extended so the PCB and battery could fit underneath. We wanted the chassis to be as small as possible and achieved the minimum given the hardware.

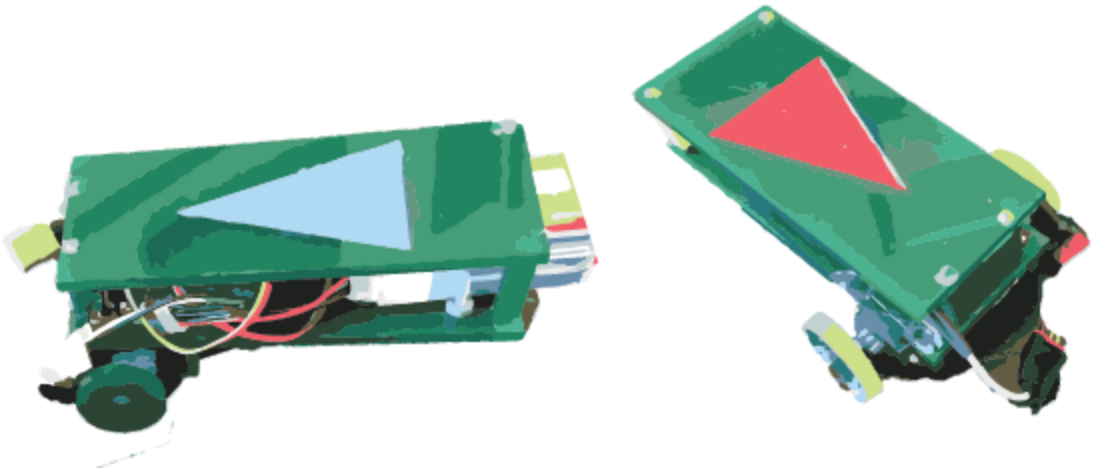


Image 3: Physical Robot Chassis and Build

Build

The aim for this construction was to not screw anything together besides a PCB holding screw. This was a bit more difficult than typical lasercutting projects like Roborodentia where the robot was a box. Standoffs, wheels, and etching to create the divot in the wheels and the triangle on top needed to be done. An additional issue we ran into was the standoffs were too small and snapped too easily. Multiple revisions had to be done on the robot to get it where it is now

The general build of the robots was a clam-shell case covering the main robot components in a consistent color and shape. The chassis was put together so that the major robot parts were placed in between the two main covers. These main components include the robot motherboard and the robot battery. It was important to cover all components with the top and bottom shell, so the chassis color could be ignored and only the colored triangles filtered through the CV algorithm(s).

The wheels are made of smooth laser cut plexiglass which did not allow for enough traction to move the robot in a consistent manner due to the wheels spinning out. Adhering rubber bands to the wheel hub, increased traction and allowed the robot to move in a consistent manner.

Bill of Materials

PCB				
Comment	Designator	Quantity	Price	Extended Price
10uF 16V capacitor	C1, C2, C3	3	$ 0.25	$ 0.75
M7 Diode	D0	1	$ 0.14	$ 0.14
Green LED	D1	1	$ 0.34	$ 0.34
Yellow LED	D2	1	$ 0.34	$ 0.34
Arduino Nano Molex	IC1	1	$ 0.42	$ 0.42
Molex (1x3)	J2, J3	2	$ 0.38	$ 0.76
MP1584EN Buck	M1	1	$ 1.67	$ 1.67
Molex (4x2)	P3	1	$ 0.38	$ 0.38
0805 22k resistor	R1	1	$ 0.10	$ 0.10
0805 10k resistor	R2	1	$ 0.10	$ 0.10
0805 330 resistor	R3, R4	2	$ 0.10	$ 0.20
AMS1117-3.3	U1	1	$ 0.80	$ 0.80
		TOTAL	$ 5.02	$ 6.00

ROBOT				
Comment	Designator	Quantity	Price	Extended Price
Arduino Nano	x	1	$ 4.00	$ 4.00
Servos	x	2	$ 1.80	$ 3.60
NRF2401+ Module	x	1	$ 1.20	$ 1.20
3S Lipo Battery	x	1	$ 13.50	$ 13.50
Screw Mounted Bearing	x	1	$ 2.40	$ 2.40
		TOTAL	$ 22.90	$ 24.70

COMPUTER				
Comment	Designator	Quantity	Price	Extended Price
Computer (2x core w/ GPU)	x	1	$ 300.00	$ 300.00
Webcam 720p	x	1	$ 14.00	$ 14.00
Webcam Stand	x	1	$ 16.00	$ 16.00
		TOTAL	$ 330.00	$ 330.00

TOTAL				
Comment	Designator	Quantity	Price	Extended Price
PCB	x	2	$ 6.00	$ 11.99
ROBOT	x	2	$ 24.70	$ 49.40
COMPUTER	x	1	$ 330.00	$ 330.00
		TOTAL	$ 360.70	$ 391.39

Lessons Learned

Mytch Johnson

I learned a lot about embedded system design, power system design, and PCB fabrication for this project. I spent most of my time designing and debugging the embedded system for the robot. It was a lot of trial and error to design a single sided PCB that could fit and route to everything within a reasonable amount of space. Even once the system was designed, there was a lot of debugging with the design. Most complication came from the multi-stage power system and I learned a lot about proper routing protection for a DC-DC system.

If I were to do anything different, I would skip right to SMT devices on our PCB as soon as possible. Through-hole devices have a tendency to wear quicker and the traces are more apt to lift with the PCB milling machine we ended up using to fabricate our PCBs. I would also added power protection circuitry earlier in the design process.

Darius Holmgren

I learned a lot about the implementation of computer vision and different ways of optimizing it. Using colors and contouring is a lot faster than I ever would have imagined. Trying to use multiple frames to remove artifacting and noise improved the image only moderately and proved to not be worth it. Processing the images in real time at a reasonable framerate was more important.

If I were to do this project again, I would have done more planning as far as creating a custom structure for holding abstracted data which would allow me to do more complex tasks extensibly instead of hacking functionality together.

Matthew Ng

I always imaged computer vision to be a daunting task. After working with some color detection and contouring and using the OpenCV library, I realized how well Python works at capturing frames and detecting colors based on their hue. I also learned how computer vision is extremely difficult when there are other colors in the area. During my freshman year, there was a roborodentia team that used computer vision to detect goals and shoot directly into them. This robot was amazing when there was no one else in the room. During competition however, their robot spazzed out like crazy. We ran into this issue as well when we moved into the ATL. The lights pointed straight down compared to the IEEE lights that pointed upwards diffusing the light. The computer vision saw those lights and if our robot drove under it, the light reflected off the robot would make the robot disappear.

If I were to do this again, I would want to create a course that could be under an awning that was lit evenly. I plan on working more with computer vision and image processing for my graduate thesis and hope to improve on this senior project idea.

Tam Van

I learned a good deal about computer vision and real time operations. It was fascinating to see how effective the computer vision was in detecting the colors and shapes on the robots. Filtering out noise was more computationally expensive and did not provide any significant improvements.

If I were to do this project again, I would have spent more time developing the computer vision. There are times when the computer vision would have difficulty detecting the robots if an ample amount of light was not present.

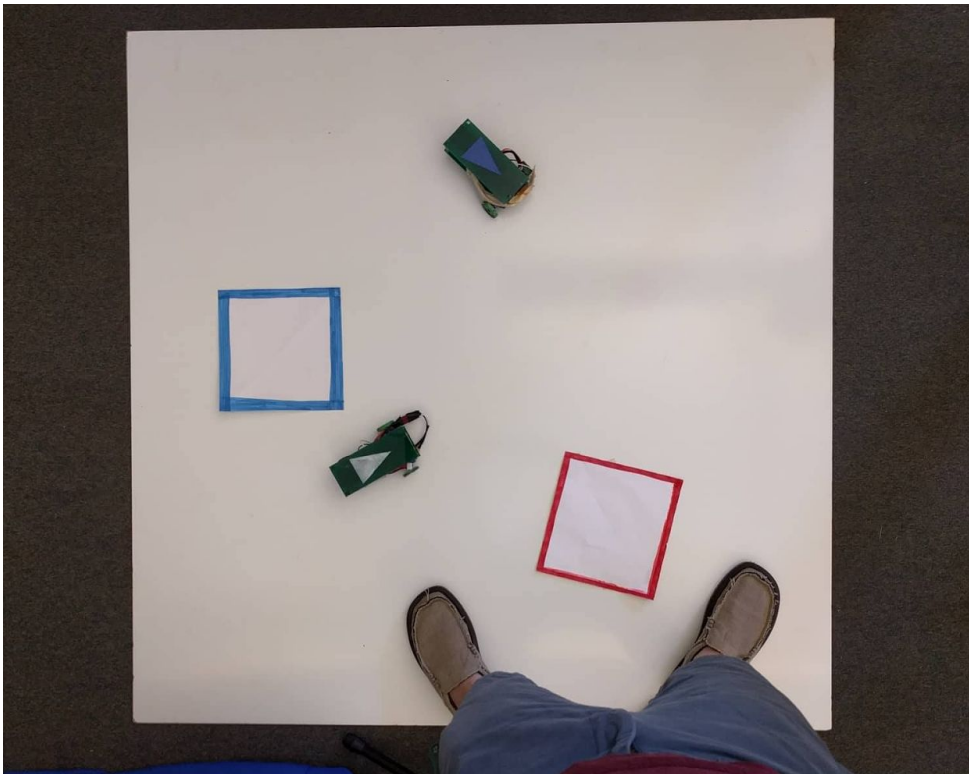
Conclusion

All in all, MotherBrain was a great exercise in computer vision, fleet robotics commutation, and small scale power electronics. It was interesting to build a fleet robotics approach from the ground up, and to see which ideas worked well and which ones did not. Even though the project's scope was limited to two robots, the software was theoretically able to scale up. The challenge was finding more colors to differentiate one robot from another. This project gave all four of us a taste of what a modern day robot might look like, and how to solve some of the more complex obstacles it faces.

Appendix

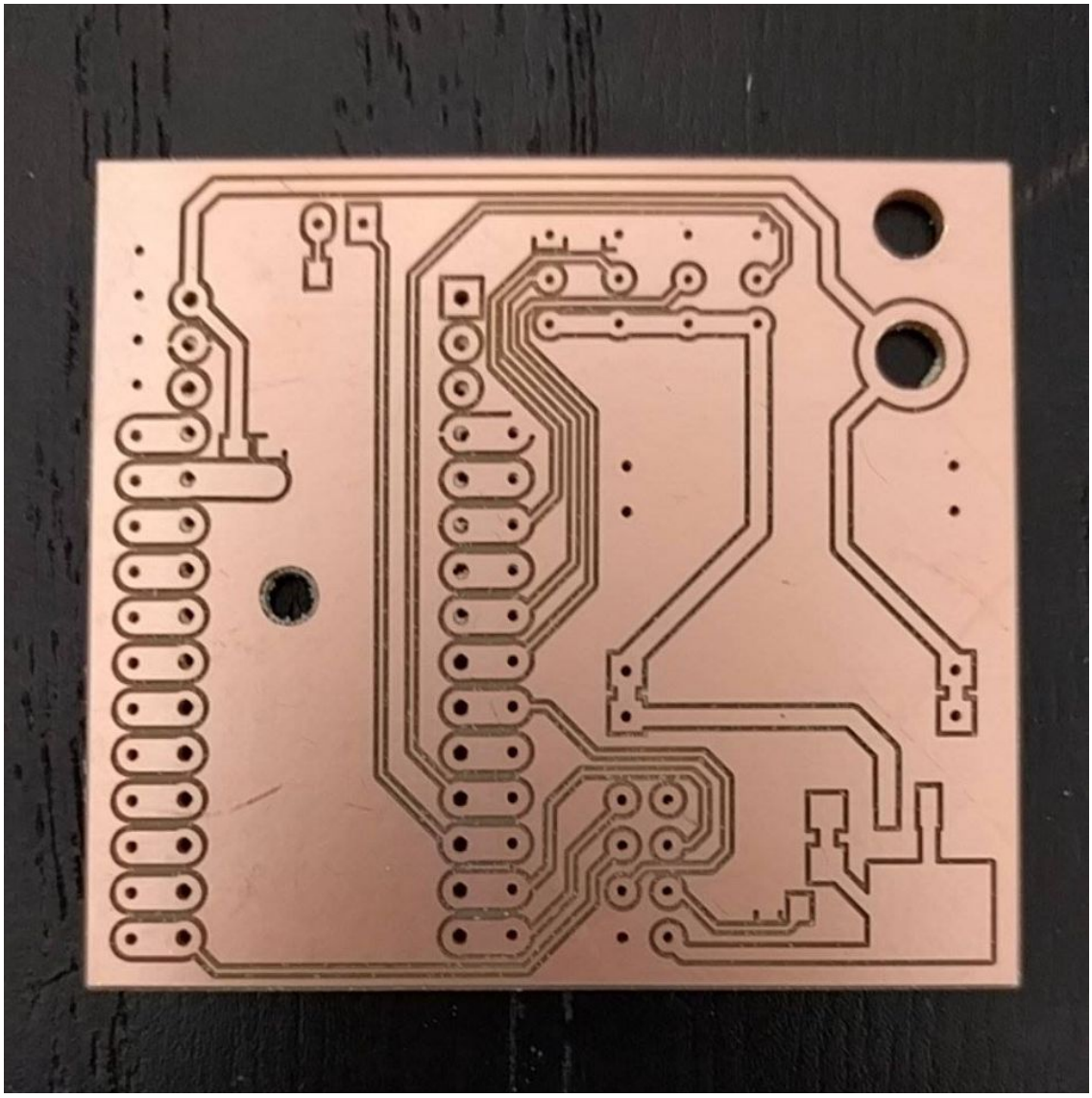
Link to Github Repo: https://github.com/DariusHolmgren/MotherBrain
RF24 library: https://maniacbug.github.io/RF24/classRF24.html

Photographs

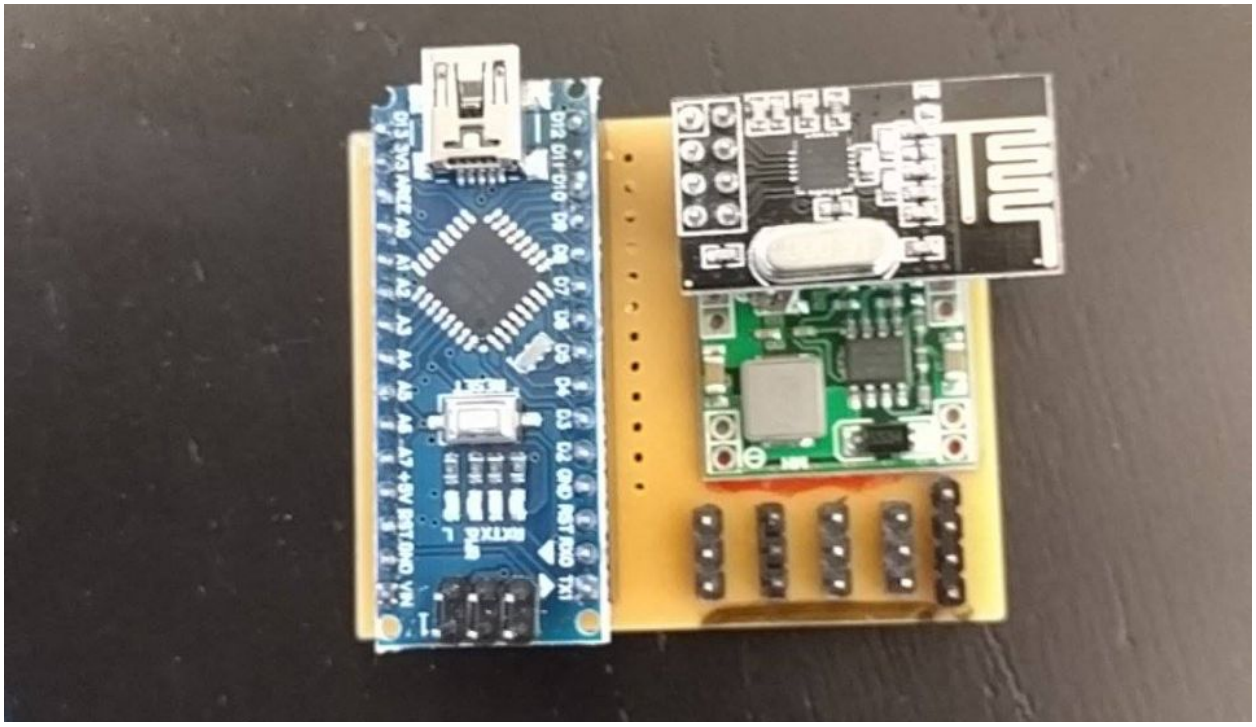


A top-down view of the board and robots.

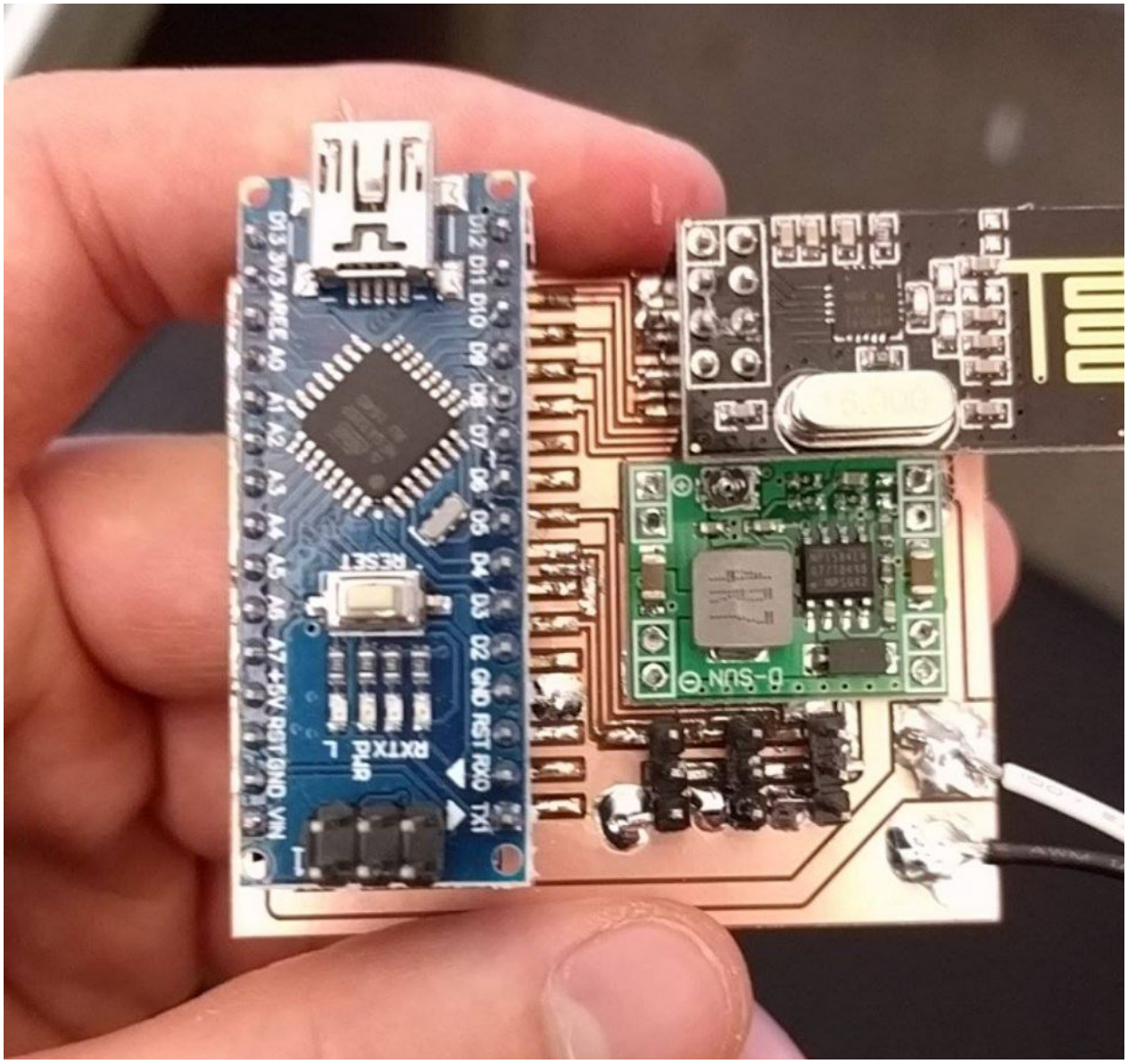
The microphone stand used to capture the top-down view of the board. A webcam is attached to the stand.



A through-hole iteration of the motherboard.



Version one of the robot motherboard (through-hole)



Final version of robot motherboard (surface-mounted)