Fast, Low-Cost Swarm Robots
ECE 445, Senior Design

By
Michael Bartmess,
Peter Cork,
And Paul Ernst

Abstract

Our project was to build a fleet of robotscapable of moving in formation. It uses a web camera to take images, which are then used to find our fabricated robots and tabulate their positions and orientations. Destination points are set and then paths are calculated for each individual robot to reach its target position. Each robot only required a circuit board, motors, and chassis, as the lack of sensors on our devices was a desired goal. Our project resulted in a group of robots able to independently move to create the formations specified by a user.

Table of Contents

1. Introduction:

By getting numerous robots to work together we created a platform to test route-planning algorithms using machine vision in real time for control purposes. We used the swarm robots to visualize transitions and shapes where each robot could move quickly and independently, with their positions tracked by a web camera. It was exciting to have the robots converge upon set and random destination points and see a physical demonstration of pathing.

While a robot tends to be very good at knowing its own state, it often lacks the environmental awareness required to coordinate its movements with others. Current methods of introducing this environmental awareness have been to add more sensors to the robot, but in doing so the price per robot increases. Our goal was to design and build a robot that is low-cost by reducing the number of sensors required to identify the position of the robot. In addition, we want to allow high precision and but also high responsiveness of the robot. Our method for doing this is to create a base-station that utilizes advances in machine vision to report the position of the swarm robots. In doing so we can thoroughly simplify the swarm robots.

There are two major components to the project, the hardware, and the Vision system parts. Figure 1 shows the blocks, which make up the hardware portion of our design.

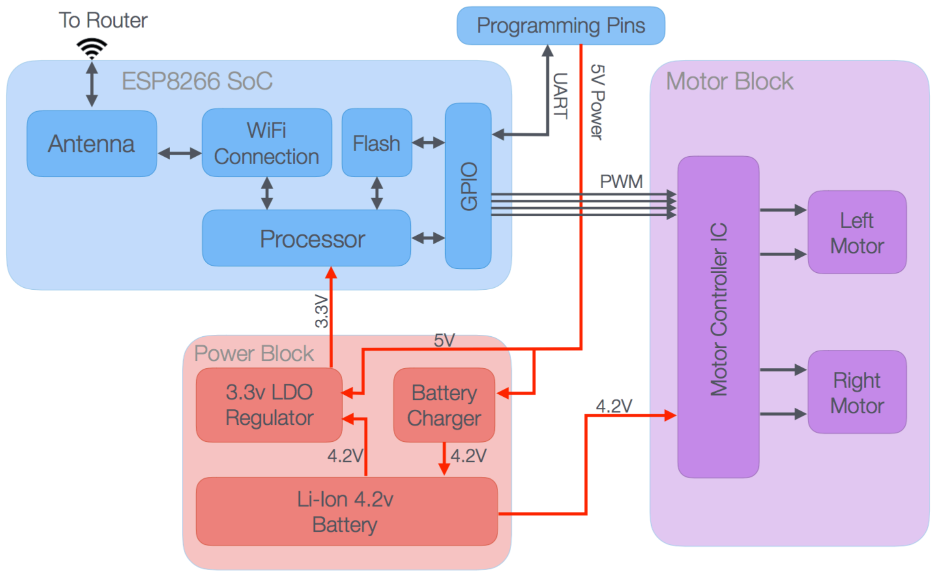

Figure 1 Hardware Block

The hardware portion of our design consisted of a Motor module, Power module and Wi-Fi module, which was also our system on chip (SoC). The SoC handled the communication with the router to receive packets, which contained the information necessary for storing and processing motor commands. The Soc outputted pulse width modulated signals to drive the motors using four outputs. The motor module was formed by a motor controller integrated circuit (IC), which could simultaneously drive two motors simultaneously, independently, and bidirectionally.
The power module regulated the voltage supplied to the rest of the circuit and consisted of a voltage regulator, battery and a battery charger.
The three hardware blocks worked together to make up the electromechanical part of the robot. The processor controlled the motors with the commands supplied by the software part of our project, and was supplied with power from the battery through the voltage regulator.

The second major component of our project the Vision System block is shown in Figure 2.

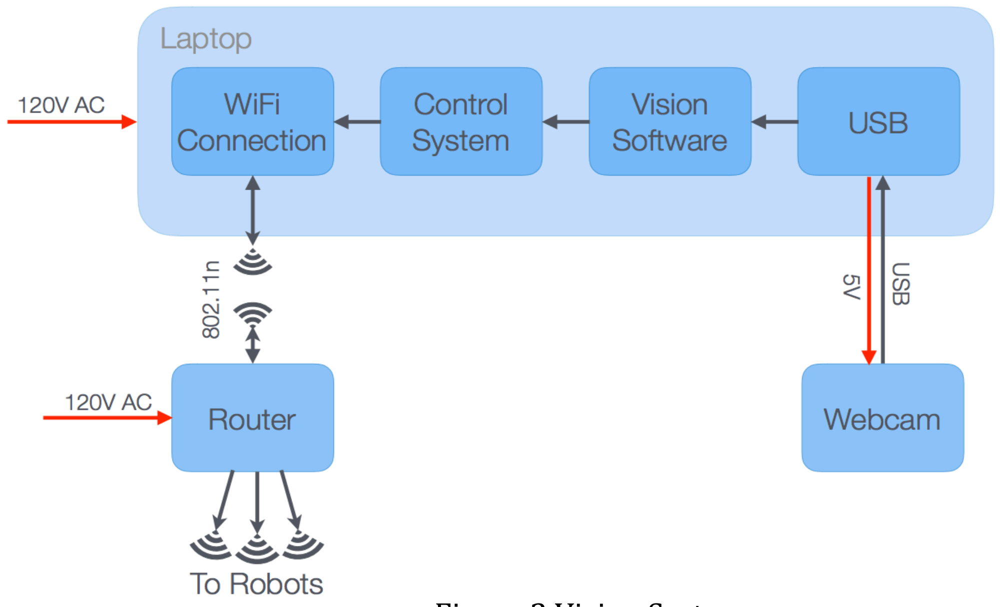

Figure 2 Vision System

The software block is made up of the webcam, a laptop, and the router. The laptop handled the USB connection to the webcam, the vision and control loop processing, and the wifi connection to the router.

2. Design:

Voltage Regulator: Overall the power block needed to provide the energy necessary to power the rest of the circuit. When we designed this block, we set it up so the circuit could be run by either a 5V source from the wall or from our lithium ion battery. To prevent the circuit from drawing power from both the battery and an external voltage source we placed a diode (D2) leading into the voltage regulator as shown in Figure 3

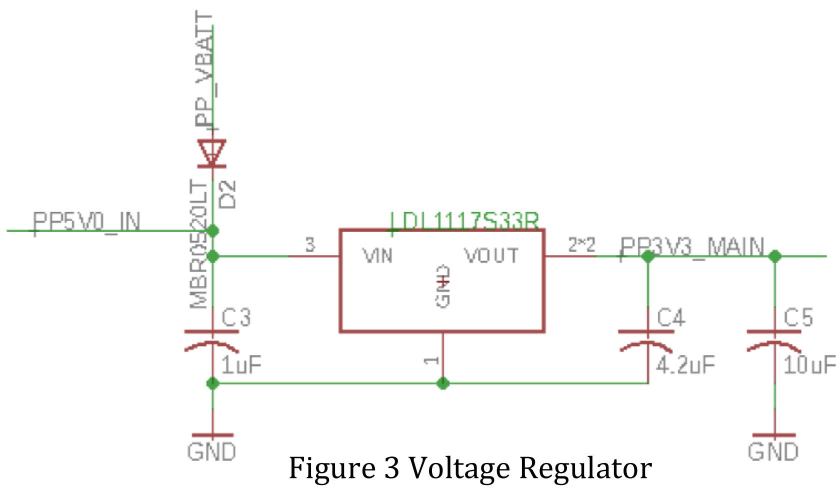

Figure 3 Voltage Regulator

Charging Circuit: The charging circuit was created to make the battery charge at 100mA. We placed a status LED to notify us when the battery was charging. The charging current was set by R2 as shown in Figure _ and we wanted to be able to charge the battery in around 30mins. When we actually tested the charging circuit it was charging the battery at around 40mins. This turned out to be incorrect because when we laid out PCB we switched the silkscreen labels for R1 R1 and R2 so the circuit was actually utilizing a 1KΩ. When the 10KΩ resistor was finally put into the correct place the battery charged in 4 hours.

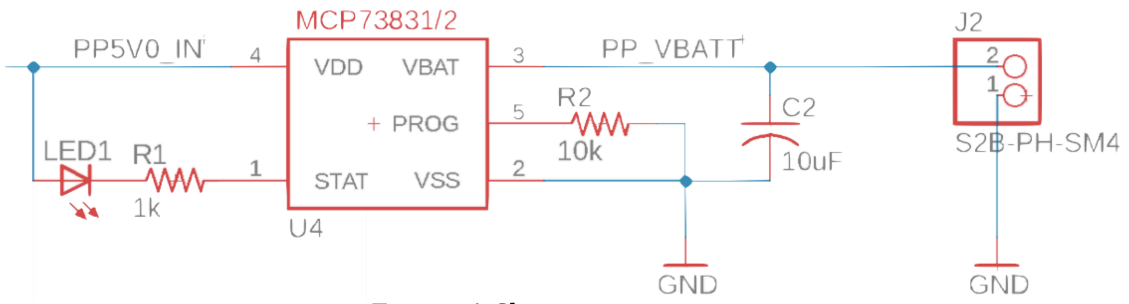

Figure 4 Charging circuit

Battery: We selected a 500mAh Lithium Ion battery. We wanted the battery to run between 3.5 and 4.2V and to be able to power our robot for a long amount of time. In practice we had to reverse the battery wires because ground and voltage were connected to the wrong pins for the battery connector. It required us re-soldering the wires connecting the batteries, which we then wrapped in electrical tape.

Motors: We selected DC motors for our robots instead of using stepper motors. Stepper motors are more accurate but the cost of using two stepper motors per robot would be too high. However, we should have selected motors that had a lower minimum speed, because with the motors we used our robots started and stopped abruptly. The quick starting and stopping made the motion of the robots

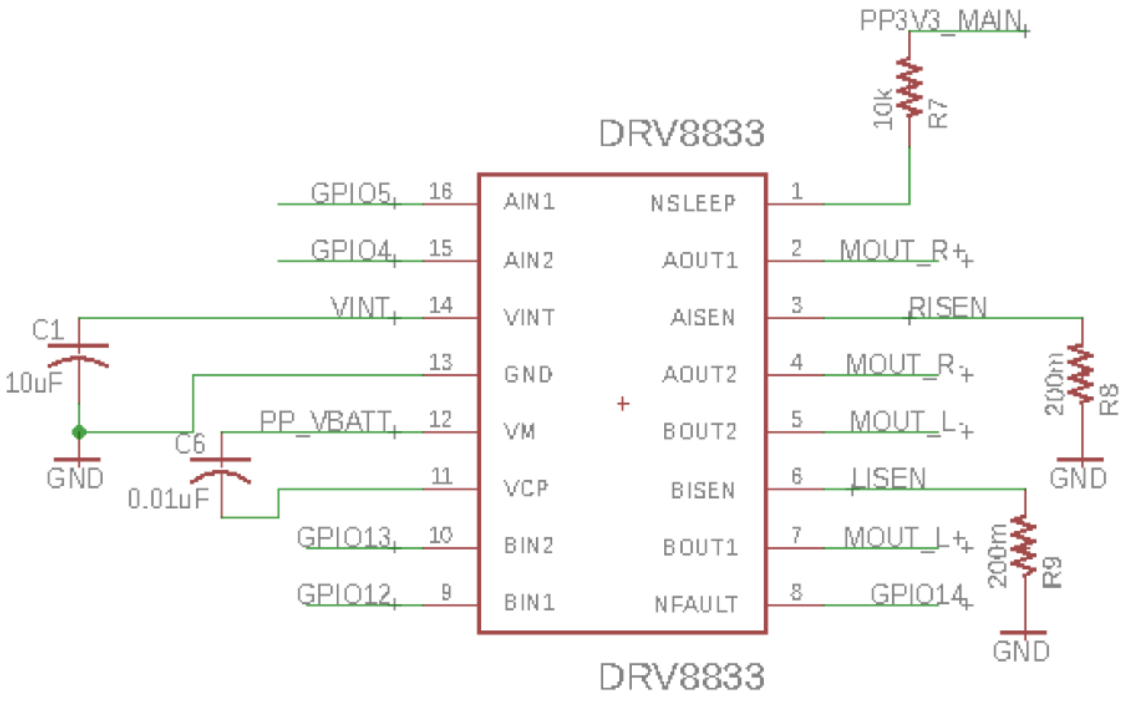

Figure 5 Motor Controller IC

Motors: We selected DC motors for our robots instead of using stepper motors. Stepper motors are more accurate but the cost of using two stepper motors per robot would be too high. We should have selected motors with smaller increments of motion, because with the motors we used our robots started and stopped abruptly. The quick starting and stopping made the motion of the robots seem jittery and was avoidable with better motor selection

Software Design:

The overall goal of the software was to have a control loop where: the computer takes a photo of the field with the web cam, the computer does manipulations with the image to find the position and orientation of the robots, the computer calculates the motor commands to get the robots to their specified endpoints, and finally send those commands to the robots. The first step of the vision software was to create a black and white image of the field with no distortions. An example of what a normal picture from the camera looked like is in Figure 6. In order to get the image into an easier to work with state, we used OpenCV to a perspective transform into a square image with no distortion. Also, we used OpenCV to change the exposure and contrast so that image targets appear more vivid. The next step we used was to use OpenCV to convert the image to black and white so that the robots would be one solid color as in Figure 7 (Left). From this black and white picture, we used OpenCV's ORB function to find all the corners and their orientations in the image as seen in Figure 7 (Right).

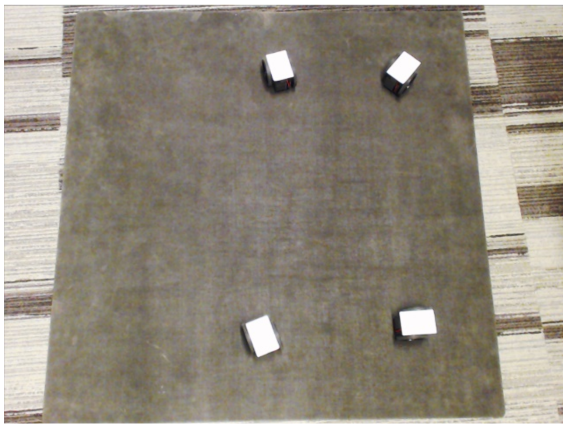

Figure 6 – Unaltered picture from the web cam

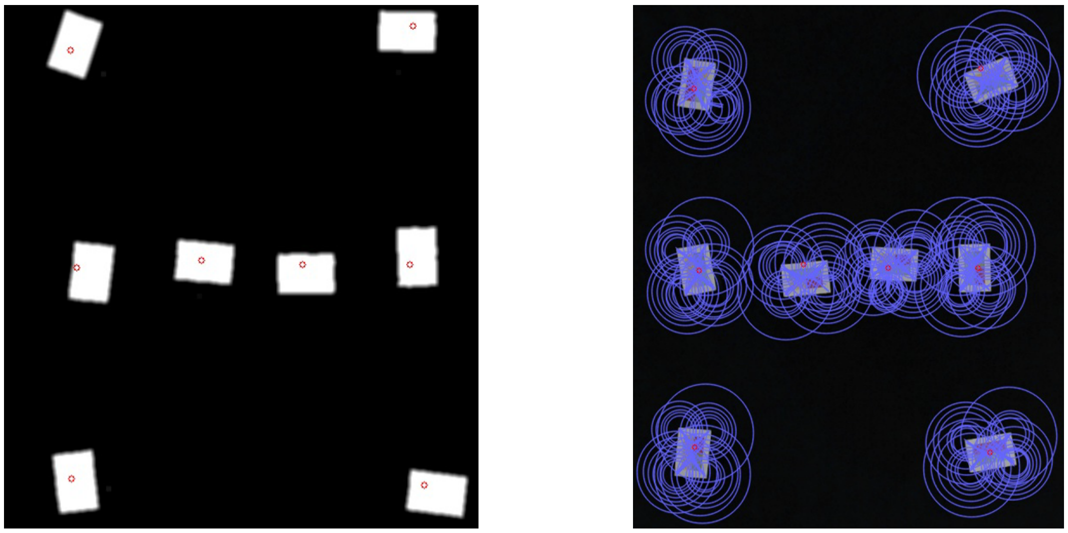

Figure 7 Image converted to black and white, and the found

After receiving finding all the corners, we try to cull all the duplicated corners so that the future calculations have less data to handle (Figure 8 Left). With the found corners, we try to guess where the other corners of the robot are. For every corner there are two possible orientations that could be guessed for each corner. Since the corners generated by ORB have at most 6 degrees of error. Given the equation in Equation EQA, we had at most 5.75 pixels of error due to the angle

$$\text{EQA: } Error = Width * \sin(6°)$$

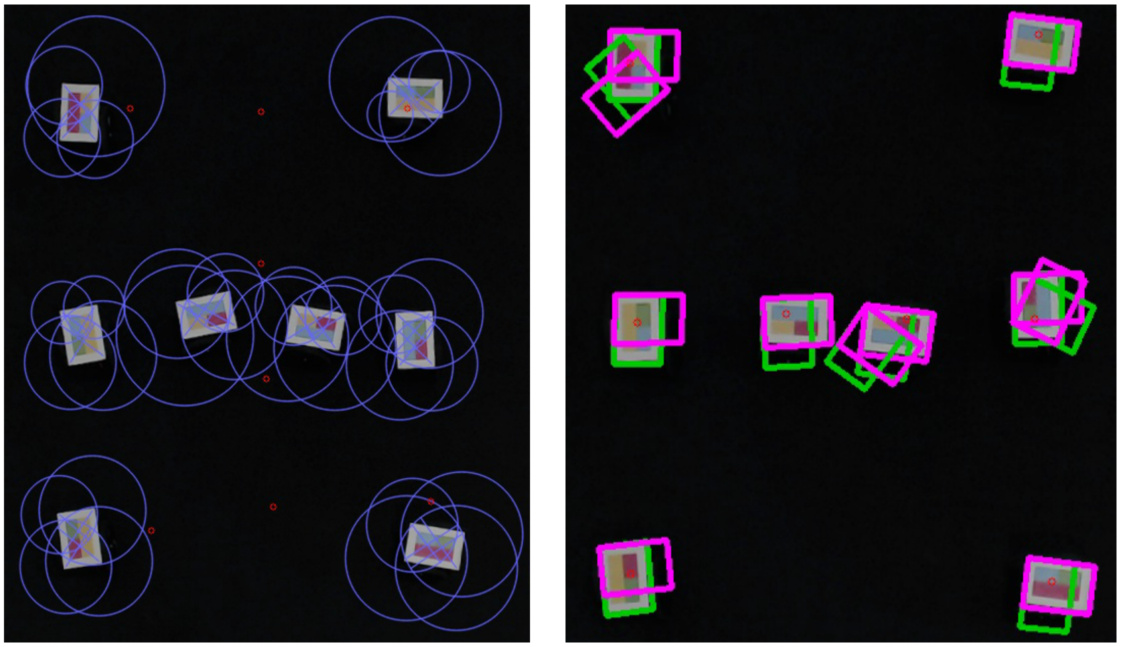
Figure 8 - Merged corners found orb on left, guessed locations

With the rectangles that we found, we look at the colored image to find the colors of the vision target. Once we have the four colors, we calculate the orientation such that top left corner of the double length bar of color is point of reference for our calculations. With the colors that we found, we have all the necessary information to know the location, rotation, and identity of the robot. The tag we use to identify the robots is an abbreviated color code that follows the colored boxes around clockwise. For example, the robot in Figure 9 (right) has the colors Cyan, Cyan, Orange, Magenta, so it would be labeled CCOM by the software. An example of correctly identified robots is in Figure 10.

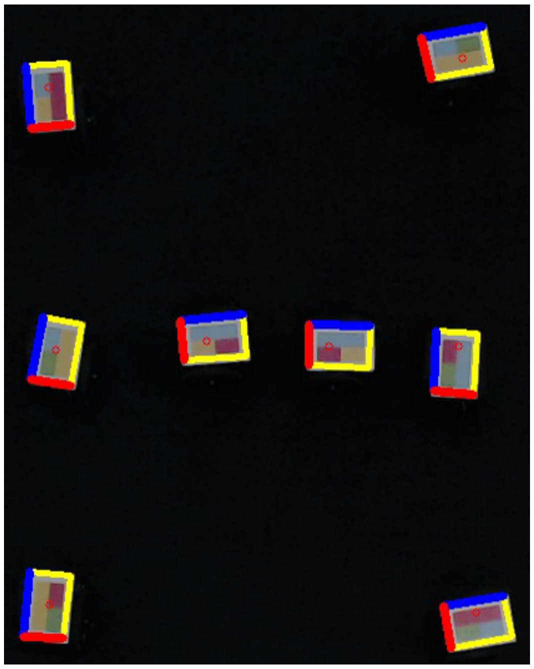
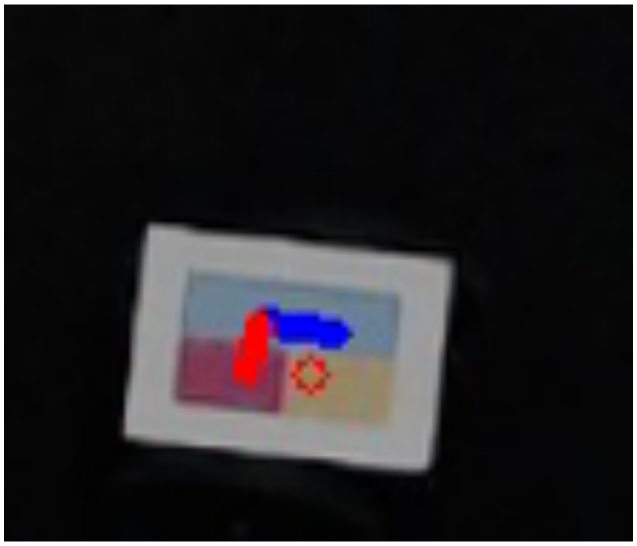
Figure 9: Rectangles that were found (Left)
Sampling colors from the vision target (Right).

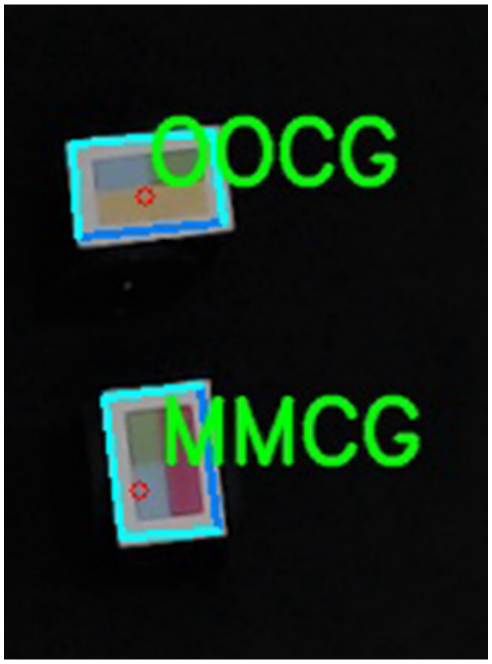

Figure 10 - Examples of correctly identified robots

Once we have an identified robot, we can calculate the distance between robot and endpoint and calculate angle necessary to travel to its endpoint. We send pure rotation commands until pointing within 30 degrees of the endpoint. With in 30 degrees, we send a forward command with a slight rotational component to help it reach the end point. Originally, the motors at minimum power moved too quickly to accurately point towards and arrive at the set point. In order to compensate for this effect, we had to add a cool-down that the robot had to wait for before running new commands
To send the commands to the robots, we sent UDP packets over 802.11n. The packets included a version and robot id, to ensure that communication was happening appropriately. In addition, the packets from the robot included other

Packet to Robot from Computer			
"C"	"T"	"R"	"L"
Version	Cooldown	Rotational Command	
Straight command		Time to run	
Flags		Robot ID 0	Robot ID 1
Robot ID 2	0xFF	Reserved	

Table 1

Packet to Computer from Robot			
"S"	"T"	"A"	"T"
Version	Cooldown	Reserved	
Total Sent Packets		Time to run	
Total Received Packets		Robot ID 0	Robot ID 1
Robot ID 2	0xFF	Reserved	

Table 2

The firmware the robot ran was a fairly simple C program. It had a control loop that ran every 10 milliseconds, and there also was a state machine (Figure 11) that mainly handled connection information. The first state was connecting to router, which was a secure Wi-Fi access point. The next state was a disabled state where the robot would wait for a command from the computer. After receiving a packet, it set the motors to run and then either go to a cool down state where the motors were disabled for a certain amount of time, or wait for a new packet while

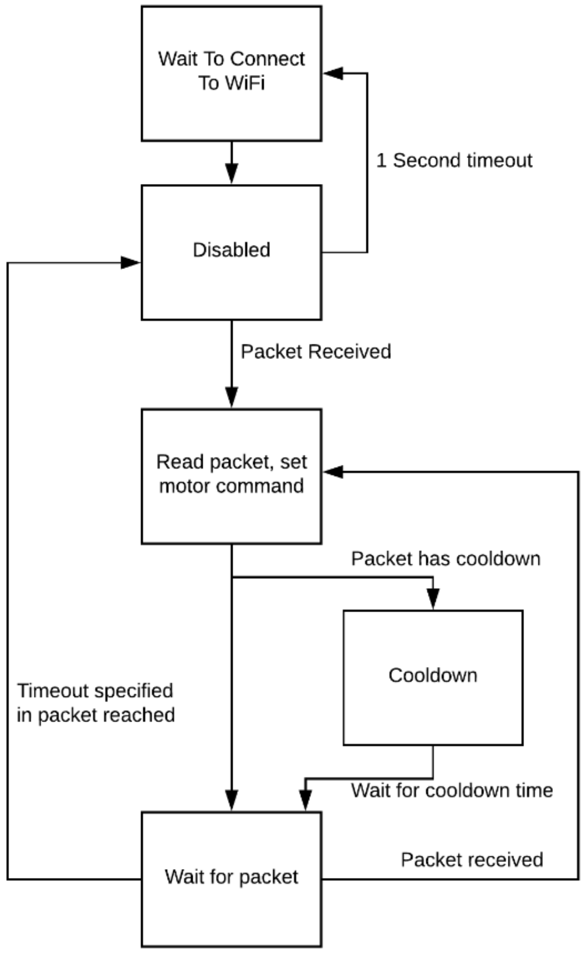

Figure 11: State machine of the robot.

Physical Design: The chassis was designed in Inventor and printed in the innovation study. The bottom and middle hold the motors in, and the middle and top hold the PCB in place.

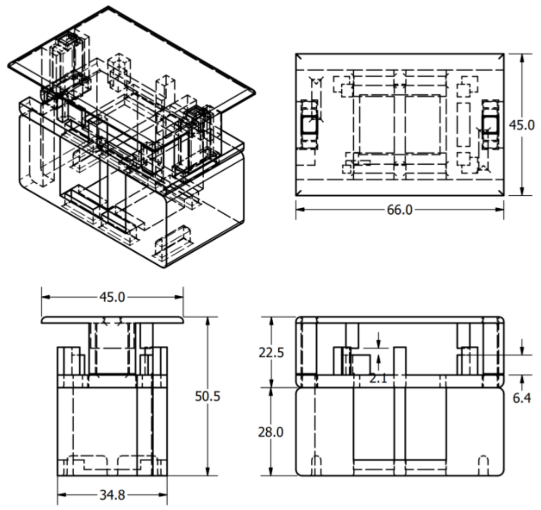

Figure 12: Physical Diagram

3. Cost & Schedule:

3.1 Parts:

Refdes:	Part:		Supplier:	Quanity	Price
U1	ESP12-S	SoC	Adafruit	20	125.20
U2	LDL1117S33R	LDO Regulator	Digikey	20	6.58
U3	DRV8833	Motor Controller	Digikey	20	43.20
U4	MCP73831	IC Charger	Digikey	20	11.80
D2	MBR0520LT	Shotkey Diode	Digikey	20	6.00
LED1-2	LH R974-LP-1	Red LED	Digikey	40	7.08
R1, 12	CR0805-JW-102ELF	Resistor, 1k	Digikey	40	0.76
R3-7, 10-11	CR0805-JW-103ELF	Resistor, 10k	Digikey	140	1.11
R8-9	CSR0805FKR200	Resistor, 200m	Digikey	40	9.69
C1-2,5	JMK212BJ106KD-T	Capacitor, 10uF	Digikey	60	6.07
C3	GRM21BR71C105KA01L	Capacitor, 1uF	Digikey	20	1.90
C4	LMK212BJ475KD-T	Capacitor, 4.7uF	Digikey	20	3.20
C6	GRM216R71H103KA01D	Capacitor, .01uF	Digikey	20	1.70
J2	S2B-PH-SM4-TB(LF)(SN)	Battery Conn	Digikey	20	10.94
PCB	---	PCB	PCBWay	20	5.00
Chassis	---	Chassis	UIUC	16	13.44
Battery	503035-500	Battery, 500mAh	Adafruit	20	143.20
Motor	---	DC Motor	Adafruit	40	126.00
Wheels	FS90R	Wheels	Adafruit	40	90.00
Total:					**612.87**

3.2 Schedule and Labor:

Week of:	General Task:	Mike:	Paul:	Peter:
2/26	Prototype motion	Put together motor testbench	Work on vision software, complete 16 target classification	Put together motor testbench
3/5	Complete PCB Design	Work on schematics and layout	Complete 16 target classification on real image	Work on schematics and layout
3/12	Complete RF tests, buffer week	Work on schematics and layout	Complete position computations for vision targets	Work on schematics and layout
3/19	Spring Break	None	None	None
3/26	Buffer Week	Lead time for ordering PCB	Complete color calibration	Lead time for ordering PCB
4/2	Buffer Week	Lead time for ordering Parts	Test vision software accuracy	Lead time for ordering parts
4/9	Initial full test	Assemble Initial PCB	Make software more robust	Assemble Initial PCB
4/16	Demo presentation preperation	Assemble remaining PCB's, failure analysis	Prepare for demo, emphasis on software	Assemble remaining PCB's, failure analysis
4/23	Demo	Demo	Demo	Demo

We calculated the following hours and salaries to come up with the labor cost:

	Hours:	Salary:	Cost
Mike:	95	$40	$3800
Paul:	105	$42	$4410
Peter:	90	$40	$3600
Totals:	290	$122	$11810

4. Requirements & Verification:

Motor Block: We wanted the motors to each draw less than 150mA with no load at 4V. In reality the motors drew 190mA but since the power profile of the SoC was 100mA under what we expected so this was a non-issue. We wanted the Motor Controller to operate from 3.7 to 4.2V, which turned out to be the case with some mild speed changes. We also needed the Motor Controller to be able to run a DC motor bidirectional, which succeeded.

Power Block: We wanted to be able to charge the battery in less than an hour. When we had our R1 and R2 values swapped on accident we charged in 40 minutes. We also wanted the Voltage Regulator to maintain a steady output between 2.95 and 3.4 Volts and we ended up having a range of 3.27 to 3.29V in our actual circuit. We also wanted a steady draw of 500mA from the battery, which occurred in the actual circuit. The battery was able to last 30mins for multiple robots and the output from the battery stayed between 3.7 and 4.18V.

SoC Block: We wanted the SoC to be fast enough to respond to compute and respond to loop corrections in under 100ms. We were able to connect to a 2.4GHz signal from the router. The SoC was able to transmit two PWM signals at the same time and drive two motors independently. The SoC ran reliable between 2.95 and 3.4V and there was no measurable change in packet loss at 15ft. The SoC ended up only drawing 90mA when sending and receiving packets at 100Hz, which was less than expected.

Vision system block: We wanted the camera taking video to be capable of producing images that were 1920 pixels by 1080 pixels and also be capable of sending 10 frames per second back to the computer. Our camera was able to stream at that resolution, but at 30 frames per second. Later in design, we realized that we the resolution to be lower so that we could compute the locations of the robot fast enough, so we streamed at 900 pixels by 720 pixels. The laptop needed to be able to analyze the video at greater than 10 frames per second and with changing the resolution to 900 pixels by 720 pixels, we were able to process 15 frames per second. The laptop and the router we used had to be able to transmit all the packets with 2.4 GHz Wi-Fi at a high enough success rate, that command timeouts would not be reached. With a packet failure rate of less than 3%, we were able to send packets reliably enough.

5. Conclusions:

Executive Summary: Our design provided a low cost solution to a complicated problem of design swarm interfaces. Our innovated use of machine vision for position detection allowed for reduction in complexity of the hardware, as well as increasing flexibility of solution deployment.

Safety and Ethics: There are multiple safety considerations for our swarm robotics. The most important of which, is the lithium ion battery in each robot. Lithium Ion batteries have been known to, and can, fail. Their failure can cause injury in the event of an explosion when the batteries cell and discharge rate create a positive feedback loop leading to overheating.

To prevent this situation from occurring we plan to design a charge protection element. The protection element separates the battery from the circuit charging elements. It monitors the temperature of the power source insuring that there is no runaway heating. The other precaution is to prefabricated batteries from a trustable source. Each robots charging circuitry will be checked to insure that the proper voltage is drawn from each battery. We will not make any modifications to the prefabricated batteries to uphold our responsibilities to section 9 of the IEEE code of ethics[4] – to prevent injuries to others. Our design will comply with the International Technical Commission standards for ion battery safety specifically 62133 for portable electronics.

A second concern is that in our project there is a web cam. Web cams are easily tampered with and provide the possibility for someone unwanted to spy on your property or person. We shall implement a method to automatically turn off the web cam when robots are no longer detected. As in section 1 of the IEEE code of ethics our design should concern the welfare of the safety of the public, specifically the users of our design. Privacy is a right that should be enjoyed. Warnings will be provided for situations in which our electronics should not be used. For damaged components of our devices such as those caused by an element falling off the desk, we suggest discontinuing the specific robots use. The same is true of any parts of our project damaged by water. Other guidelines for proper usage will be established so that our project is enjoyed safely and ethically. Our robots are not designed at a size where children could consume them and provide a choking hazard. All parts of our project are not large enough to cause a threatening physical force to humans.

To properly adhere to minimizing conflicts of interest in our project as per Section 2 and 3 of the IEEE code of ethics, we will be specifically careful in our citations as to avoid plagiarism. Our project was inspired by the zooids project done at Stanford. All parts of our project will make sure to provide a citation when they are derived or closely related to their work. Special care will be made to clearly state what is our own design and work.

6. References:

[1] Mathieu Le Goc, Lawrence Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, et al.. Zooids: Building Blocks for Swarm User Interfaces. Proceedings of the Symposium on User Interface Software and Technology (UIST), Oct 2016, New York, NY, United States. pp.97 - 109, 2016, .

[2] Ada, L. (2016, March 19). Adafruit_product_schem.png [PNG]. New York City: Adafruit Industries. ESP8266 Feather Schematic

[3] OpenCV: ORB (Oriented FAST and Rotated BRIEF). (n.d.). Retrieved February 23, 2018, from https://docs.opencv.org/3.4.0/d1/d89/tutorial_py_orb.html

[4] Institute of Electrical and Electronics Engineers, Inc.. (2006). Code of Ethics IEEE, http://www.ieee.org/. Retrieved at July 28, 2009