Roshinie Persaud

Oleksandr Litus

Gemma Bruce

November 1, 2018

GE 1501 Cornerstone

Dr. Susan Freeman

**Section 1:**

**Introduction:**

We had two processes in terms of strategizing with our robot. One process was visual wiring of the robot and the second was coding orientation of the robot and both were centralized around immense organization.

**Strategy:**

Our strategies were organized into multiple functions that were considered modes. We created variables that allowed our robot to turn left, right, forward, reverse, and turn at certain angles for certain periods of time through multiple types of move functions.

Our strategy consists of 5 modes, attack mode, detecting mode, defense mode, border mode, and trick mode. The robot enters the ring and is initiated through an on and off button. Once on we can check through our calibrated bool function if we place it true to run to see if all of the sensors are working properly. Once that is completed we decided to place our robot into fight mode pressing the button and uploading the code once again leading to the rest of our modes in a strategic order. We decided to create a bool detecting mode in which through our variables, once this is placed true the robot turns 360 degrees left or right -because we placed a random function- until it sees and opponent less than or equal to 17 inches and then it proceeds into attack mode. Attack mode consists of of the strategy that once the opponent is spotted through the detecting mode, it moves forward towards the opponent until the opponent is pushed out of the ring or when both of our robots see white. We also have a bool Border mode which always checks and ensures that our robot avoids the white lines. It states that through the border mode if the left sensor detects white it, the robot will immediately reverse backwards for one second, turn right at 90 degrees, and then move forward for one second and the proceed back to detecting mode and go through the process of the detecting the robot and then attack mode. If the robot’s right sensor detects white, the robot will immediately reverse backwards into the ring for one second, turn left at 90 degrees, and then move forward for one second and the proceed back to detecting mode and go through the process of the detecting the robot and then attack mode. If both sensors see white and the opponent is close to the sensor the robot will proceed to go into defense mode. Unfortunately defense mode was a function we were unable to accomplish and will strive to complete for the next competition.

Our overall and final approach towards this robot is to create a simple main function with multiple functions consisting of bools that displayed our tactics that we could organize and display in our main loop.

Our final approach was not to be only defensive or only offensive but to be both offensive and defensive and above all, to be wary of the white lines and stay within its parameters. Our offensive and defensive modes are up to our expectations, however we hope to install more modes such as trick mode and defensive mode, which we did not get time to program this time around. Defensive mode was hope to achieve if both lines see black, white, and then white while the opponent is pushing on the distance sensor, this means that the robot is being pushed out of the ring and our robot will begin to defend itself by pushing back. It was hoped to move specifically 10 degrees left then backwards, ten degrees left, then backwards, until it overwhelmed and dodged the forced of it’s opponent and goes around it and back into the ring. We also hoped to create a trick mode in which if our robot is on the opposite side of the ring and detects our opponent, it will instantly move towards the other side of the ring and attack the opponent from its side. These two of the many more goals we hope to strive to accomplish in our next competition, however now we are sufficed with the strategies we have installed in our robot.

**Our Process:**

We initially created a simple move function that consisted as follows: Move (1,1) meant moving forward, Move (-1,-1,) meant the robot would reverse Move (1,-1) meant the robot would turn left, Move (-1,1) would turn right, and Move (0,0) would make the robot stop. We then proceeded to incorporate time as well into this function in which it would declare how long the robot should make this orientation. For instance (1,1,1) declared that the robot would move forward for 1 second. We then wanted to create a smarter move function that incorporated angles and created an angle function in which we took the angles we desired and divided it by 360 and multiplied it by the amount of time it the robot to rotate 360 which is 2.7 seconds. This created and produced the time it would take for the robot to move the angle we so desired depending on the left or right orientation it takes. This would be valuable to our attack and defense functions. We then established if our sensors saw white it would do it’s best to avoid white as described in the strategy section. This idea remained consistent throughout our process. However in the beginning we decided that we were going to plug all of our functions into our main function, however this would create a very complex main function consisting of many if statements. Functions we initially created included random motions of side to side to confuse the opponent, attack function, turn around etc. However, we realized that these functions needed to be used more than once and that it would be quite complicated and bulky in the main function. Instead as Oleks puts it we decided to use the main function as a controller and made each incorporated voids and booleans into functions in which the switch would make functions easily true or false in the main function so it would be easier to call and rescind. These “Modes” are described in the strategy section as well. Our final process included improvement in naming variables and functions. For instance, our single Move function became multiple voids in which. We created two sections, movement for by degrees and movement by seconds. How movement works left and right are still the same as described in section 2 comments of the code. Movement seconds establishes direction and the amount of time we desire while movement by degrees establishes the turn of the robot at a certain angle by equalling the seconds to the amount of time it takes the robot to make that specific angle. Through these two main void functions we were able to make specific void functions in terms of time, direction, and angle such as “Void Stop Movement”, “Void turnRightForDegree”, “Void turnLefttForDegree”, “Void turnLeftForSec”, “Void turnLeftForSec”, “Void driveBackwardForSec”, and “Void driveForwardForSec”. We decided to create multiple descriptive functions to keep ourselves organized and available to make easy changes in terms of adding functions and troubleshooting. Through these complex movement functions we were able to create our modes that consist of voids and booleans to create tactics for our robot in the ring as explained in the strategy section. Our overall process of the development of ideas for our robot began with a very simple movement function that move backward, forward, left, and right and a main function that consisted of functions with various if statements that were hard to incorporate without multiple loops. We then slowly advanced to multiple descriptively named specific organized move functions, and strategies that are incorporated as modes with the use of booleans and voids in which the switch allowed us to call on strategy tactics easier. This development was trial and error and we still have much more to improve.

**Section 2:**

**Coding:**

**GitHub link for better experience (USE “MODES” BRANCH):** [**https://github.com/o1eks/Project\_KY.git**](https://github.com/o1eks/Project_KY.git)

/\* Title: Prototype 5

\* Date: October 30, 2018

\* Project\_KY is designed to control a sumo bot with variation of strategies in mind and ability to easily contribute with your own fight modes

\*/

// LIBRARIES--------------------------------------------------------------------

#include <Servo.h>

#include <time.h>

// SETTING UP SERVOS, SENSORS & BUTTON------------------------------------------

Servo leftServo;

Servo rightServo;

const int downLeftSensor = 12;

const int downRightSensor = 13;

const int trigPinFrontSensor = 8;

const int echoPinFrontSensor = 9;

const int onOfButtonPin = 2;

void setup() {

leftServo.attach (4);

rightServo.attach (7);

pinMode (downLeftSensor, INPUT);

pinMode (downRightSensor, INPUT);

pinMode (trigPinFrontSensor, OUTPUT);

pinMode (echoPinFrontSensor, INPUT);

pinMode(onOfButtonPin, INPUT);

Serial.begin (9600); // Setting serial output to 9600 speed

}

// DECLARING MODES SWITCHES-----------------------------------------------------

bool fightModeSwitch = false; // Setting the bool for Fight Mode to "false" at the start of the robot

bool calibrationModeSwitch = false; // Change to "true" in case need to calibrate sensors and servos

bool borderModeSwitch; // Switch that shows if Border Mode turned "on"

bool detectionModeSwitch = true; // Switch that shows if Detection Mode is turned "on", for the start of Fight Mode the robot goes straight to detection, so value is "true"

bool attackModeSwitch; // Switch that shows if Attack Mode turned "on"

//bool defenseModeSwitch; // Defense mode is not used in the code, was created for future improvments

// DECLARING MODES FUNCTIONS----------------------------------------------------

void fightModeFunction(); // Function for the Fight Mode, a controller with if statements to turn on and off other modes

void waitModeFunction(); // Function that is on, when Fight Mode is "false"

void borderModeFunction(); // Function for the case when robot in on the border line

void detectionModeFunction(); // Function for the process of detecting the opponent on the ring

void attackModeFunction(); // Function for attack of the opponent

// DECLARING GLOBAL CONSTANTS---------------------------------------------------

const double robotVelocity = 7.5; // Constant velocity of the robot measured with 4AA Energizer batteries (In/sec)

const double secondsFor360Turn = 2.7; // Constant amount of time it takes robot to turn 360 degrees in seconds measured with 4AA Energizer batteries

// DECLARING MAIN FUNCTIONS-----------------------------------------------------

void mainFunction();

// DECLARING MOVEMENT FUNCTIONS-------------------------------------------------

void basicMovementSec(int powerLeft, int powerRight, double moveSeconds);

void basicMovementForDegree(int powerLeft, int powerRight, double turnAngle);

void stopMovement();

void turnRightForSec(double moveSeconds);

void turnRightForDegree(double turnAngle);

void turnLeftForSec(double moveSeconds);

void turnLeftForDegree(double turnAngle);

void driveForwardForSec(double moveSeconds);

void driveBackwardForSec(double moveSeconds);

// DECLARING SENSORS FUNCTIONS--------------------------------------------------

bool leftSensorDetectWhite(); // Function for the left ground sensor (outputs 1 for white and 0 for black)

bool rightSensorDetectWhite(); // Function for the right ground sensor (outputs 1 for white and 0 for black)

float frontSensorDistanceInches (); // Function for the front sensor (outputs distance to the object in front of the robot)

// DECLARING ADDITIONAL FUNCTIONS-----------------------------------------------

void pressOnOffButton(); // Function detects the press of the OnOff Button and changes the value of the Fight Mode

double angleToSeconds (double turnAngle); // Function transforms desired turning angle to seconds for movement functions

void calibrationModeFunction(); // Function that stops the servos, and creates a Serial Outputs for sensors to calibrate

// STANDARD LOOP----------------------------------------------------------------

void loop(void) {

if (calibrationModeSwitch) { // If statement to go into Calibration Mode in case the Calibration Mode Switch is "true"

calibrationModeFunction();

} else { // Else go into Main Function

mainFunction();

}

}

// MAIN FUNCTION----------------------------------------------------------------

void mainFunction() {

pressOnOffButton(); // Calls for the function that controls On/Off Button

if (fightModeSwitch == true) { // If flight Mode Switch is "true" go into fight mode function

Serial.println("fightModeFunction");

fightModeFunction();

} else { // Else go into Wait Function

Serial.println("waitModeFunction");

waitModeFunction();

}

}

// MODES FUNCTIONS--------------------------------------------------------------

// Function for the Fight Mode, a controller with if statements to turn on and off other modes

void fightModeFunction(){

//delay(5000); // Wait 5 Seconds when fight mode is started (uncomment in case required by the competition)

// Conecting all sensor functions into local variables

double frontSensorReadingInches = frontSensorDistanceInches();

bool leftSensorOnWhite = leftSensorDetectWhite();

bool rightSensorOnWhite = rightSensorDetectWhite();

if (leftSensorOnWhite || rightSensorOnWhite){ // If left or right sensor detects white turn Border Mode on

borderModeSwitch = true;

}

// If statements that are calling for Mode Functions in case Mode Switch is "true"

if (borderModeSwitch){

borderModeFunction(leftSensorOnWhite, rightSensorOnWhite);

}

if (detectionModeSwitch){

detectionModeFunction(frontSensorReadingInches);

}

if (attackModeSwitch){

attackModeFunction(frontSensorReadingInches);

}

if (defenseMode){

defenseFunction();

}

}

// Function that is on, when Fight Mode is "false", was created to give ability in future add something while waiting, insted of just "Stop Function"

void waitModeFunction(){

stopMovement(); //do nothing

//Serial.println(digitalRead (downLeftSensor));

}

// Function for the Border Mode

void borderModeFunction(bool leftSensorOnWhite, bool rightSensorOnWhite){

if (leftSensorOnWhite){ // If Left Sensor detects white

driveBackwardForSec(0.5);

turnRightForDegree(90);

driveForwardForSec(1);

detectionModeSwitch = true; // Turn Detection Mode On

borderModeSwitch = false; // Turn Border Mode Off, because robot moved away from teh border

} else if (rightSensorOnWhite){ // If Right Sensor detects white

driveBackwardForSec(0.5);

turnLeftForDegree(90);

driveForwardForSec(1);

detectionModeSwitch = true; // Turn Detection Mode On

borderModeSwitch = false; // Turn Border Mode Off, because robot moved away from teh border

}

}

void detectionModeFunction(double frontSensorReadingInches){

Serial.println("detectionModeFunction");

if (random(1,0) == 1){ // Randomly decides in what direction to turn

if (frontSensorReadingInches >= 17){

turnLeftForSec(0); // Turns Left until front Sensor Detects an object within 17 inches

} else {

attackModeSwitch = true; // Turn Attack Mode On

detectionModeSwitch = false; // Turn Detection Mode Off, because the oponent was detected

}

} else {

if (frontSensorReadingInches >= 17){

turnRightForSec(0); // Turns Right until front Sensor Detects an object within 17 inches

} else {

attackModeSwitch = true; // Turn Attack Mode On

detectionModeSwitch = false; // Turn Detection Mode Off, because the oponent was detected

}

}

}

void attackModeFunction(double frontSensorReadingInches){

Serial.println("attackModeFunction");

if (frontSensorReadingInches <= 17){

driveForwardForSec(0); // Continue Attack (moving straight into opponent) till the front sensor distance is less than 17 inches

} else { // When front sensor distance is more than 17 inches (the oponent was lost)

detectionModeSwitch = true; // Turn Detection Mode On

attackModeSwitch = false; // Turn Attack Mode Off, as no opponent is seen

}

}

//void defenseFunction(){}

// MOVEMENT FUNCTIONS-----------------------------------------------------------

// Main two movement function that control servos based on the input

void basicMovementForSec(int powerLeft, int powerRight, double moveSeconds) {

int leftS, rightS; // Declaring local variables for left and right servos

// If input for first function variable is 1 - drive forward, if -1 - drive backward, else stop the servo

if (powerLeft == 1) {

left = 180;

} else if (powerLeft == -1) {

leftS = 0;

} else {

leftS = 90;

}

// If input for first function variable is 1 - drive forward, if -1 - drive backward, else stop the servo

if (powerRight == 1) {

right = 0;

} else if (powerRight == -1) {

rightS = 180;

} else {

rightS = 90;

}

// If input for time is 0 - drive forever, in nozero, use the number as the number of seconds of the delay

if (moveSeconds == 0) {

leftServo.write(leftS);

rightServo.write(rightS);

} else {

leftServo.write(leftS);

rightServo.write(rightS);

Serial.println("moveSeconds = ");

Serial.println(moveSeconds \* 1000);

delay(moveSeconds \* 1000);

}

}

void basicMovementForDegree(int powerLeft, int powerRight, double turnAngle) {

int leftS, rightS; // Declaring local variables for left and right servos

double moveSeconds; // Declaring local variable for the number of seconds to move

// If input for first function variable is 1 - drive forward, if -1 - drive backward, else stop the servo

if (powerLeft == 1) {

left = 180;

} else if (powerLeft == -1) {

leftS = 0;

} else {

leftS = 90;

}

// If input for first function variable is 1 - drive forward, if -1 - drive backward, else stop the servo

if (powerRight == 1) {

right = 0;

} else if (powerRight == -1) {

rightS = 180;

} else {

rightS = 90;

}

// Assigns moveSeconds a value from the function that transforms angle into the time needed for the turn

moveSeconds = angleToSeconds(turnAngle);

// If input for time is 0 - drive forever, in nozero, use the number as the number of seconds of the delay

if (moveSeconds == 0) {

leftServo.write(leftS);

rightServo.write(rightS);

} else {

leftServo.write(leftS);

rightServo.write(rightS);

Serial.println("moveSeconds = ");

Serial.println(moveSeconds \* 1000);

delay(moveSeconds \* 1000);

}

}

// Other functions are created to improve the readability of the code and easier syntax

void stopMovement(){

basicMovementForSec(0,0,0);

}

void turnRightForSec(double moveSeconds){

basicMovementForSec(1, -1, moveSeconds);

}

void turnRightForDegree(double turnAngle){

basicMovementForDegree(1, -1, turnAngle);

}

void turnLeftForSec(double moveSeconds){

basicMovementForSec(-1, 1, moveSeconds);

}

void turnLeftForDegree(double turnAngle){

basicMovementForDegree(-1, 1, turnAngle);

}

void driveForwardForSec(double moveSeconds){

basicMovementForSec(1, 1, moveSeconds);

}

void driveBackwardForSec(double moveSeconds){

basicMovementForSec(-1, -1, moveSeconds);

}

// SENSOR FUNCTIONS-------------------------------------------------------------

// Function for the left ground sensor (outputs 1 for white and 0 for black)

bool leftSensorDetectWhite() {

int leftSensorReading = digitalRead (downLeftSensor);

if (leftSensorReading == 1){

Serial.println ("Left Detects White");

return 1;

} else {

Serial.println ("Left Detects Black");

return 0;

}

}

// Function for the right ground sensor (outputs 1 for white and 0 for black)

bool rightSensorDetectWhite() {

int rightSensorReading = digitalRead (downRightSensor);

if (rightSensorReading == 1){

Serial.println ("Right Detects White");

return 1;

} else {

Serial.println ("Right Deytects Black");

return 0;

}

}

// Function for the front sensor (outputs distance to the object in front of the robot in inches)

float frontSensorDistanceInches () {

float echoTime;

float calculatedDistanceInches;

digitalWrite (trigPinFrontSensor, HIGH);

delayMicroseconds (10);

digitalWrite (trigPinFrontSensor, LOW);

echoTime = pulseIn(echoPinFrontSensor, HIGH);

calculatedDistanceInches = echoTime / 148.0;

return calculatedDistanceInches;

}

//ADDITIONAL FUNCTIONS----------------------------------------------------------

// Function check if On/Off Button pressed and changes Fight Mode status accordingly

void pressOnOffButton() {

//read the state of the "On/Off" button value:

int onOffButtonState = digitalRead(onOfButtonPin);

//in case "On/Off" button is pressed change to and from the "fight mode"

if (onOffButtonState == HIGH) {

if (fightModeSwitch == false) {

fightModeSwitch = true;

Serial.println("Fight Mode --> ");

Serial.print(fightModeSwitch);

} else {

fightModeSwitch = false;

Serial.println("Fight Mode --> ");

Serial.print(fightModeSwitch);

}

delay(200); // adds delay for user to remove the finger

}

}

// Function trans forms turning angle into time (s) based on the measured rotation speed

double angleToSeconds (double turnAngle) {

double turnTime;

turnTime = (turnAngle \* secondsFor360Turn) / (360);

Serial.print ("turnTime --> ");

Serial.print (turnTime);

return turnTime;

}

// Calibration funciton (for ground sensors, servos, and front sensor)

void calibrationModeFunction() {

stopMovement();

double frontSensorReading = frontSensorDistanceInches();

Serial.println ("Front Distance (in) --> ");

Serial.print (frontSensorReading);

delay (100);

int leftSensorReading = digitalRead (downLeftSensor);

int rightSensorReading = digitalRead (downRightSensor);

if (leftSensorReading || rightSensorReading){

Serial.println ("");

Serial.println ("-------------------------------------------------");

Serial.println ("");

if (leftSensorReading == 1){

Serial.println ("Left Sensor is Not Calibrated");

} else {

Serial.println ("Left Sensor is Calibrated");

}

if (rightSensorReading == 1){

Serial.println ("Right Sensor is Not Calibrated");

} else {

Serial.println ("Right Sensor is Calibrated");

}

delay (100);

}

else {

Serial.println ("Both Sensors are Calibrated");

}

}

//END---------------------------------------------------------------------------

**Section 3:**

**What did you accomplish as a team?**

As a team we accomplished the overall strategy of the robot and decided that we did not want to be only offensive or defensive but in fact both with a little bit of random motion included. Not only that but we decided and accomplished as a team the aesthetic visuals of our robot in terms of wiring organization and cleanliness (and braiding), as well as the name (Ky), and how many functions we would like to incorporate in our function as well as the decision to use all of the sensors and that the main goal no matter what is to keep the robot in the ring and avoid the white lines as much as possible. We accomplished as a team a visually pleasing robot that avoids the white lines when it is encountered, detects the opponent with a 360 observation, and once spotted attacks immediately with an attack function and moves forward push the object out until the robot is out of the ring and our robot sees white.

**How did the team successfully work together?**

The team successfully worked together in terms of collaboration of ideas and work ethic. We are very good at compromise. For instance, when it came to discussing what types of functions we should create in terms of strategies, we all divided up the types of functions we should create based on strategy and created pseudocode on how these strategies should be executed. We were successful in terms of understanding our task and understanding the process as to how to code the robot and set it up. For instance, Gemma was in charge of setting up the calibration of the sensors, while Roshinie was in charge of developing the pseudocode for the modes such as the detecting mode, attack mode, and border mode and wiring and construction of the robot, while Oleks worked on the development and organization of the code and the wiring and construction of the robot as well.

**What will your team improve on next time?**

I believe our team can work on our meeting times a little better and perhaps communication skills as to creating time parameters to work, but for the most part we managed to all work together collaboratively even if we didn’t meet each other due to the creation of a shared file on google drive in which we uploaded our code to everytime we worked on it.

**What did you learn from this phase of the project?**

We learned that time management was a major component when it came to designing this robot and that pseudocode is key when developing the different strategies for the robot and how we wish to execute these ideas, and above all that organization is key when it comes to developing this robot. Whether it’s how you declare your variables or wire your board a mistake as small as a semicolon can happen anywhere if you aren't tedious and you can spend up to an hour trying to find this mistake if you aren't organized. We also learned that the First Year Learning Center is a very useful resource and that sometimes when you can’t figure out something on your own that stubborness is not key and that it is ok to ask for help from wiser older students who are willing to help as long as you try.

**What do you call a success?**

We call a success a robot that is aesthetically pleasing to the eye and does not have a wire “man bun” and is able to enter the ring, calibrated and is able to stay in the ring and avoid the white lines when it is encountered and above all, push the opponent out of the ring and win! A success is meeting all of the requirements of competition one and completing a fair competition match while staying in the ring for 3 minutes.

**What might be a failure to learn from?**

A failure to learn from might be not accomplishing our defensive mode function and trick mode function we so desired to complete on time. But it’s ok because this will be our goals to create next time. The defense mode would have prevented the the robot to be pushed completed out of the ring after both sensors sense white and the distance sensor senses the opponent. This is because with our current border mode it only knows to avoid the white lines when either sensor senses white. However when an opponent pushes our robot consistent the our robot will only sense white for a second and the black color form the out ring and will not fight back. Therefore we needed this defense mode so that when the robot senses “black white then black again” on both sensors and the opponent is close, our robot will know immediately to continue to defend itself by reversing back into the ring.

**Section 4:**

**Individual Reports:**

**Roshinie Persaud:**

**What part was primarily yours?**

I contributed in creating pseudo code for our functions in terms of modes such as attack mode, border mode, trick mode, defense mode, and detecting mode which Oleks used to code onto Arduino. I also helped build the physical robot as well as wire the sensors and servos to the board. I also typed up and created the calibration code for all of the sensors.

**What part(s) did you contribute to?**

Specifically I helped contribute the specific pseudocode functions I described above as well as the ability for our robot to turn a specific angle through time through the equation seen in the code in section 2. Through this process my team and I timed 10 times how long it took for the robot to rotate 360 degrees and it came out to be 2.7 seconds. Even though I developed pseudocode, Oleks was the one who wrote the actual code based on the pseudocode for the Modes. I also helped build the robot by screwing the wheels on and taping the wheels so they would not fall of, and I replaced the old velcro. In terms of wiring, all team wired the robot using precision and organization so that our wires would not stick out of the robot and get damaged during the fight I helped wire the distance sensor and the motors and was very pleased with my braiding skills for the wires attached to the distance sensor… It adds character. I also typed up and created the calibration code for all of the sensors. I am also the the person in charge of typing up the reports as we discuss what to put into it.

**What strengths do you bring to the team?**

The strengths I bring to the team involves being organized and time management as in organizing when to work. I think my abilities to articulate strategies in words such as in reports and in pseudocode is my strength. I think my abilities to bring everyone together to work and divide up work and meet a deadline is my strength.

**What would you like to do better as a contributor next time, if any?**

Truthfully, it really frustrated me that I have a weakness in coding and memorizing syntax and the process as to when to apply them in the most logical way. I feel that I was a weak link when it came to coding, especially because Oleks is very advanced and I felt I slowed the team down with my constant questions and slow process to develop and understand the code and how to implement and create functions. I want to improve my coding skills next time.

**Oleksandr Litus:**

**What part was primarily yours?**

My primer task was rewriting pseudo code of robot logic into Arduino code and debugging itm optimizing, cleaning. Writing comments for the code was done by me in lots of occasions. And discussing the code options and opportunities with others was something I was leading in the team.

**What part(s) did you contribute to?**

Most of the logic was written in the team, so I added a lot there. Building of the robot and wiring was team work, where I did initial installation of the Red Board and other components, that were wired by the team. I contributed a lot to the code in the final report.

**What strengths do you bring to the team?**

I brought my knowledge of the programming and ability to cooperate. I am also highly motivated for the win, so I believe it helped our team to achieve success.

**What would you like to do better as a contributor next time, if any?**

I believe I could have given more opportunities for the code righting itself to my team members. Net time I would create a GitHub repository, so everyone can work on their part of the code.

**Gemma Bruce:**

**What part was primarily yours?**

I handled the calibration of our motors and helped plan strategy for the actual battle, like its offensive and defensive techniques.

**What parts did you contribute to?**

I helped calibrate the various sensors in the FYELIC and helped to test the robot. I also helped construct the robot and offered thoughts on logistical things.

**What strengths do you bring to the team?**

I am a good communicator and good at organizing tasks for the group members. I ask questions and help clarify things when I believe that something is unclear and might be unclear to other group members as well.

**What would you like to do better as a contributor next time, if any?**

Next time I would like to be able to help more with the coding of the robot so I could learn more. I would also like to work on the mechanical side of actually building the robot.