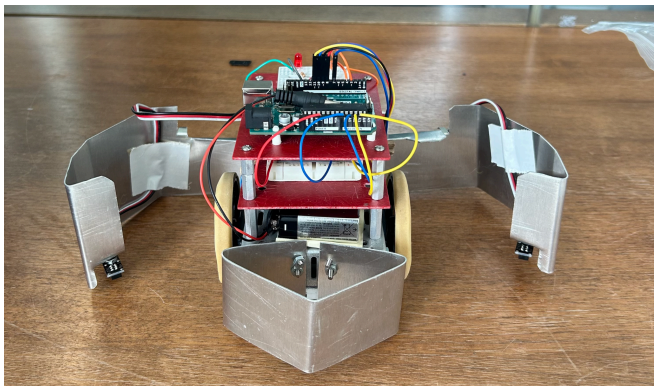
MAE 3780: Robot Project Final Report

Team 27: TEC
May 16th, 2024

Chimdi Anude - cca54w
Tyler Wisniewski - ttw24

ROBOT DESIGN AND STRATEGY OVERVIEW

Our mechanical robot design was a plow in front of the robot that would guide the cubes into the robot's arms. The arms were curved to keep the cubes in one place as we moved around the board. We used steel for the robot arms as it would make the arms less likely to bend out of shape. At the start of the competition, the idea is that the arms are raised above the robot to stay within the size constraint, and as the robot moves, the arms come down. Our electrical design used the color sensor, two QTI sensors, and an LED. We used the color sensor to detect whether the robot was on the board's yellow, blue, and black sides. The QTI sensors were to detect if the robot's arms were at the border. The LED signaled when the robot was collecting blocks and ready to return home. The software design was for 45 seconds; the robot goes to the other side and collects blocks. On the away side, the robot goes straight unless it detects a border from the QTI sensors. Then, it turns 90 degrees and 180 degrees if the color sensor detects black. Once the 45 seconds are up, the LED turns on, and the robot returns to its home side since it carries all the cubes in its arms.



DESIGN PROCESS REFLECTION

So, for Milestone One, we presented the flowchart in Appendix D and explained our strategy for the competition to the TAs. Milestone Two was simply assembling the robot with the parts made available in the lab and having it move in a particular path. One thing we noticed was that our robot drifted to the left. We experimented with different types of wheels to mitigate this, but it was a persistent issue that we worked around in our code. We were still on track with our original design in Milestone Two. That changed with Milestone Three. We struggled with Border detection. Initially, we used a pin change interrupt for the color sensor to get new readings from the sensor. From there, it would decide whether it was blue or yellow. However, we had difficulties getting values for the QTI sensors, which led to us getting checked off a day late. To pass Milestone Three, we

used the color sensor to detect all three colors and figure out the QTIs later. By Milestone Four, we figured out how to use the QTI sensors and assembled the plow and robot arms. We programmed it to take the path we proposed in Milestone One without the timing, and it worked very well. Our robot could carry at most fifteen cubes in its arms. The biggest problem we had leading up to the competition was timing the code for 45 seconds and having it return home. We tried to implement a timer and even a counter, but there were many bugs we could not solve in time for the robot. The code we used to compete in was untested, so we had no idea how it would perform in the competition.

COMPETITION ANALYSIS

During the competition portion of this project, our robot ran into some unexpected issues that resulted in its overall poor performance. Our robot competed in 7 matches, finishing with a record of 2 wins, one draw, and four losses. Our robot's performance was identified as a bug in the code. Due to unfortunate time circumstances, the code on our competition robot was untested. As such, our robot displayed very bizarre behavior whenever it started on the blue side. Due to the different lighting in the Duffield Atrium, the color sensor reads Blue as Black; thus, the robot is stuck in a retreating cycle anytime it finds itself in Blue.

Alternatively, our robot's mechanical design and material selection became a strength during the competition. Its metal frame allowed our robot to maintain its shape after unexpected contact/engagement with an opposing robot. In addition, the functionality of the wing, which was used to maintain its starting position, worked well throughout the competition. This allowed us to maximize the width of our robot, which was beneficial when securing blocks.

After the competition, a slightly edited version of the code was uploaded, and the robot did significantly better. TEC won several exhibition matches against strong robots, including tournament Quarter-Finalists and the ASML robot.

CONCLUSIONS

Our robot design journey was marked by significant learning experiences, technical challenges, and eventual triumphs. The robust mechanical design effectively controlled cubes and performed well in securing blocks and maintaining integrity during collisions. The robot performed so well in collisions that forcing collisions even became a viable strategy in later exhibition matches.

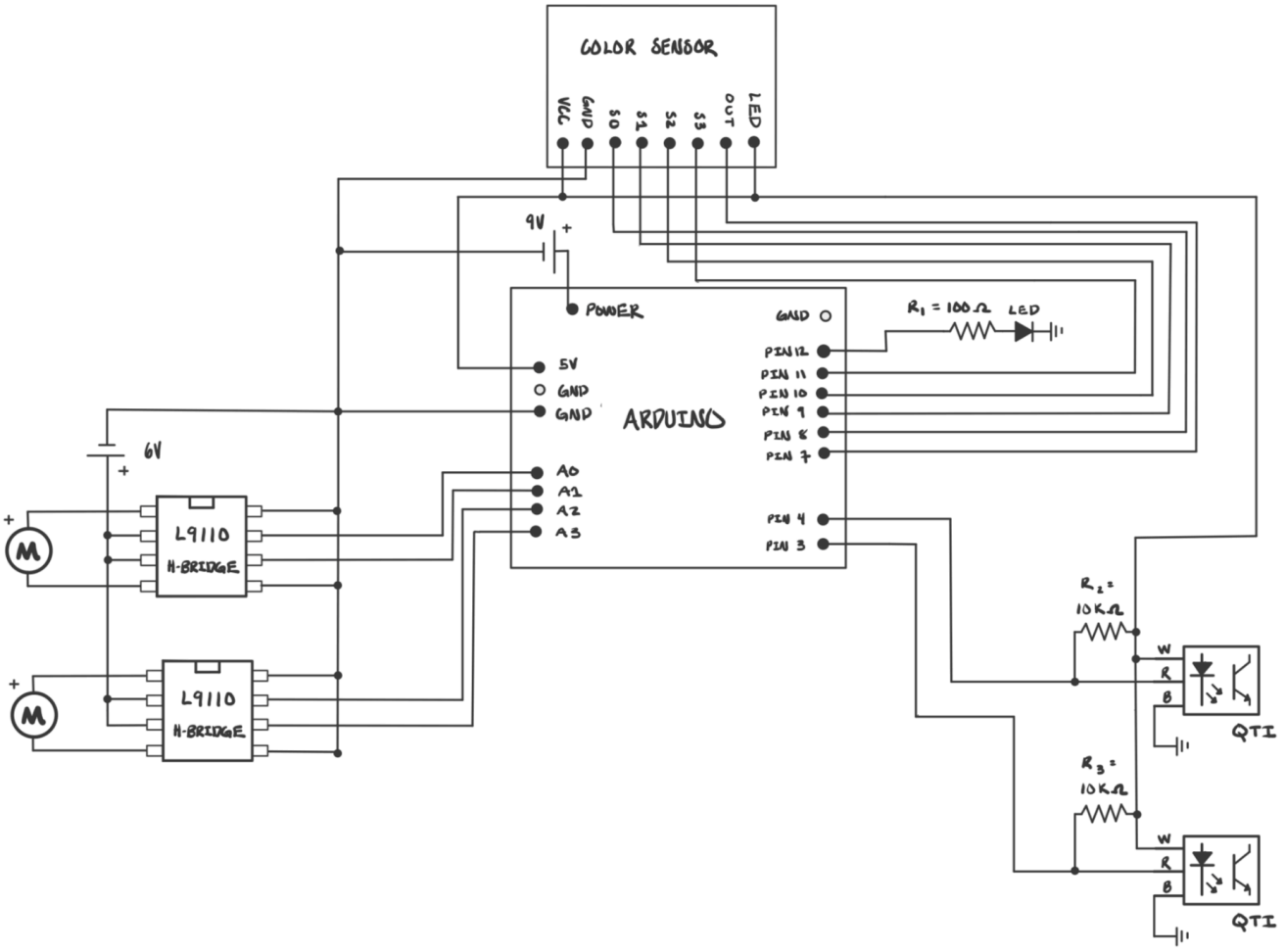
We should have made better use of open hours before the competition so that we could test our final code before the competition. This project highlighted the importance of thorough testing and adaptability, providing valuable insights for future robotics endeavors. We encourage future groups to precisely budget time for testing and tuning their robot so that your team can finalize a strategy and test code in advance of the competition.

Also, we recommend that future groups properly use their budgets. Our team was significantly under the allotted budget, and we would have bought cliff bars or Swedish fish on McMaster to fuel us throughout the competition.

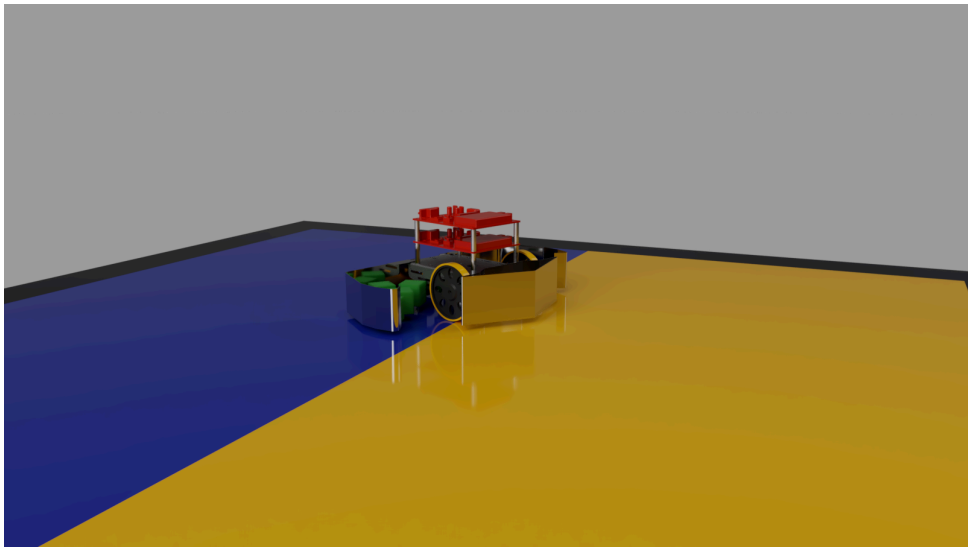
APPENDIX A: BILL OF MATERIALS

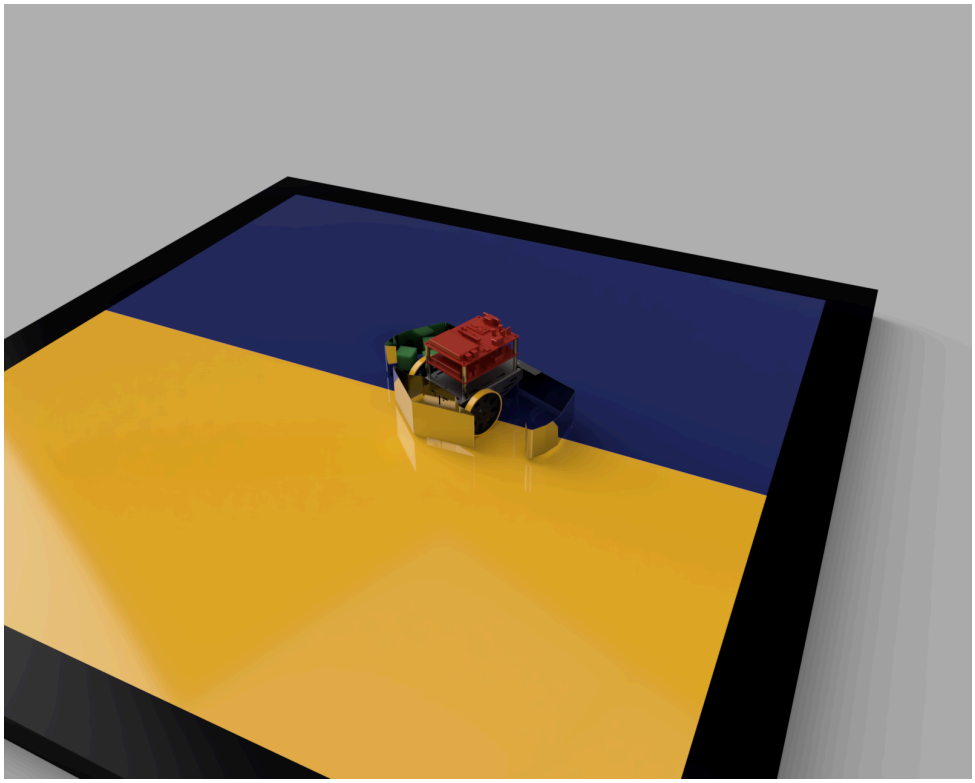
Vendor	Item Name	Unit Price	Quantity
Lab	Half Breadboard	$1	1
Emerson Machine Shop	12x12 Sheet Metal	$5	1
Lab	Pair of 70 mm Swervo Wheels	$6.45	1
		Total Cost	$15.45

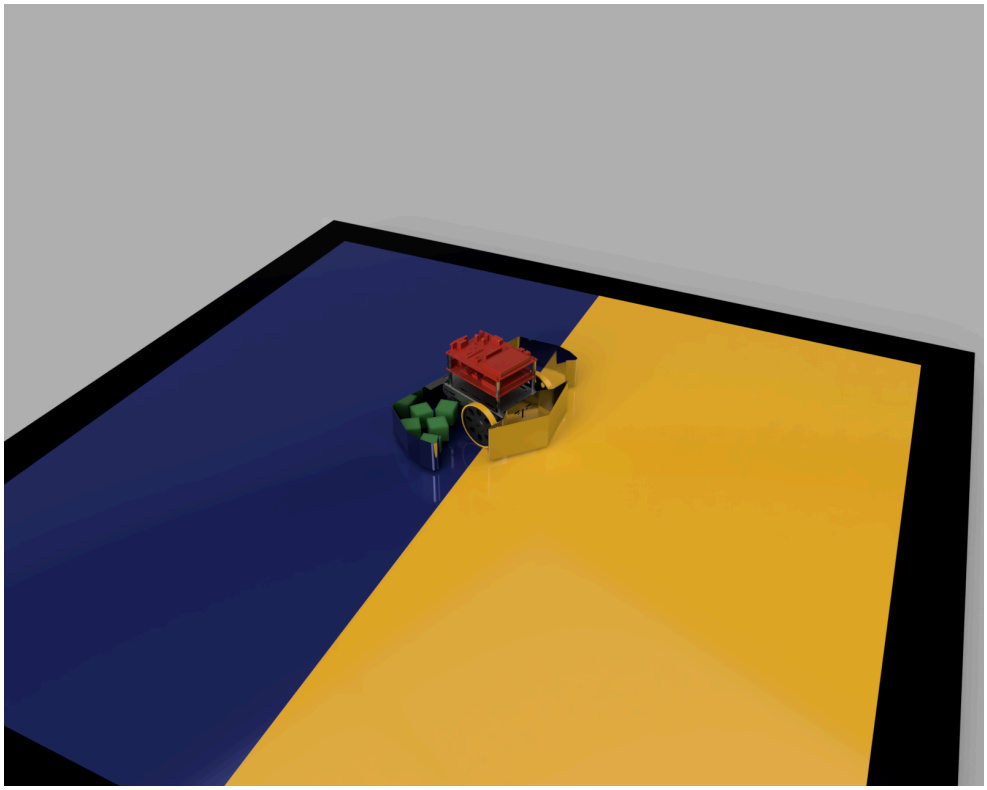
APPENDIX B: CIRCUIT DIAGRAM

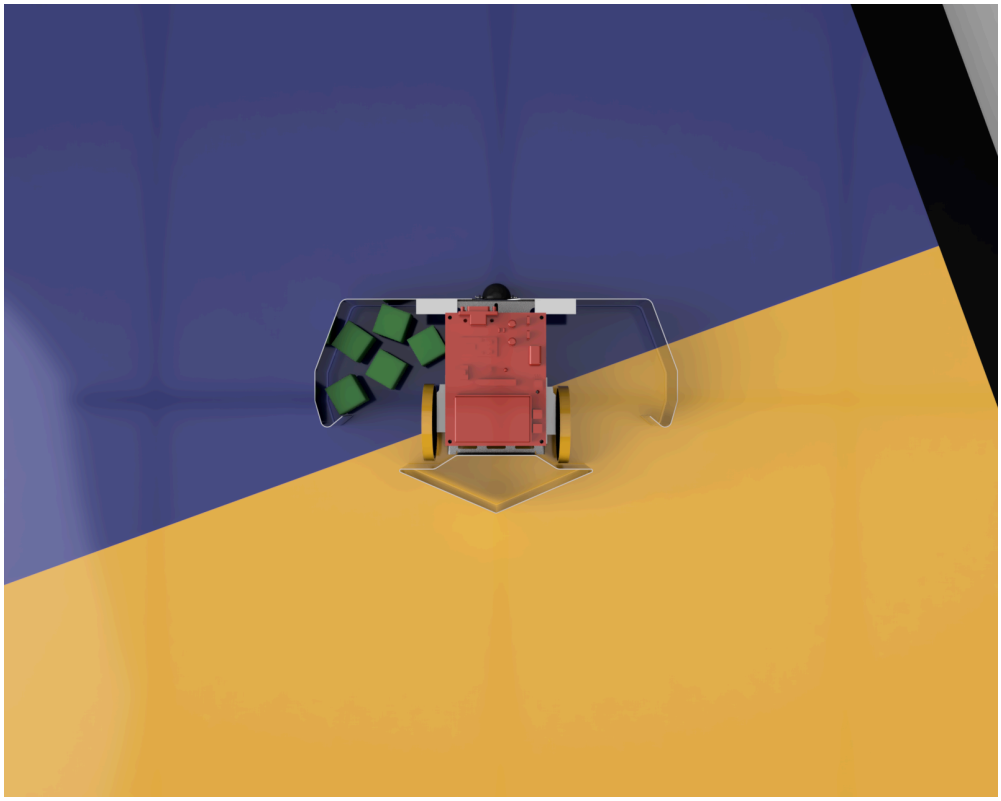


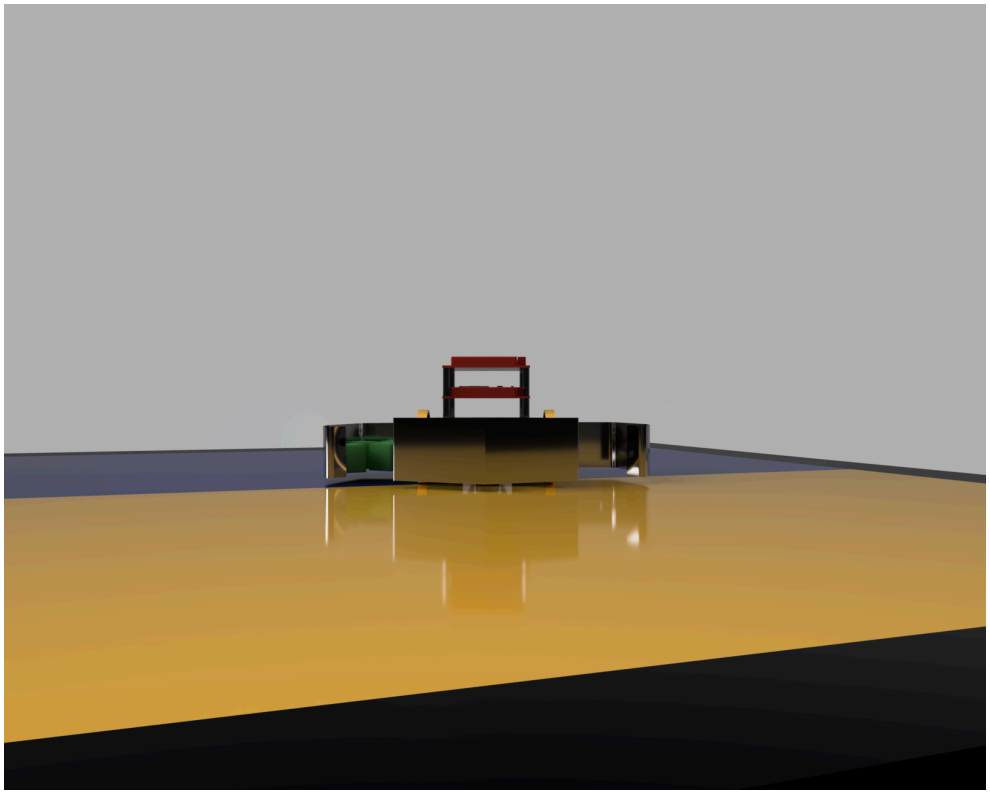
APPENDIX C: CAD FILES AND DRAWINGS

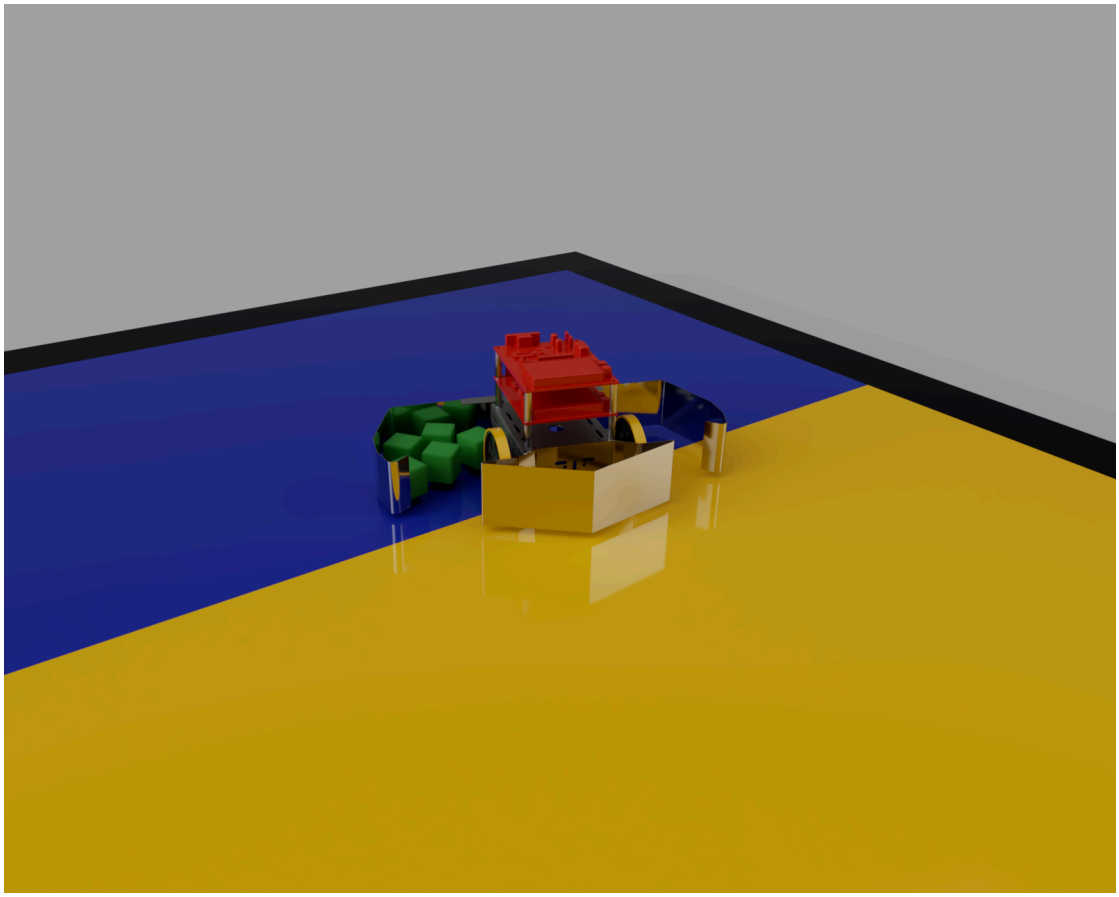


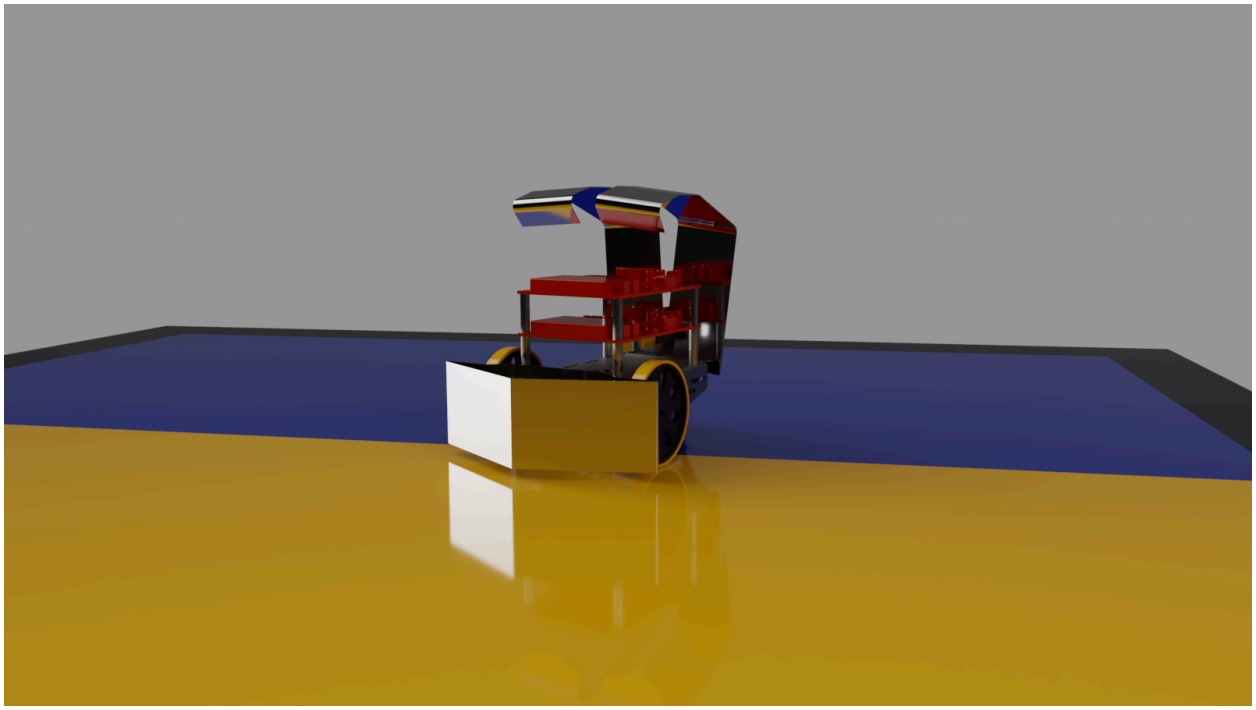


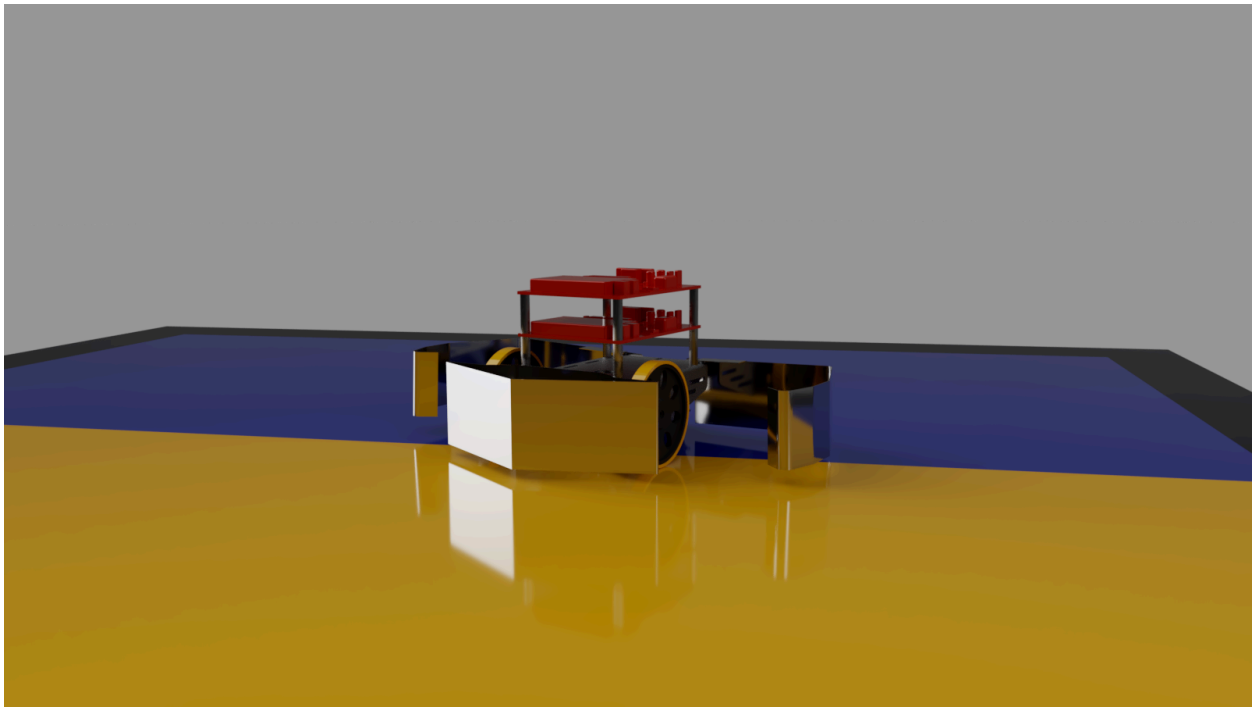




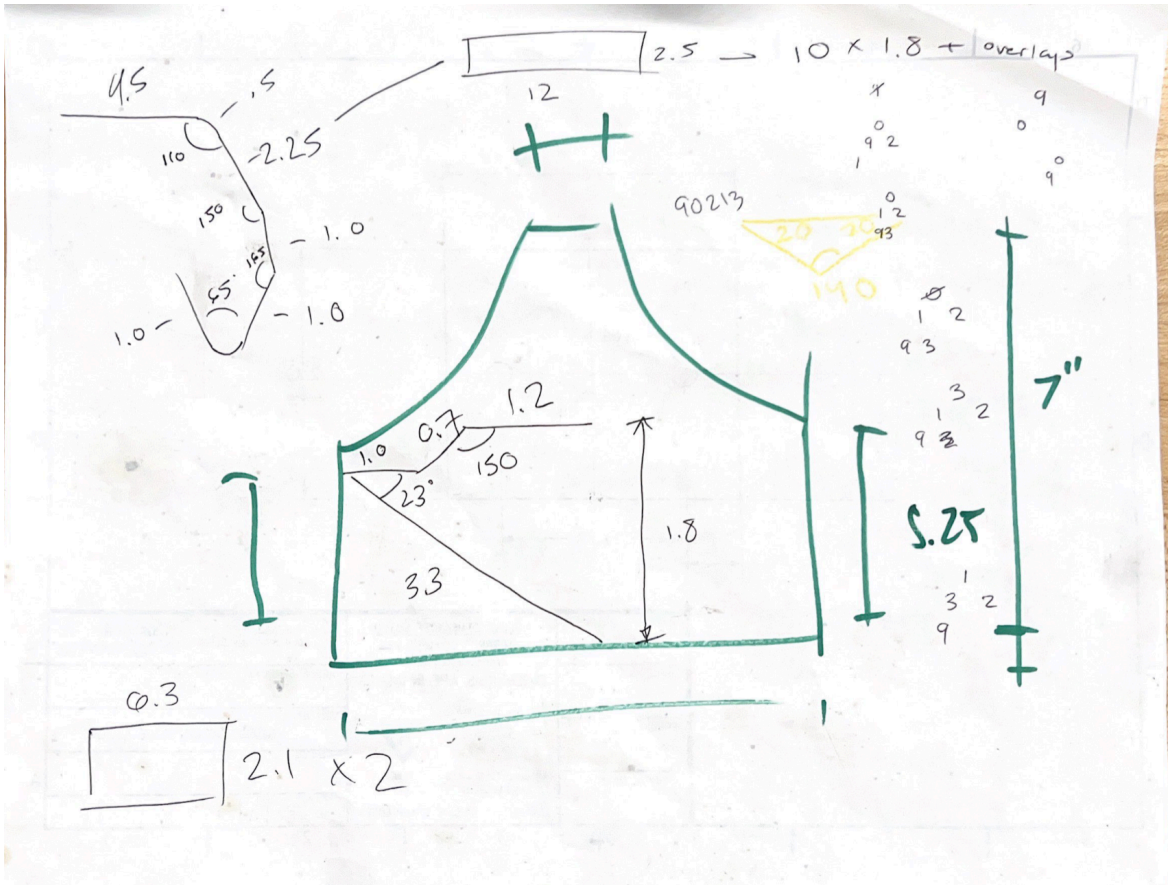




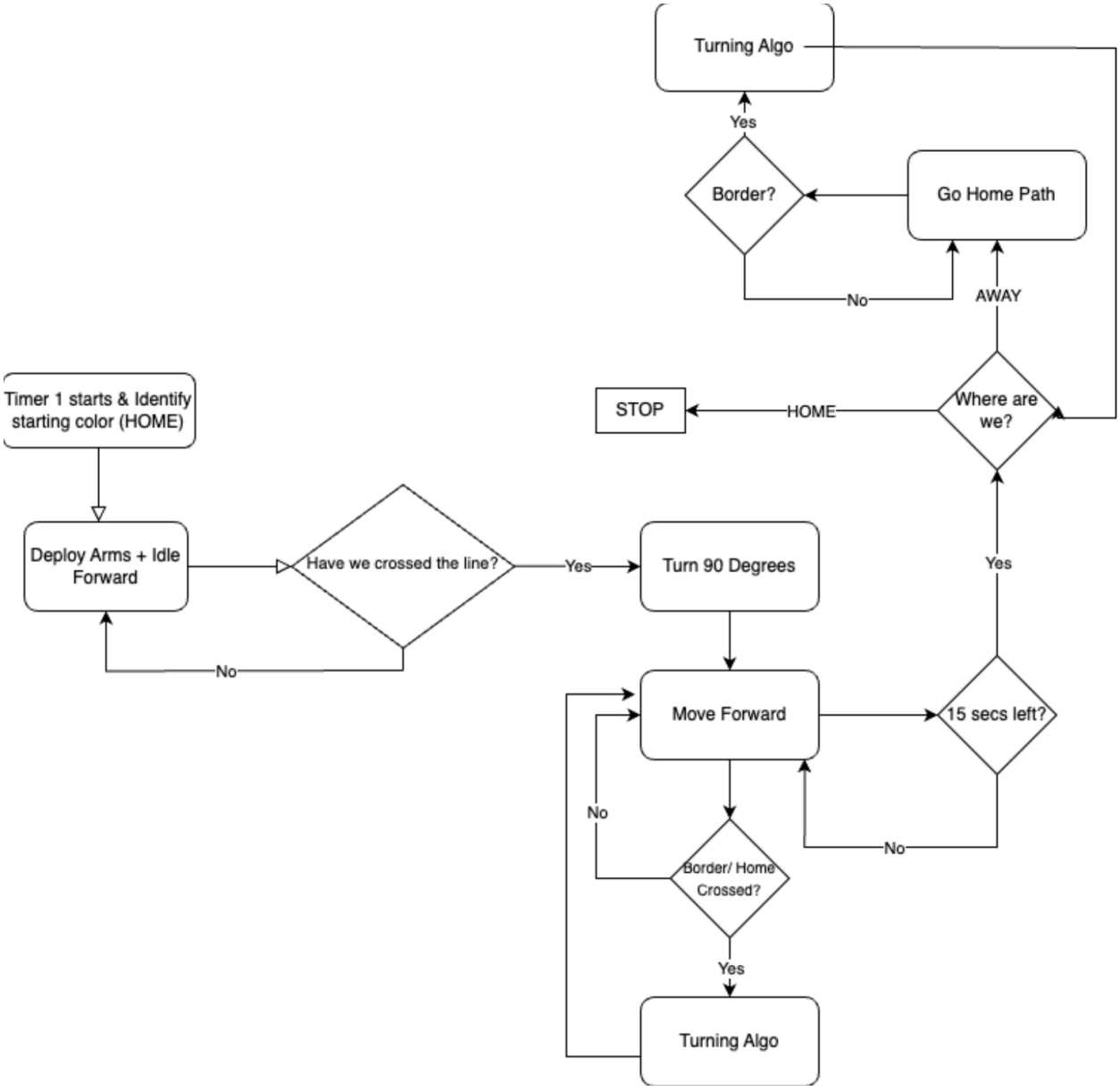




Reference used when making steel arms and plows



APPENDIX D: FLOWCHART



APPENDIX E: CODE

Acknowledgement: The color sensor code heavily references TA Stephan's code

Competition Code:

```
/*
Team 27
Team Members: Tyler Wisniewski, Ethan Sarpong, Chimdi Anude

HARDWARE CONNECTIONS:
PIN A3 & A2 -> Left H-Bridge Motor 3 4
PIN A1 & A0 -> Right H-Bridge Motor 5 6
PIN 0 ->
PIN 1 ->
PIN 2 ->
PIN 3 -> Left QTI Sensor R / pin change
PIN 4 -> Right QTI Sensor R / pin change
PIN 5 ->
PIn 6 ->
PIN 7 -> Color Sensor output/ pin change
PIN 8-11 -> S0-S3 on the Color sensor
PIN 12 ->
Pin 13 ->

WIRE COLOR CODE FOR BOARD:
*Not specified*

Note: The Left H-Bridge Green and Blue Wires were Flipped on the Bread Board
*/


//Global Variables
  int forward = 0b00001010; // pin A3 & A1 are HIGH 3, 5
  int backward = 0b00000101; // pin A0 & A2 HIGH 4, 6
  int right = 0b00001001; // PIN A3 & A0 HIGH (Right Wheel Backward - Left
Wheel Forward) 3, 6
  int left = 0b00000110; // pin A1 & A2 HIGH (Right Wheel Forward - Left Wheel
Backward) 4, 5

  // "period" : stores the value of the output wave period in microseconds
  volatile int period;

  // "timer" : stores the value of TIMER1
  volatile int color_timer;

  char homeColor; // of whatever we land on
  char awayColor; //
```

```
//Global Variables for QTI
  int PIN_QTI_LEFT = 0b00001000; // pin 3
  int PIN_QTI_RIGHT = 0b0010000; // pin 4

// Pin 12 for LED
  int led_toggle = 0b00010000;

// Internal timer
//80 ~= 10 sec  150 is about 20 s
  int htime = 80*4; // time in home ; every ten counts = 10 seconds
  int atime = 160; // time in away ;



ISR(PCINT2_vect) {
    if(PIND & 0b10000000){
      TCNT1 = 0x00;
    }
     else { // and stores the timer value in a variable ("timer") on a falling
edge (or vice versa).
      color_timer = TCNT1;
      //Serial.println("3");
    }
}


/**
 * Fit a number into a new range.
 */
float range(float x, float in_min, float in_max, float out_min, float out_max)
{
    return (x - in_min) / (in_max - in_min) * (out_max - out_min) + out_min;
}

int getColor() {
    // Set interrupt pin
    PCMSK2 |= 0b10000000;

    _delay_ms(5);

    // Divide the step count by 16 to convert from clock ticks
    // to microseconds, then multiply by 2 to get to full period
    int us = color_timer / 8; // (1/16) * 2

    // Reset interrupt pin
    PCMSK2 &= ~0b10000000;

    return us;
```

```
}

// Convert get color into a more user friendly format
char where_am_i_color (){
        // Read red color and map to 0-255 range
        int red = getColor();
        //int R = limit(range(red, 25.0, 435.0, 255.0, 0), 0, 255);
        // Serial.print("color:");
        // Serial.println(red);
        // _delay_ms(50);

        if (red<700 && red>115){
          //on blue
          return ('B');
        }
        else if (red<100 && red>1){
          //on yellow
          return ('Y');
        }
        else if (red>800){ //(3) Using the color sensor to detect black. If it
hits black in the middle does command
          PORTC = backward;
         // Serial.println("qti front");
          _delay_ms(300);
          PORTC = left;
          _delay_ms(300);
        }
}

void initColor() {
    // Enable interrupts globally
    sei();

    // Initialize interrupts
    PCICR |= 0b100;

    // Initialize the timer
    TCCR1A = 0b00; // normal mode
    TCCR1B = 0b01; // prescaler = 1
    TCNT1 = 0; // reset timer
}

void avoidBorder(){ //
    bool edge_left = PIND & PIN_QTI_LEFT; //return T or F
    bool edge_right = PIND & PIN_QTI_RIGHT;
    _delay_ms(10);

// identify qti sensing border and respond accordingly
    if(edge_left && !edge_right){
```

```
      PORTC = backward;
      _delay_ms(300);
      PORTC = left;
      //Serial.println("qti left");
      _delay_ms(250);
    }
    if(!edge_left && edge_right){
      PORTC = backward;
      _delay_ms(300);
      PORTC = right;
      //Serial.println("qti right");
      _delay_ms(250);
    }
    if(edge_left && edge_right){
      PORTC = backward;
      //Serial.println("qti front");
      _delay_ms(300);
      PORTC = left;
      _delay_ms(300);
    }
}

int main(void){
  init();
  Serial.begin(9600);

  //set all GPIO pins (as labeled above)
  DDRC = 0b00001111; //set pins A0-A3 as outputs
  DDRD = 0; // pins 3,4 QTI sensor, pin 7 Color Sensor Ouput
  DDRB = 0b00011111; // pins 8-12 are outputs (sensors)
  PORTB = 0b00000001; //20% output frequnecy with blue filter

  initColor();

  homeColor = where_am_i_color(); // current reading from getColor()

  // With the addition of black, the logic works better if we manually assign
the home color
  if (homeColor == 'B'){  //Checks Color. Sets the color as home and does the
opposite as away.
    homeColor = 'B';
    awayColor = 'Y';
  } // Works within a binary so the code only changes depending on yellow or
blue
  else{
    homeColor = 'Y';
    awayColor = 'B';
  }
```

```
  int i = 0; // internal counter.
  int int_strat= 0; //counter for to hard code initial strategy

  PORTC= forward;
  _delay_ms(1000);

  //Loop(1): Goes Forward and avoids borders.
  while( 1 ){  //Wrote this without testing but recommend playing with the time
    // htime and atime = total time
    PORTC = forward;
    where_am_i_color();
    avoidBorder();
    _delay_ms(10);

// move forward and then turn once you hit away color
    if (where_am_i_color() == awayColor && int_strat == 0){ //First time we hit
away color, turn 90 degrees
        _delay_ms(300);
        PORTC= left;
          _delay_ms(610); // CHECK THIS VALUE BY EXPERIMENTATION..... GOAL IS
90ISH DEGREES
        PORTC= forward;
        int_strat= 1; // this is a one time maneuver
    }

    //Loop (2): Stay on AWAY side until we hit our home border 3 times. on the
third, pass through
    while(where_am_i_color() != homeColor && i < 3){
      //using a while loop here helps avoid unnecssary polling and the robot
only focuses on what within this loop
        PORTC = forward;
        _delay_ms(50);
        where_am_i_color();
        avoidBorder();


        PORTB |= led_toggle; //Sets the led ON
        // c_delay_ms(5); //Delay for the led to flash

        if (where_am_i_color() == homeColor){
           PORTC = backward;
          _delay_ms(200);
          PORTC = left;
          _delay_ms(1000);
          i += 1;
        }
      }
```

```
    //Loop (3): Get to Home within the time limit and stop after safely within
Home
    while ( i > 3 && where_am_i_color() == homeColor){ // After the htime is up
and the home color is found.
      PORTC = forward; //Go forward and avoid borders
      _delay_ms(50);
      avoidBorder();
      where_am_i_color();
        PORTB ^= led_toggle; //Sets the opposite of whatever is in PIN 12
        _delay_ms(5); //Delay for the led to flash

       if (where_am_i_color() == awayColor){
           PORTC = backward;
          _delay_ms(250);
          PORTC = left;
          _delay_ms(800);
       }
    }
  }
}
```