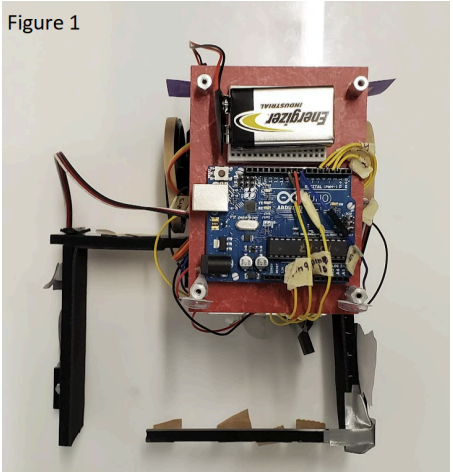
MAE 3780 Cube Craze Final Report

Group 19 – Alex Eagan, Evelyn Chiu, Nila Narayan

1. Robot Design and Strategy Overview

Figure 1



We utilized the following in our mechanical design; the kit chassis and wheels, a straight arm, a 90 degree arm, rubber, and a “curtain”. Our main cube-collecting mechanism included creating box-like arms that would “hug” the cubes on one side. These arms were created from acrylic, laser-cut rectangular pieces that screwed onto our robot. One side created the larger 90 degree arm, with two acrylic pieces that slotted onto each other that held a majority of the cubes. This stuck out from our chassis. The other side had a single piece that was meant to ensure blocks we collected would stay within our arm. On the surface of both arms, we glued a piece of rubber from McMaster to create a frictional surface to ensure blocks would remain within the arms. Each arm also had a QTI mounted on the outside with acrylic, glue, and screws. Upon testing, we noticed that whenever the robot turned, some cubes would fall from the arms. Based on this, we included a curtain design made from flaps of paper that helped hold the cubes in place but was weak enough to let them pass when we wanted it to. This was taped onto the top of the straight arm.

Electrically, we utilized a main color sensor that was housed underneath the chassis with the given mount. This gave it a covered housing that allowed us to minimize color distractions from opponents. We also utilized QTI sensors on the left and right arms to detect which side the border was on. If one arm detected black, the border was to one side of us, and if both arms detected black, the border was in front. We also utilized H-bridges and servos to power the wheels, which were connected to four pins on our Arduino. We also used the built-in timer in the Arduino to program a 60 second timer.

In the software that we implemented in the competition, we utilized two primary loops. Our strategy is described in the flow chart in appendix D. When starting off our robot, we had it face left towards the border, drive straight until it detected two black QTIs, turn right, and continue going straight until it reached the middle of the board (other color). These two 90 degree turns enabled more accurate alignment with the center of the board. Then the robot swept the middle, collecting the cubes there, and once it detected the black border on the right side of the board with both QTIs, it would turn right and deposit the cubes on our side by reversing. Then, we would enter our second loop, where the robot would turn 180 from its current position and make a loop using a similar path but with left turns instead. The second loop ideally occured around the middle of the other team’s color, collecting whatever cubes

were there until it returned back. Then, our robot would ideally hang around the middle of the board, making short turns to collect the remaining cubes in the center of the board.

While these loops were running, we had QTI sensors to detect the border and to make small turn adjustments based on whether the border was on both sides (meaning we would need to fully reverse), the left side (meaning we would need to turn slightly right), or the right side (meaning we would need to turn slightly left). We also had a timer running for the sixty-second match – once fifty seconds were up, if we were on the opposite side that we had started on, our robot would turn 180 degrees after it detects a border and keep moving forward. The purpose of this is that either our robot would turn back to our side and stop, or if it was along the left and right border, it would simply run off the board with any cubes it had so that our opponent would have less cubes that counted on their side.

2. Design Process Reflection

Our group achieved the first two milestones fairly quickly. In order to accomplish our design pitch, we first sat down and had a brainstorming session together. Getting all of our ideas out, even if they ended up being unrealistic, helped us to better contextualize what our robot needed to accomplish and how we could do so the most efficiently. Our design pitch went smoothly and we were able to accomplish the second milestone quickly by duplicating the code for the one motor we created in lab, and making small adjustments.

We found milestone three to be the most challenging. Firstly, we went through technical issues when all three of our kit-provided color sensors were malfunctioning. This took a lot of debugging, in which we individually replaced and tested all circuit/Arduino components, as well as went through a debugging process with our code. Eventually, when we got a new sensor, the testing process improved. At this point, we had also not been able to successfully configure our QTI sensors. Because of this, we attempted to configure two separate color sensors, to serve a dual purpose of sensing the yellow-blue boundary and the black edges. However, we were unable to configure the timers in such a way that both color sensors worked. We were able to complete the milestone despite this, and ended up configuring and testing our QTI sensors afterward. For milestone four, we had not been able to laser-cut the final version of our arms in time; to solve this issue, we created cardboard prototypes of our arm. Our completion of milestone four left us with only three days until competition, and we had to reject some of our initial ideas. Specifically, we cut the motorized arm and sonar sensor from our strategy.

Most of our hardcore testing occurred after we already completed all the milestones. We went through iterative prototyping of our arm. Our initial design was too difficult to mount stably, so we went through a redesign, which ultimately improved our overall performance. We also added our “curtain” component, as described in our Strategy Overview. Similarly, the testing of our code was also iterative. We would program and test parts of the loop at a time (see Appendix D for flowchart). Because our driving and turns were time-based, we followed a

“guess and check” methodology. At this point, our color sensors and QTI were already working consistently, so we were primarily testing our strategy itself. We consistently ran into a “Programmer Out of Sync” error on the Arduino IDE. Our process for debugging this involved: ruling out our code as the issue first, swapping out circuit components one by one, and finally asking Professor Kress-Gazit for help. Our final conclusion was that our USB-C to USB-A adapter may have been causing the issue.

3. Competition Analysis

Overall, our robot did well during competition day. We lost the first match, as we went up against a large robot, met in the middle, and we ended up getting pushed off of the edge. However, for the next five matches in the round robin, our robot performed exactly as we intended regardless of whether it met another robot head on our not – except when both ended up going off the edge of the board while interlocked. The programming of our robot to account for different possibilities in every while loop allowed for our robot to recover and even push some other robots off of the mat, which was one of our largest strengths. However, in our first round of 16 match, our robot never turned when hitting a color change and drove off the board without interference from the other robot. We believe that this was because for some reason our color sensor did not fully read, as there was nothing in our testing nor our previous matches to have us guess this would happen. Because we never read a color change, we did not end up getting blocks as planned which lost us that match.

4. Conclusions

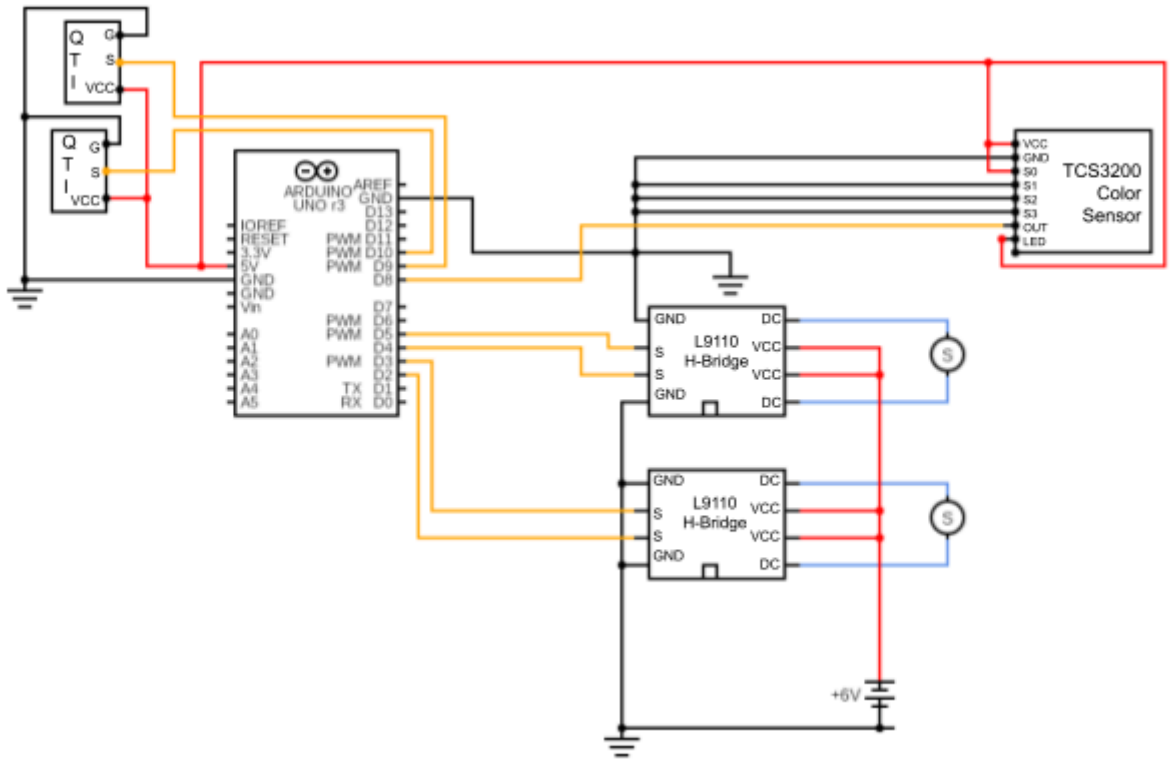
Overall, we feel that we had good time management throughout this project. While we wish we had more time for testing in order to be able to implement a sonar sensor, we feel we made a successful robot within the time constraints. We also found it very beneficial that we had some scrimmages with other teams, in order to better simulate real matches rather than only testing an ideal scenario. Regarding our mechanical design, other teams implemented a trapping strategy – that is, a way to not only collect the blocks but also not let them go afterward- that would have helped our performance. Given a second opportunity, we would implement a more offensive strategy using more of our budget (we only used half of our budget), such as messing up our opponent’s color sensors with LEDs or with black paper.

Our advice to future students is to achieve the milestones quickly, to maximize time spent testing. Many teams end up with similar designs and strategies, so the more time dedicated to making your robot respond well in unexpected circumstances, the more likely you are to excel. That said, it is also important to know when to cut your losses, as above all else this competition relies on the consistency and reliability of your robot. Programming ends up being more significant than mechanical design.

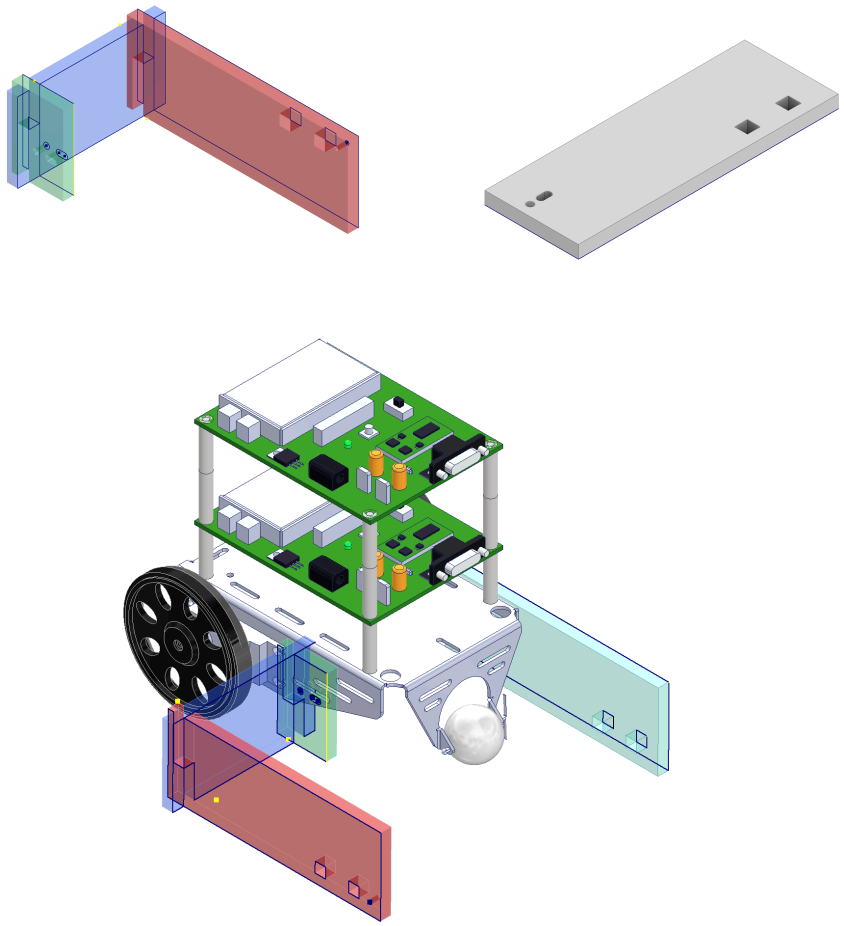
Appendix A: Bill of Materials

Item	Vendor	Qty	Price
Provided chassis, wheels, color sensor, color sensor housing, QTI sensor (x2), miscellaneous wires, standoffs, screws, etc.	MAE 3780	N/A	$0.00
Rubber Sheet Strip	McMaster Carr	1	$3.52
Zip Ties	McMaster Carr	1	$3.53
12”x12”’ x 6mm Acrylic	RPL	1	$5.00
RPL Laser Cutting Costs ($.50/part, $.05/cut)	RPL	4 parts, 52 cuts	$4.60
		Total	$16.65

Appendix B: Circuit Diagram



Appendix C: CAD files, drawings



Figures 2, 3, 4: the design of our “wide arm”, consisting of 3 acrylic pieces which slot together (top left), our “side arm” design (top right), and a full CAD assembly (bottom)

Appendix D: Flowchart

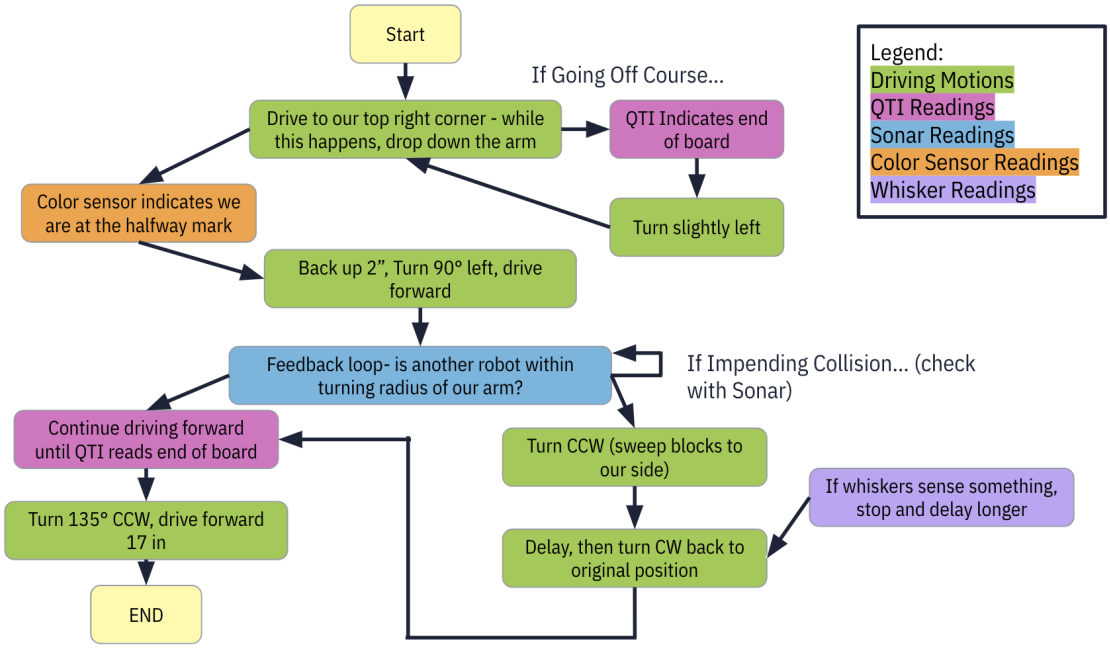


Figure 5: Proposed flowchart in milestone one

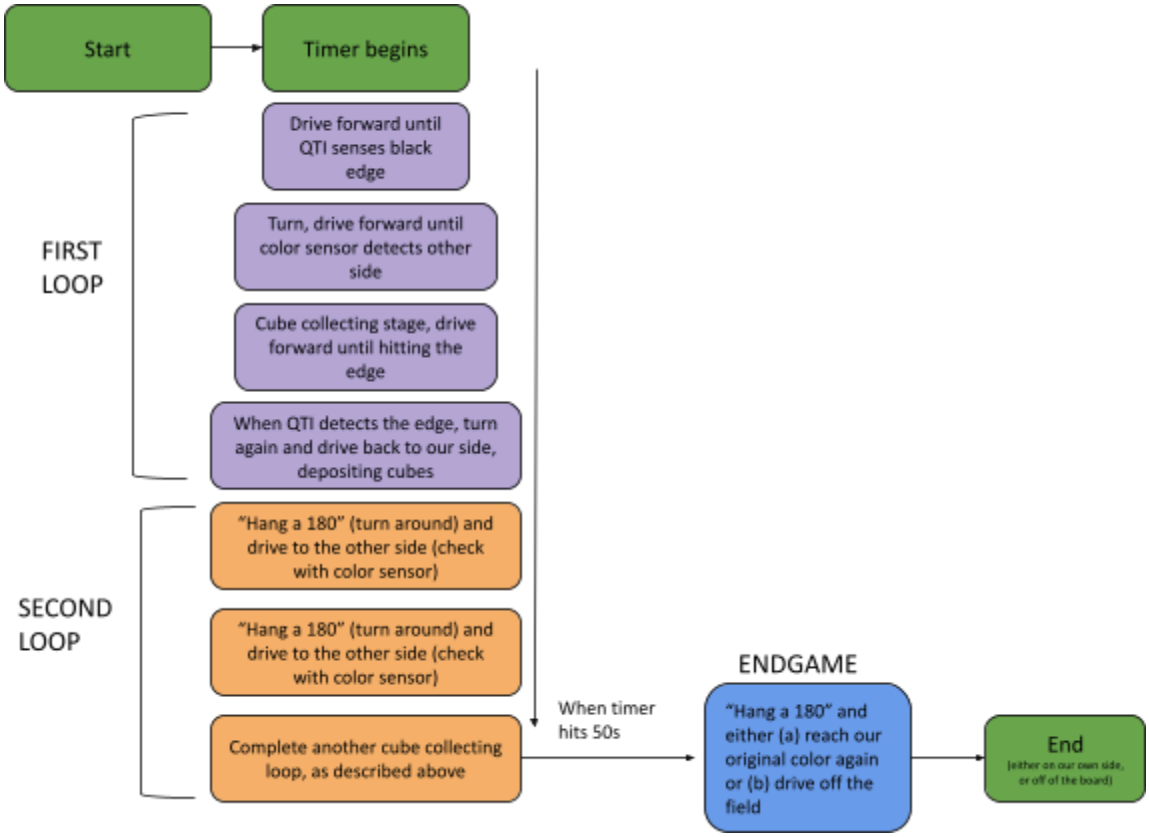


Figure 6: Actual strategy implemented during competition

Appendix E: Code

```
/*
 * Mechatronics Robot Project
 *
 * Nila Narayan, Alex Eagan, Evelyn Chiu
 */

//**********************MOTOR CODE*********************************//
int inch;
int ffdegs;

int driveB() //Driving Backward
{
  motoroff();
  //Serial.println("DriveB");
  _delay_ms(10);
  PORTD |= 0b00010100; //Set pins 2,4 on for forward: Pins 2,3 pair, 4,5 pair
}

int driveF() //Driving Forward
{
  motoroff();
  //Serial.println("DriveF");
  _delay_ms(10);
  PORTD |= 0b00101000; //Set pins 3,5 for backward: Pins 2,3 pair, 4,5 pair
}

int turnR() //Turn to the right, one motor forward, one back
{
  motoroff();
  _delay_ms(1);
  PORTD |= 0b00011000; //Set pins 3,4 for left: Pins 2,3 pair, 4,5 pair
}

int turnL() //Turn to the left, one motor forward, one back
{
  motoroff();
```

```
  _delay_ms(1);
  PORTD |= 0b00100100; //Set pins 5,2 for right: Pins 2,3 pair, 4,5 pair
}

int motoroff() //Turn all motors off
{
  PORTD &= 0b11000011; //Set pins all off: Pins 2,3 pair, 4,5 pair
  _delay_ms(10);
}
//********************END MOTOR CODE************************//

//******************COLOR SENSOR CODE************************//
int period;
int timer;

ISR(PCINT0_vect)
{

 //Serial.println("ISR triggered");
 //code to reset timer to 0 on rising edge, and stores timer in variable on falling edge
 if (PINB & 0b00000001){ //check if interrupt is active
  TCNT1 = 0;
 }
 else{ //falling edge
  timer = TCNT1;
  //Serial.println(timer);
 }
}

void initColor()
{

 sei();

 //set pin 9 = 0 by setting pin 1 in port B as output
 DDRB = DDRB | 0b00000010;
```

```
  //enable pin change interrupt
  PCICR = 0b00000001; //enable PCINT0

  //initialize timer
  TCCR1A = 0b00000000; //don't need this, TCCR1A is 0 by default
  TCCR1B = 0b00000001;

}

int getColor()
{

  //enable specific bit for pin change interrupt
  PCMSK0 = 0b00000001; //enable PCTINT0 aka pin 8

  _delay_ms(5);

  //disable specific bit
  PCMSK0 = 0b00000000;

  //timer measures half a period, so convert this to time
  period = (timer + 1) * 2 * .0625;
  //Serial.println(period);
  return period;
}

//*****************END COLOR SENSOR CODE***********************//

//*******************QTI SENSOR CODE**************************//
int qtiBlack = 1250; //What value is black
int rQTI() //(pin 9)
{
  DDRB = 0b00000010;//Initialize output pin 9
  PORTB = 0b10; //Set high
  _delay_ms(1);//Let capacitor charge
  PORTB = 0b00; //Set Low
```

```
 DDRB = 0b00000000;//Initialize input pin 9
 int testr = 0;
 int done = 0;
 while(done == 0){
  testr = testr + 1;
  if(PINB & 0b10){
   //Serial.println("h");
  }
  else{
   //Serial.println("l");
    done = 1;
  }
 }
 //Serial.println(test); - Use to debug w/greyscale value
 if (testr > qtiBlack){
  return 1;
  }
 else{
  return 0;
  }
 }

int lQTI() //(pin 10)
{
 //Serial.println("ReadinglQTI");
// _delay_ms(20);
 DDRB = 0b00000100;//Initialize output pin 10
 PORTB = 0b100; //Set high
 _delay_ms(1);//Let capacitor charge
 PORTB = 0b000; //Set Low

 DDRB = 0b00000000;//Initialize input pin 9
 int testl = 0;
 int done = 0;
 while(done == 0){
  testl = testl + 1;
  if(PINB & 0b100){
```

```
    //Serial.println("h");
   }
   else{
    //Serial.println("l");
     done = 1;
   }
  }
  //Serial.println(testl); //- Use to debug w/greyscale value
  if (testl > qtiBlack){
   return 1;
   }
  else{
   return 0;
   }
  }


//********************END QTI SENSOR CODE************************//

//********************START TIMER CODE************************//
   int timesteps = 0;
   int endgame = 2769; //60.2 ticks per second 3010 for 50s

ISR(TIMER0_OVF_vect){ //Timer interupt
  TCNT0 = 0;
  timesteps = timesteps + 1;
  //Serial.println(timesteps);
}

//********************END TIMER CODE************************//

int start_color;
int curr_color;
bool lqti;
bool rqti;

int main(void)
```

```
{

//  Serial.begin(9600);
//   Timer CODE

   TCCR0A |= 0b00000000; //Normal mode operation
   TCCR0B |= 0b00000101; //Prescaler 24

   TIMSK0 |= 0b00000001; //Enable OVerflow interupt
//   END Timer Code

  //set up wheels
  DDRD = 0b00111100; //set pins 2,3,4,5  to output, Pins 2,3 pair, 4,5 pair

  //Sometimes the initial color reading is off, so read it a second time after a small delay just in
case.
  initColor();
  _delay_ms(250);
  start_color = getColor();
  _delay_ms(500);
  start_color = getColor();
  bool isBlue = false;  //assume we start on yellow
  if (start_color > 250) isBlue = true;

  int blackMinPeriod = 400;
  int yellowMaxPeriod = 175;
  int blueMaxPeriod = 350;
  bool startout = true;
  bool cube_collecting = false;
  bool cubes_home = false;
  bool offense = false;

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

    while (1)
    {
```

```
    // start of every while loop, figure out where we are
    curr_color = getColor();
    lqti = lQTI();
    rqti = rQTI();

    // just starting out, going from the middle to the edge, lining up to the point where we can
turn
    while (startout)
    {
      curr_color = getColor();
      lqti = lQTI();
      rqti = rQTI();
      //Serial.println("Startrqti");
      //Serial.println(rqti);
      //Serial.println(lqti);

     // both the sensors have crossed the line, so we hit the edge and need to turn now
     if (rqti && lqti) //&& curr_color > blackMinPeriod
     {
      //Serial.println("Startbothqti");
      //_delay_ms(100);
      motoroff();
      _delay_ms(100);
      //driveF();
      //_delay_ms(100);
      turnR();
      _delay_ms(450);
      lqti = lQTI();
      _delay_ms(2);
      while(lqti){
        turnR();
        lqti = lQTI();
        _delay_ms(50);
      }
      motoroff();
     }
```

```
      // we have reached the middle, so turn with the big arm
      else if ((isBlue && curr_color < yellowMaxPeriod && !lqti && !rqti) || (!isBlue &&
(curr_color > yellowMaxPeriod) && !lqti && !rqti))
      {
        //Serial.println("Start-nowdiffcolor");
        startout = false;
            //I am unsure, do  we need to go forward here, eg where will we turn to
        driveF();
        _delay_ms(520);
        motoroff();
        _delay_ms(100);
        turnR();
        _delay_ms(450);
        motoroff();
        cube_collecting = true;
      }
      else
      {
        //Serial.println("StartDriveF");
        // we haven't driven up to any lines yet, no qti sensed
        driveF();
      }
      _delay_ms(75);
    }

    while (cube_collecting)
    {
       curr_color = getColor();
       lqti = lQTI();
       rqti = rQTI();
       //Serial.println("cubecollqti");
       //Serial.println(rqti);
       //Serial.println(lqti);
      // this phase we are collecting cubes at the middle
      if (rqti && lqti)
      {
```

```
    // both the sensors have crossed the line, so we hit the edge
    motoroff();
    _delay_ms(100);
    driveB();
    _delay_ms(100);
    turnR();
    _delay_ms(500);
    motoroff();
    cube_collecting = false;
    cubes_home = true;
  }
  else
  {
    // should just be driving straight
    driveF();
  }
  _delay_ms(75);
}

int timfor = 0;
while (cubes_home)
{
    curr_color = getColor();
    lqti = lQTI();
    rqti = rQTI();
    //Serial.println("cubehomeqti");
    //Serial.println(rqti);
    //Serial.println(lqti);
  // bring cubes collected from the start back to the base
  if (rqti && lqti)
  {
    // reached the end of the board again - Turn and deposit forward
     motoroff();
    _delay_ms(5);
    turnR();
    _delay_ms(750);
    driveF();
```

```
      _delay_ms(1000);
      motoroff();
    }

     else if (lqti)
    {
      // Side QTI hit, turn a bit
       motoroff();
      _delay_ms(1);
      turnR();
      _delay_ms(75);
      motoroff();
    }

    else if(timfor>750)
    {
      //Serial.println("timfor surpassed_____________________");
      motoroff();
      cubes_home = false;
      offense = true;
      // leave the cubes, reverse a bit
      driveB();
      _delay_ms(1800);
      motoroff();
    }
    else
    {
      driveF();
      _delay_ms(50);
      timfor = timfor +50;
      //Serial.println(timfor);
    }

    _delay_ms(75);
   }

//-------------------------------------------------------------------------------------------------------------
```

```
    int offstage;

    //First loop complete, we want to go around again and collect more blocks
    while (offense)
    {
       curr_color = getColor();
       lqti = lQTI();
       rqti = rQTI();
       //Serial.println("offense");
       //Serial.println(rqti);
       //Serial.println(lqti);

      //While we have not yet reached the 50s mark
      while(timesteps<endgame)
      {
       //Serial.println(timesteps);  //HERE is where to add loop number 2
       offstage = 1;
       turnR();
       _delay_ms(1000); // hanging a 180

       while (offstage == 1 && timesteps<endgame)
       {

        //turnR();
        //_delay_ms(1000); // hanging a 180

        curr_color = getColor();
        lqti = lQTI();
        rqti = rQTI();

        //When we reach their side, turn to begin cube collecting
        if ((!isBlue && (curr_color > yellowMaxPeriod))||(isBlue && (curr_color <
yellowMaxPeriod))) // if we are on their side
        {
          offstage = 2;
          driveF();
```

```
  _delay_ms(1600);
  turnL();
  _delay_ms(425); // drive a little to get out of the middle
 }

 //If we are driving at a slant, correct ourselves to get away from the edge
 else if (lqti && !rqti)
 {
  turnR();
  _delay_ms(300);
 }
 else if(rqti && !lqti)
 {
  turnL();
  _delay_ms(300);
 }
 else
 {
  driveF();
  _delay_ms(50);
 }
 // offstage 1 should get us on their side
}

//Drive forward and collect cubes
while (offstage == 2 && timesteps<endgame)
{

 curr_color = getColor();
 lqti = lQTI();
 rqti = rQTI();
 //turnR();
 //_delay_ms(525);

 //If we are driving at a slant, correct ourselves to get away from the edge
 if (lqti && !rqti)
 {
```

```
          turnR();
          _delay_ms(300);
         }
         else if(rqti && !lqti)
         {
          turnL();
          _delay_ms(300);
         }
         else if (lqti && rqti)
         {
          turnL();
          _delay_ms(1000);
         }
         int Fofftwo = 0;
         while(Fofftwo < 2500){
          driveF();
          _delay_ms(50);
          Fofftwo = Fofftwo + 50;
         }
         turnL();
         _delay_ms(750);
         offstage = 3;
        }

        //Bring the cubes back to our side!
        while (offstage == 3 && timesteps<endgame)
        {
         curr_color = getColor();
         lqti = lQTI();
         rqti = rQTI();
         if ((isBlue && (curr_color > yellowMaxPeriod))||(!isBlue && (curr_color <
yellowMaxPeriod)))
         {
          driveF();
          _delay_ms(1500);
          driveB();
          _delay_ms(800);
```

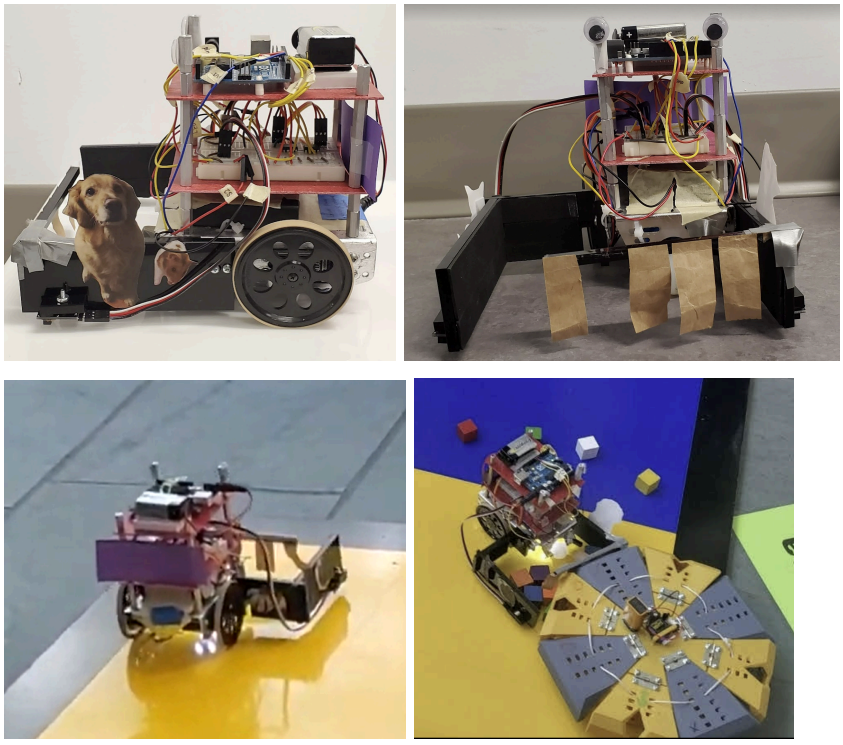
```
          //offstage = 1;
          turnR();
          _delay_ms(1500);
         }
         else if(lqti || rqti)
         {
          motoroff();
          _delay_ms(1000);
         }
         else
         {
          driveF();
          _delay_ms(50);
         }
        }
    }

    //After this you should no longer have loop 2 involved

     //now in endgame - meaning we are in the last 10 seconds of the match
     if((!isBlue && (curr_color > yellowMaxPeriod))||(isBlue && (curr_color <
yellowMaxPeriod))) //We are on the wrong color in endgame
     {
      //Drive forward, while incorrect color
      //IF within 3s hit black,
      //hang 180
      //Drive F while incorrect color
      while((!isBlue && (curr_color > yellowMaxPeriod))||(isBlue && (curr_color <
yellowMaxPeriod)))
      {
       curr_color = getColor();
       lqti = lQTI();
       rqti = rQTI();
       if((lqti || rqti) && timesteps < (endgame + 240)) //Endgame plus short forward
       {
        turnR();
        _delay_ms(1200); //hang 180
```

```
          motoroff();
         }
         else
         {
          driveF();
          _delay_ms(50);
         }
        }
        driveF(); //extra boost to get off the board
        _delay_ms(50);
       }
       else
       {
        motoroff();
       }
      }
    }
} //Ending Bracket for main
```

Appendix F: Additional Photos



Figures 7-10: Additional photos of the robot in lab (top) as well as in competition (bottom)