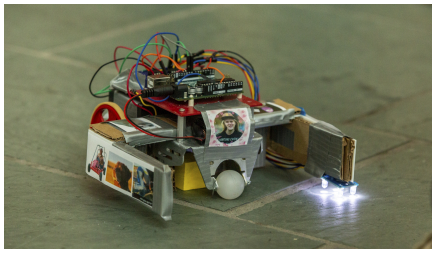
Mackemey Munion (mcm329), Ava Ianuale (ai244), Zoe Szymanski (zs5)
Group 33 - Zamboni

1. **Robot Design and Strategy Overview**.

Mechanically, we went for a simple and lightweight 3D printed ABS arms with cardboard reinforcement design. This allowed us to accelerate faster towards the blocks compared to the bulkier sheet metal designs, however when the paths collided with a larger robot, the lightweight design did not benefit us. We used the same wheels, but smaller rubber bands to both increase the radius of the given wheels and also ensure the tires did not fall off during a run.



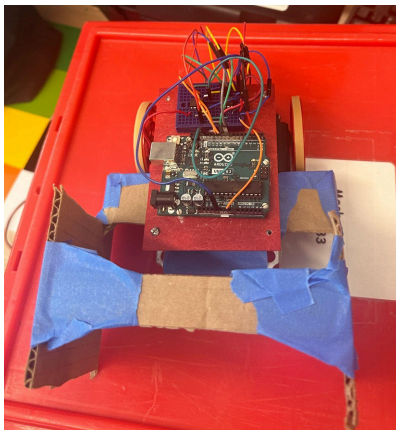
We used one arduino, one small breadboard, 9V battery to power the arduino, a 6V battery back to power the h-bridges, and a color sensor. A set path was hard coded and implemented in the main loop (see appendix D, E). Once the path was completed, the robot waited in place until it detected black with the color sensor. During testing, we found that our robot would collect more cubes if it came to a complete stop rather than continuing to move around the board to collect more cubes due to the turning resulting in cubes leaving the perimeter. This border detection feature was created with an interrupt in the while loop, seeBlack(), (see appendix E) so that if the border was detected due to the other robot pushing ours out of the board, it would push backwards towards the other robot, turn around 180deg, and move forward about a foot to avoid falling and potentially losing blocks.

2. **Design Process Reflection**.

The overall design of our robot changed and adjusted over the course of the checkpoints and for the competition. We started our design in a brainstorm phase where we identified the key features we need to have to pick up the most blocks. The features we picked were arms that extend the maximum distance allowed by the rules (either 12” Dia or 8”X8”), a way to ensure the blocks stay within the robot, and a simple design that will not be over complicated by mechanical components. With these three points in mind we started on the initial design for checkpoint 1 which was our strategy presentation.

Our initial design had fixed arms that extend the entire 8” of width starting at the center of the robot and extending to the 8” length. We decided to not “deploy” arms to the entire 12” area because of possible concerns with having a reliable mechanism. Additionally the space gained in this deployed range does not give a significant benefit for our driving plan so the needed complexity would have been overkill. At the top of these “arms” we planned to have a mechanical intake with a small servo attached to the sides. This intake would be continually spinning and would bring in blocks while having the added benefits of not letting the blocks out. We also planned to have a color sensor

under the base of the robot and use the given wheels. We picked the given wheels due to budget concerns because we knew a large portion would go to 3D printing and the smaller wheels let us have more block storage space.



The next step we took was prototyping our arm design. In preparation for the 4th checkpoint of picking up blocks we decided to use scrap cardboard around the shop to test out arm design and its rigidity. Before we started our scrappy prototype we evaluated our baseline design and decided that if we could pick up more than 10 blocks without the mechanical intake we would get rid of it from our plan. This decision was made because in the process of CADing out arms we determined that it would be more important to have stronger arms than a machinal intake. Additionally we had concerns about the weight increase to the robot as our design, while robust, does need a certain amount of speed so we can reach the blocks before they are taken. For the prototype arms we added a top support to give a bit more stability since it was only one sheet of cardboard. We believe this would make the test more accurate to the ABS it would be printed in. From our initial test of this milestone we collected 12 blocks which proved to us that we did not need the intake. However after more testing we noticed that blocks would get under the robot and specifically around the wheels which was concerning because if the blocks removed the rubber bands out robot would lose traction and fail. To fix this issue we added walls with the cardboard and later the 3D printed design.

After the success of milestone 4 we committed to print our arm design. To maximise the possible length and thickness of the part we added weight saving cutouts to the arms. We also decreased the thickness of the back walls since their only job is to prevent blocks from leaving. Once we installed the arms we also added cardboard around the arms to increase the strength. This resulted in the added benefit of surface area for photos but this was not design critical only morale. The last design change we made was moving the color sensor to be mounted under the arm instead of the robot. This was done because during testing we saw that by the time the sensor would read a value of a changed color the blocks had fallen off the board because the sensor was initially mounted under the chassis. While in this competition they could have still counted for us, when the blocks fell off the board we would lose control of them and either have the robot stop moving or the blocks would come loose. By moving the sensor to the arm we could detect a change in color before the blocks fell. After our many design changes and differences we were as prepared as possible for the competition day.

3. **Competition Analysis**.

Relative to other robots at competition, our robot was much lighter which worked in our benefit for speed, but when we collided paths with a heavier robot the heavier robot would continue on its path and off track ours. Given that our initial path was hard coded, we left a 4” border between the outermost path and the side of the robot's perimeter to account for any misalignment from the path. This was successful until the robot faced head-on collision in which our robot had flipped over completely (however, had the cubes we landed on counted as being in the perimeter, we would have won the match). In the matches where by luck our robot did not collide with the other robot, we were successful in collecting the most cubes due to our speed and maneuverability.

Additionally, due to the simplicity of our design, our robot could hold many cubes compared to some of the other more complex designs that incorporated intake systems and dropping arms. Because we had a large and open perimeter, we were able to incorporate other tricks using the additional budget we had leftover. One of these being adding double wrapped duct tape to the inside walls of the perimeter. The robot competition rules only said that duct tape could not be used on the outside of the perimeter because that would not count as being inside the perimeter, so adding this feature was legal. In the matches we won, we had collected around 9-12 cubes.

We encountered minor user errors at competition, such as the battery pack wire being unplugged from the breadboard and thus the h-bridges weren’t powered at the start of a match. Ultimately we scored 2-5 during the round robin match. 2 of our losses were from being pushed around by another robot and 1 was from user error.

4. **Conclusions**.

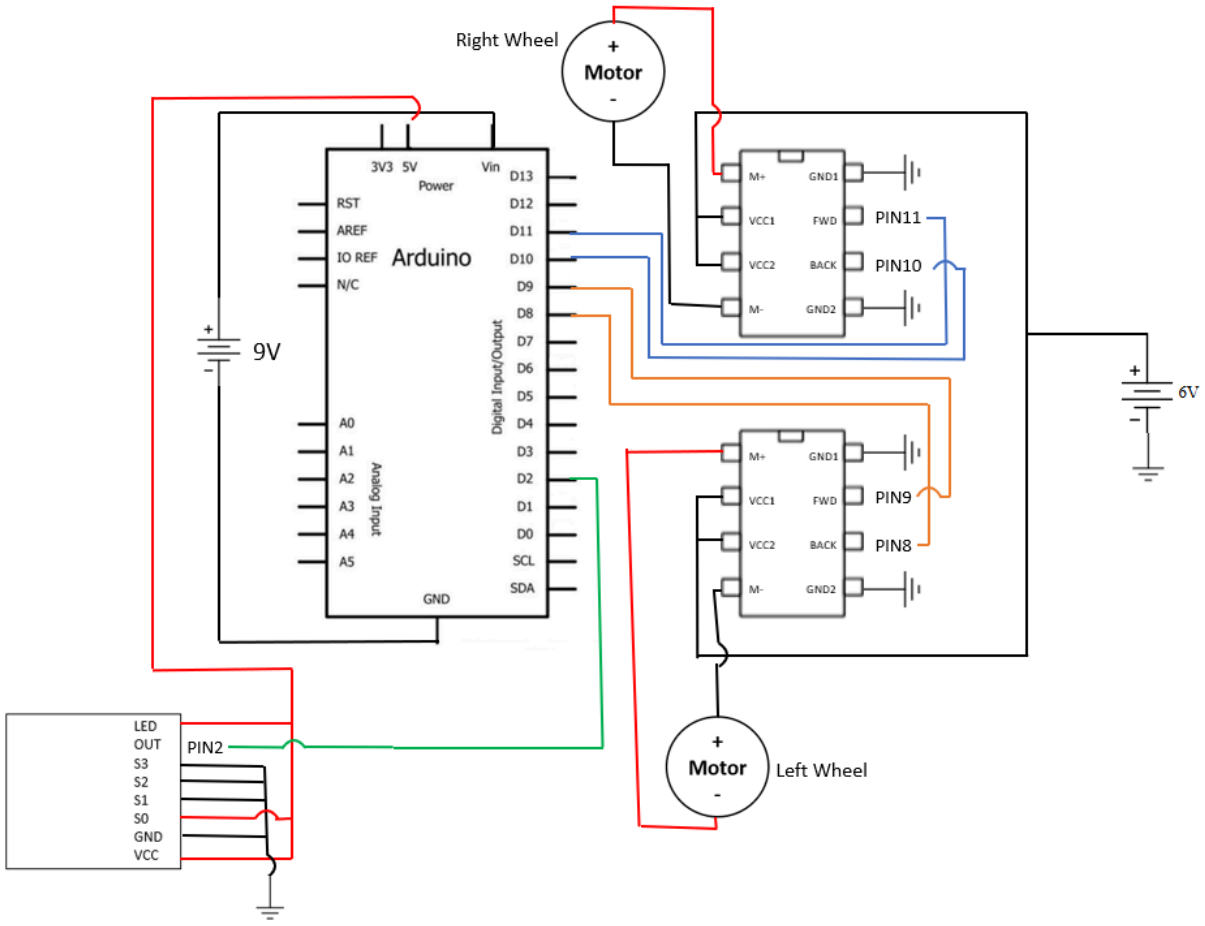
Our main disadvantage was the weight of our robot. To maintain our speed and large perimeter we would likely opt for acrylic arms that would still allow us to display our photos along the side. Additionally, we would likely purchase larger wheels from Polulu to get a larger distance traveled per rotation of the motors. The addition of acrylic and new wheels would likely get us close to the budget, so we would probably budget for these two items.

Additionally, one idea we had before competition, but did not have the time to fully implement due to needing to calibrate the path our robot travels would be getting more power to the h-bridges instead of using just the 6V battery. Due to the competition rules only allowing one 9V battery and on 6V battery pack, we would need to figure out a way to connect the 6V to the arduino unsteady of the 9V so we would use the 9V to power the h-bridges. According to the L9110 H-bridge documentation, Vcc can have a max voltage of 12V, so 9V would be acceptable to use. Alternatively, we could keep using the current setup, but use an op amp to increase the input voltage to the h-bridge.

5. **Appendix A**: bill of materials (BOM). Your bill of materials only needs to include additional parts that your group purchased and included in your budget, you do not need to include parts from your lab kit. Be sure to also include any purchased parts that did not end up being used in the final design (3D prints, lasercut components, etc.).

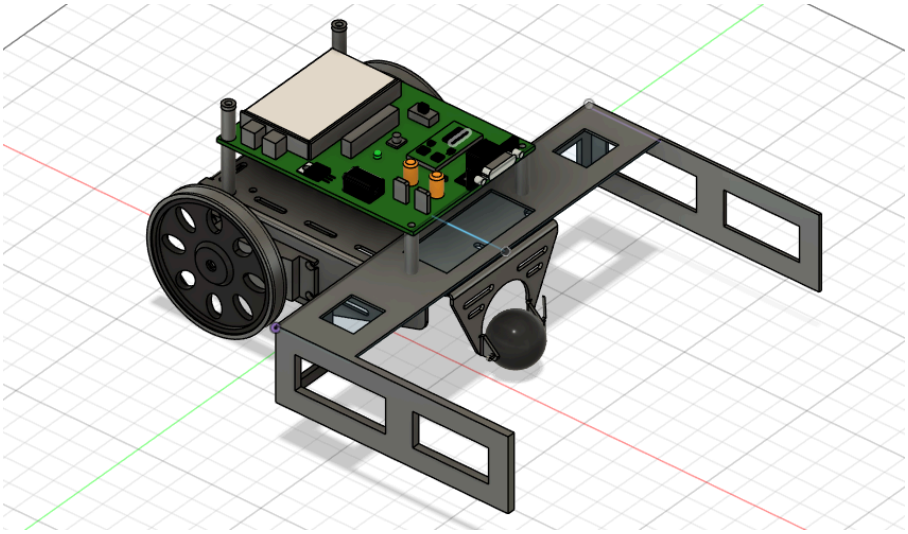
Vendor	Name of Part	Part Number	Unit Cost	Quantity	Total Cost
McMaster	Duct Tape Silver, 2" Wide, 180 Feet Long, 0.008" Overall Thickness	7788A2	6.34	1	6.34
McMaster	Rubber Bands Beige Natural Rubber, Size 62, 1/4" Wide x 2-1/2" Long	12205T111	12.48	1	12.48
RPL	Robot Arms	--	11.240325	1	11.240325
(provided in lab, comparing price to McMaster)	Cardboard	20895T104	1.82	1	1.82
					31.880325
RPL Order					
Material	**Density (g/cm^3)**	**Volume (cm^3)**	**Cost**		
ABS	1.05	18.553	11.240325		

6. **Appendix B**: circuit diagram. This should be a schematic view (using symbols to represent resistors, capacitors, etc.). Do not submit a Tinkercad “breadboard view” image or you will receive a zero for this section. A Powerpoint template with different circuit diagram symbols is available on Canvas, but you can use any software you want to generate the schematic.

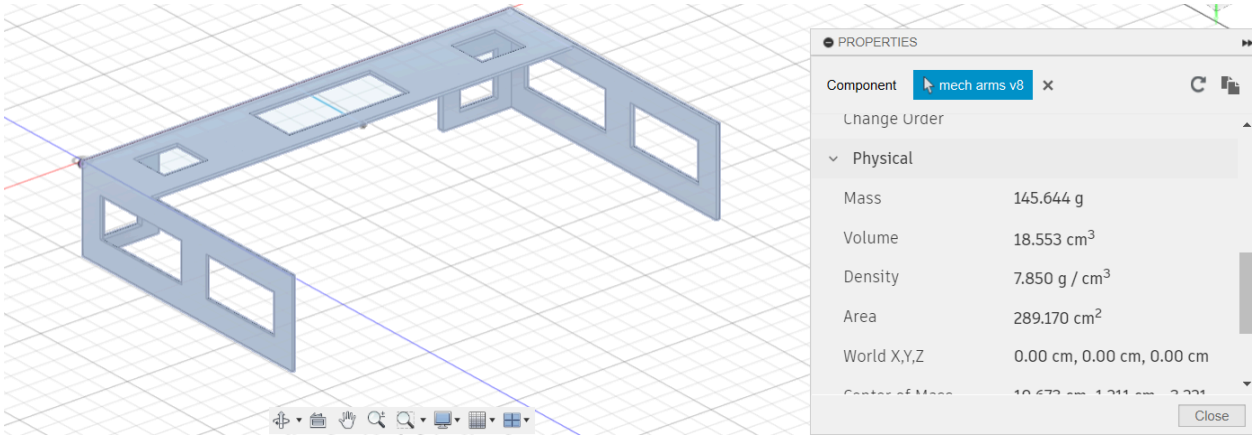


Circuit Diagram

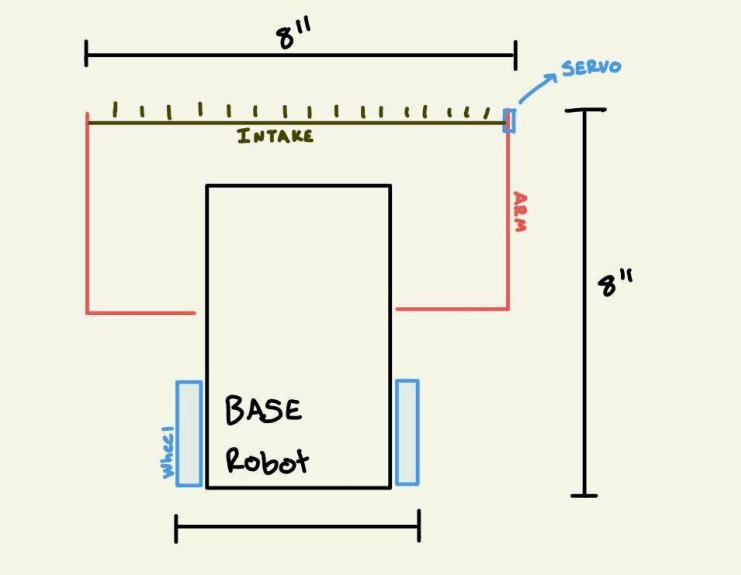
7. **Appendix C**: CAD files and drawings. Include screenshots of relevant CAD files or drawings used for manufacturing individual parts. Also include screenshots of part volumes and weight calculations (or photos of the parts being weighed) corresponding to the lines in your BOM.



CAD of Final Robot Design (No Cardboard)

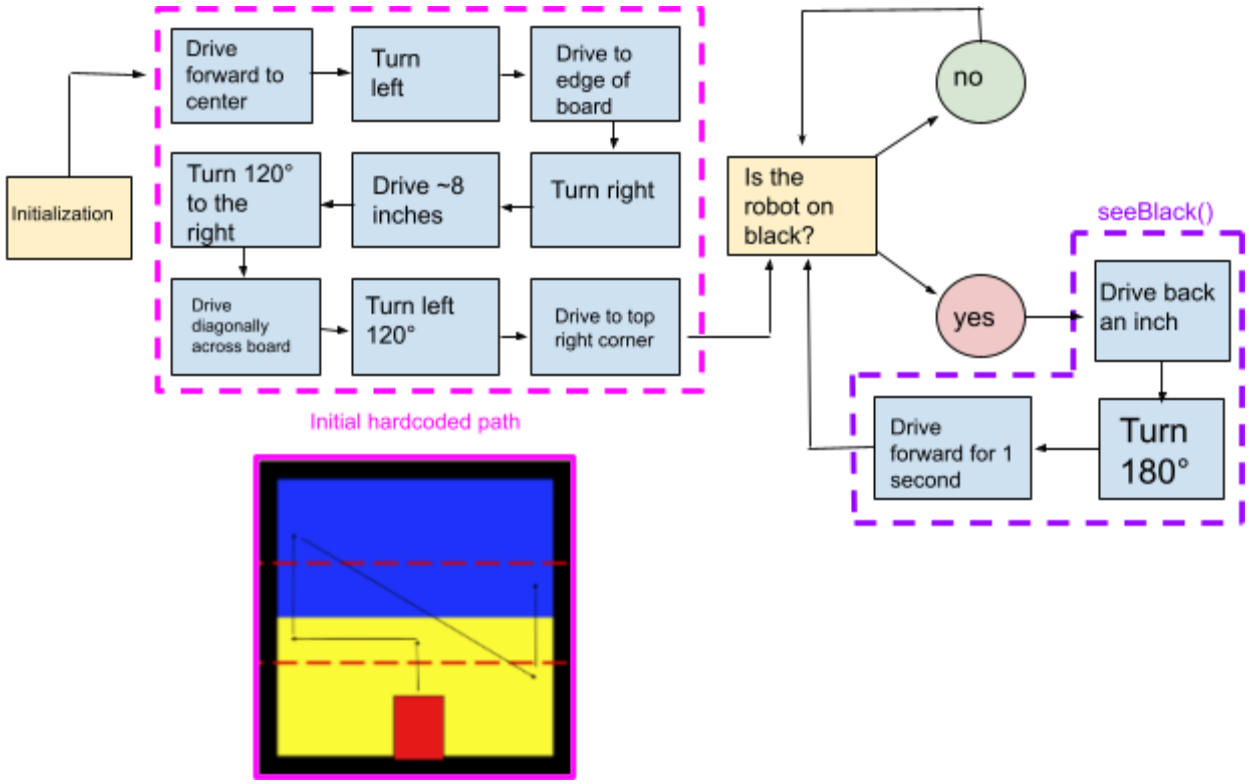


CAD of Arms and Weight Properties (Line 3 of BOM)



Initial Design Drawing for Robot (Checkpoint 1)

8. **Appendix D**: Flowchart. Your robot's strategy/algorithm represented in flowchart form. The flowchart does not need to represent every single line of code, but it should show high-level behaviors and functions.



9. **Appendix E**: Code: commented Arduino code. A clear understanding of each line of code should be demonstrated. References must be provided if used.

```
// "period" : stores the value of the output wave period (not necessarily
in seconds)
// "timer" : stores the value of TIMER1
int timer=0;
// note: because of our choice of prescaler,
// the value of the variable stored in period
// does not correspond 1:1 to an actual value in seconds
volatile unsigned int period =0;

//following used for milestones but not final design
/*
int blueMin = 140;
int blueMax = 200;
int yellowMin = 0;
int yellowMax = 139;
int m3turncount = 0;
int startColor[2] = {blueMin,blueMax};
int color[2] = {0,0};
```

```
*/

// min readings from color sensor when on black (see lab 4)
// takes into account our scaler from timer to period
int blackMin = 60000;


// updates period when getColor() is called
ISR(PCINT2_vect){
  //resets the timer to zero on a rising edge
  if(PIND &= 0b00000100){
    TCNT1=0;
  }
  //stores the timer value on a falling edge
  else{
    timer = TCNT1;
    //scale timer value to get period
    period = timer/(16*10^6);
  }
}

void initColor(){
  // Initialize I/O pins
  DDRD=0b00001000;
  // Initialize pin change interrupt
  PCICR = 0b00000100;
  sei();
  // Initialize timers
  TCCR1A= 0b00000000;
  TCCR1B = 0b00000001;
}

int getColor(){
  // Enable the specific bit for pin change interrupt
  PCMSK2 = 0b00000100;
  // short delay
  _delay_ms(10);
  // Disable the specific bit for pin change interrupt
  // to prevent further interrupts until you call getColor again
  PCMSK2 = 0b00000000;
```

```
  _delay_ms(10);
  // Return the period
  return period;
}


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

  // call initColor function
  initColor();

  //get starting color
  getColor();

  // initialize motor outputs
  DDRB = 0b00001111;

  // start set initial path
  //drive forward 20in
  drive_forward(2500);

  //turn left 90 degrees
  turn_left(625);

  //forward 20in
  drive_forward(2500);

  //turn right 90 degrees
  turn_right(600);

  //drive forward 8in;
  drive_forward(1300);

  //turn right 115 degrees
  turn_right(650);

  //drive forward 35 in
  drive_forward(4300);
```

```
  //turn left 115 degrees
  turn_left(750);

  //drive forward 16 in
  drive_forward(1700);

  // turn all motor pins off (no movement)
  PORTB=0b00000000;

  // after initial path, only move if we get pushed onto black
  while(1){

    //get current color
    getColor();
    Serial.println(period); // for debugging
    // if on black, call seeBlack()
    if(period>blackMin){
      Serial.println("BLACK");  // for debugging
      seeBlack();
    }
    _delay_ms(50);
  }
}


int drive_forward(int t){ // function to drive forward
  PORTB |= 0b00001010;  // set pins 9 and 11 high
  PORTB &= 0b11111010;  // set pins 8 and 10 low
  myDelay(t);          // wait for t milliseconds
}
int drive_backward(int t){ // function to drive backward
  PORTB |= 0b00000101;  // set pins 8 and 10 high
  PORTB &= 0b11110101;  // set pins 9 and 11 low
  myDelay(t);        // wait for t milliseconds
}
int turn_left(int t){  // function to turn left
  PORTB |= 0b00001001;  // set pins 8 and 11 high
  PORTB &= 0b11111001;  // set pins 9 and 10 low
  myDelay(t);          // wait for t milliseconds
}
```

```
int turn_right(int t){   // function to right left
  PORTB |= 0b00000110;  // set pins 9 and 10 high
  PORTB &= 0b11110110;  // set pins 8 and 11 low
  myDelay(t);          // wait for t milliseconds
}
// our own delay function, _delay_ms() sometimes doesn't work
int myDelay(int t){
  for (int i=0; i<=t; i++){
    _delay_ms(1);
  }
}
// turn around if black border sensed
int seeBlack() {
  Serial.println("BLACK"); // for debugging
  drive_backward(100);  // reverse slightly to prevent driving off when
turning and potentially push other robot away
  turn_right(1100); // turn right 180 degrees
  drive_forward(1000); // drive away from black
  PORTB=0b00000000; // turn all motor pins off (no movement)
}
```