Group 34: Hunter Delor (htd32), Jordan Sandell (jms928), Mathias Kohler (mk2227)
Robot Project Final Report, Fall 2021

Project Poseidon

1. Robot Design and Strategy Overview

1.1. Mechanical design

Our robot features a custom chassis measuring an overall 8”x5” that fits all the components necessary for our design.

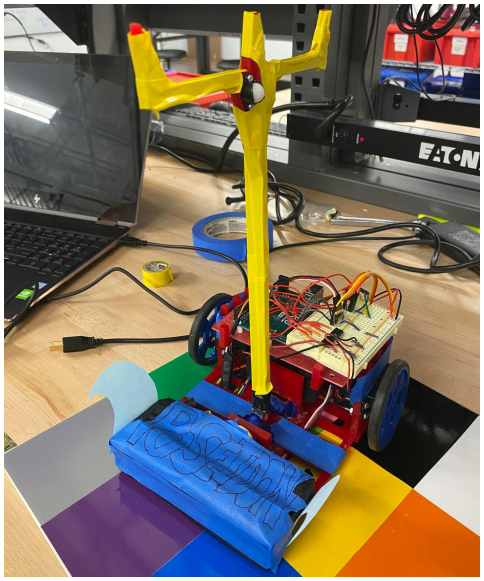

Figure 1: Photo of the robot poseidon showing its overall structure

In the rear, it features an elevated shelf to hold the arduino and breadboard with room to store the batteries and wheel motors underneath. The middle of the chassis includes holders for the two servos that hold the trident-shaped cube scooper and control its orientation. At the front of the chassis, there is a cantilevered section from which the QTI sensors hang and on which the color sensors sit. This cantilever is set 1 inch off the ground to allow the QTI sensors to hover an optimal distance from the ground such that ambient light will cause minimal disruption to their readings. To protect these

sensors from being hit by an opposing robot, our design also includes a shield in front of the cantilever.

1.2. Electrical design

The electrical design of the car is wired to be the most efficient in terms of spacing and coding. We used a lot of the provided sample code for our milestones so we decided it would be easier to code the robot using the same pins. Then with the remaining pins we organized them from back to front corresponding with the sensors and motors on the robot. For example, the sensors at the very front of the car were the two QTI sensors and the color sensor so we wired them into pins 5 and 6**.** Then we had the servo motors in the middle section of the robot, the servo was connected to pin 9**.** Lastly, we had the two H-bridges controlling the two wheel motors connected to pins 3 and 4 for the right wheel and pins 12 and 13 for the left wheel**.** That way we could wire the far away motors and sensors under the acrylic chassis and avoid them impeding the arm.

1.3. Software design

The software of our car is designed with the overall strategy of moving all of the cubes on the board onto our side. Our robot starts by checking the board's color to orient itself, and then it drives forward and deploys its trident so that it can reach the center of the grid where the cubes are quickly. After the trident is deployed, the robot continues to drive forward until it senses a color change on the board (indicating it has reached the opponent's side). When it senses this color change, it turns 90 degrees counterclockwise and drives forward, effectively scooping the cubes on the opponent's side and pushing ours. Once our robot senses that it has reached the black tape, it turns 90 degrees counterclockwise again to scoop the cubes it has collected onto our side. It then turns around by making a 270 degree turn clockwise, switches the orientation of the trident, and drives back to the other side of the grid until it senses the black tape again. As it does this, it collects the rest of the cubes so that once it reaches the black tape and turns 90 degrees clockwise, it has moved all the cubes onto our side.

2. Design Process Reflection

We started designing our robot with a brainstorming session where we all met and discussed various ideas about our robot's strategy and design. After this session, we merged our strongest ideas to create one cohesive design. We met again to map out our design in more detail, flush out any flaws we could identify, and order our parts once we were satisfied with our plan. We completed this design process prior to the milestones being due, so that we would be able to evaluate our design as we worked through the milestones. The first couple roadblocks we ran into were with our electrical components (we had one dead battery and one broken H-bridge), so those didn't indicate any major flaws in our design. We completed the second and third milestone without any major roadblocks other than wiring mishaps and code debugging. The final test of our design came during the fourth milestone, because by this point we had assembled our robot using all our ordered parts. The biggest challenge we ran into during this phase of the project was getting our positional servo operational. We had to replace the gears inside of the servo because during calibration the gears broke, and

then we had to change the wiring for everything to utilize a second timer as we ran into trouble trying to run the color sensor and servo on the same timer.

3. Competition Analysis

Our robot won 4 out of 6 matches in the round robin tournament to make it to the first round of the playoffs before it was defeated. In our first match, one wheel rotated and the robot turned off the board, which is not what we had observed the night before competition. Immediately following this, we analyzed what had changed about our robot between the night before and that morning, and we came to the realization that we uploaded the wrong code. As we weren't allowed to change the code anymore, we had to instead fix our wiring to create some sort of strategy out of the incorrect code. We decided to have our robot spin in circles and hope that the other robots would either drive themselves off the board, or accidentally push cubes onto our side. Our strategy proved effective as we were able to win our round robin bracket and move into the playoffs seeded 7/50. The flaw in this strategy was that since our robot was not doing anything proactive, as long as the other robot worked as it had planned to, we couldn't do anything to stop it. This is what led to our robot being defeated in the playoff bracket eventually. Overall, this points to the strength of our robot actually being its weakness as well. The strength, specifically, is that our robot wasn't doing anything, therefore it couldn't mess up and accidentally move cubes to the wrong side. The weakness coincidentally is that our robot wasn't doing anything, and as a result its success or failure depended entirely on what the opponent's robot did. This proved to not be the worst strategy as our robot did make it quite far, however the skill involved to get this far was obviously not really indicative of the mechanical, electrical, and software work that went into building our robot.

4. Conclusions

If we were to do this project again with all the same parameters, the big change we would make is uploading the correct code the morning of competition. When we ran our final code the night before competition, our robot did exactly what we described in the software design section of this report. After going through the competition and observing that many of the robots failed to move cubes to their own side even when they went essentially unopposed against our robot, we came to the conclusion that our robot probably would have done very well with our correct code. We are consequently extremely happy with the work that we did on this robot, and proud of ourselves for creating a robot that at least we know can do its intended job.

For future students working on this project, the biggest advice we can give is not to worry too much about specifics and to instead focus on making sure your own robot can do the task it was assigned. People spent so long trying to come up with the perfect unbeatable strategy for their robot, but in the end very few robots could even complete the task of moving more than half of the cubes to their side in a minute. Even though our robot did nothing but spin in a circle on our side during competition, we still emerged from the round robin tournament seeded 7/50 overall. If everyone had just focused on making sure their robot could move half of the cubes to their side, we wouldn't have won a single match with our broken robot.

5. Appendix

5.1. Appendix A: Bill of Materials

Part Name	Vendor/ Source	Part Number	Quantity	Price / Unit ($)	Subtotal ($)
Robot Wheel and Molded Tire	Digikey	149-28115-ND	2	3.99	7.98
180 degree positional servo	Individual Kit - amazon	SER0006	2	1.58	3.16
Acrylic	RPL	N/A	1	5.00	5.00
Laser cutting	RPL	N/A	12x12”	-	2.90
Ball caster with ⅜” plastic	Digikey	2183-950-ND	3	1.99	5.97
Total					$25.01

Figure 2: Table of bill of materials that includes all materials ordered by our lab group. Note the calculation to determine laser cutting price: *$0.50+$0.05(12”x4) = $4.10.*

5.2. Appendix B: Circuit Diagram

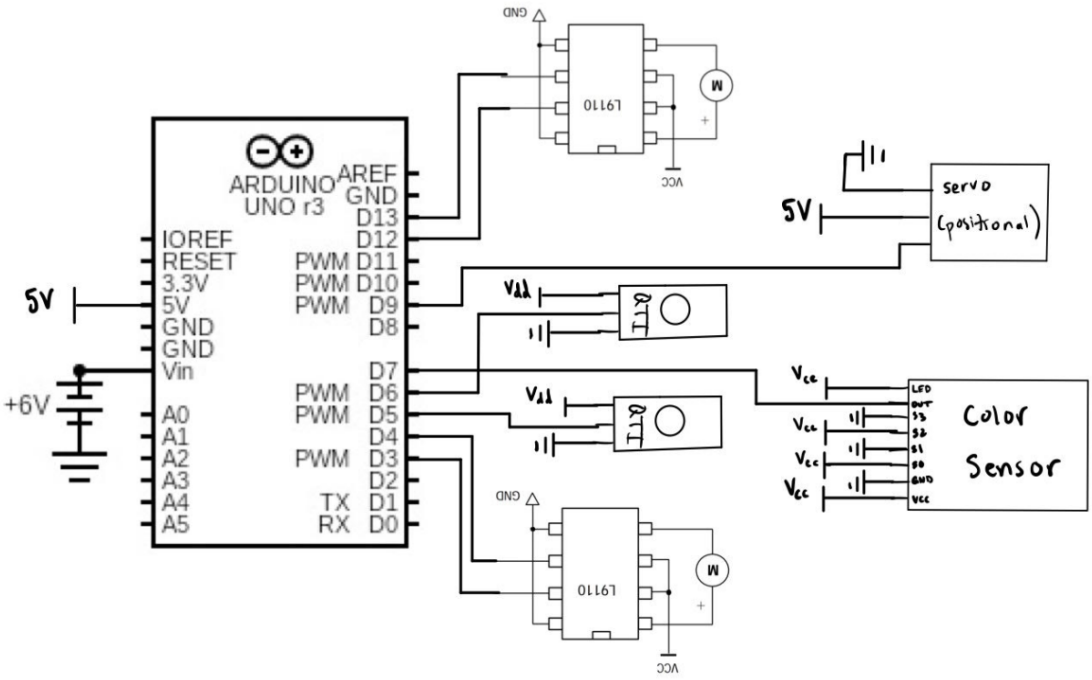

Figure 3: Circuit diagram of electrical components of Poseidon robot.

5.3. Appendix C: CAD files and drawings

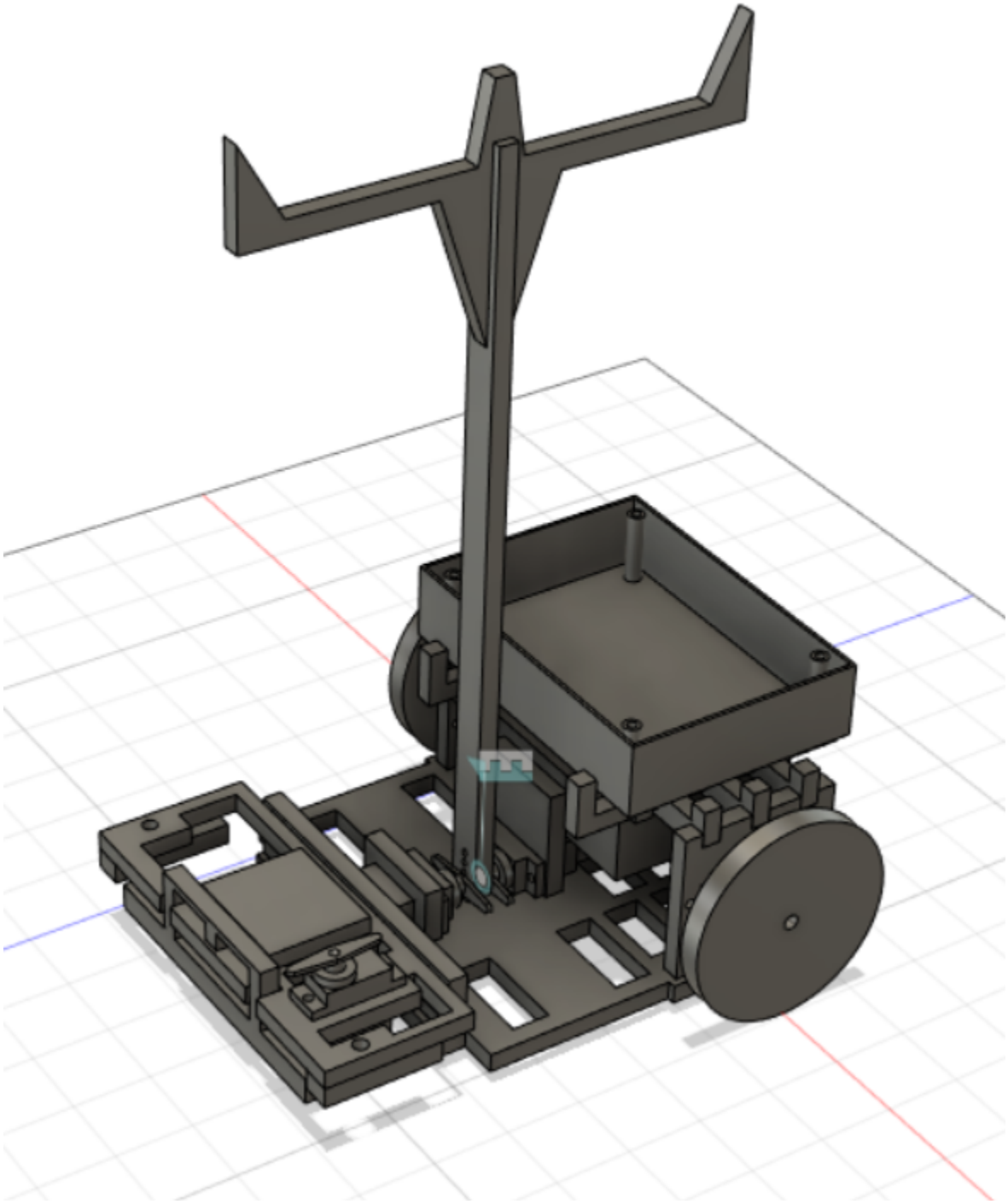

Figure 4: Full 3d model assembly of our robot on Fusion 360.

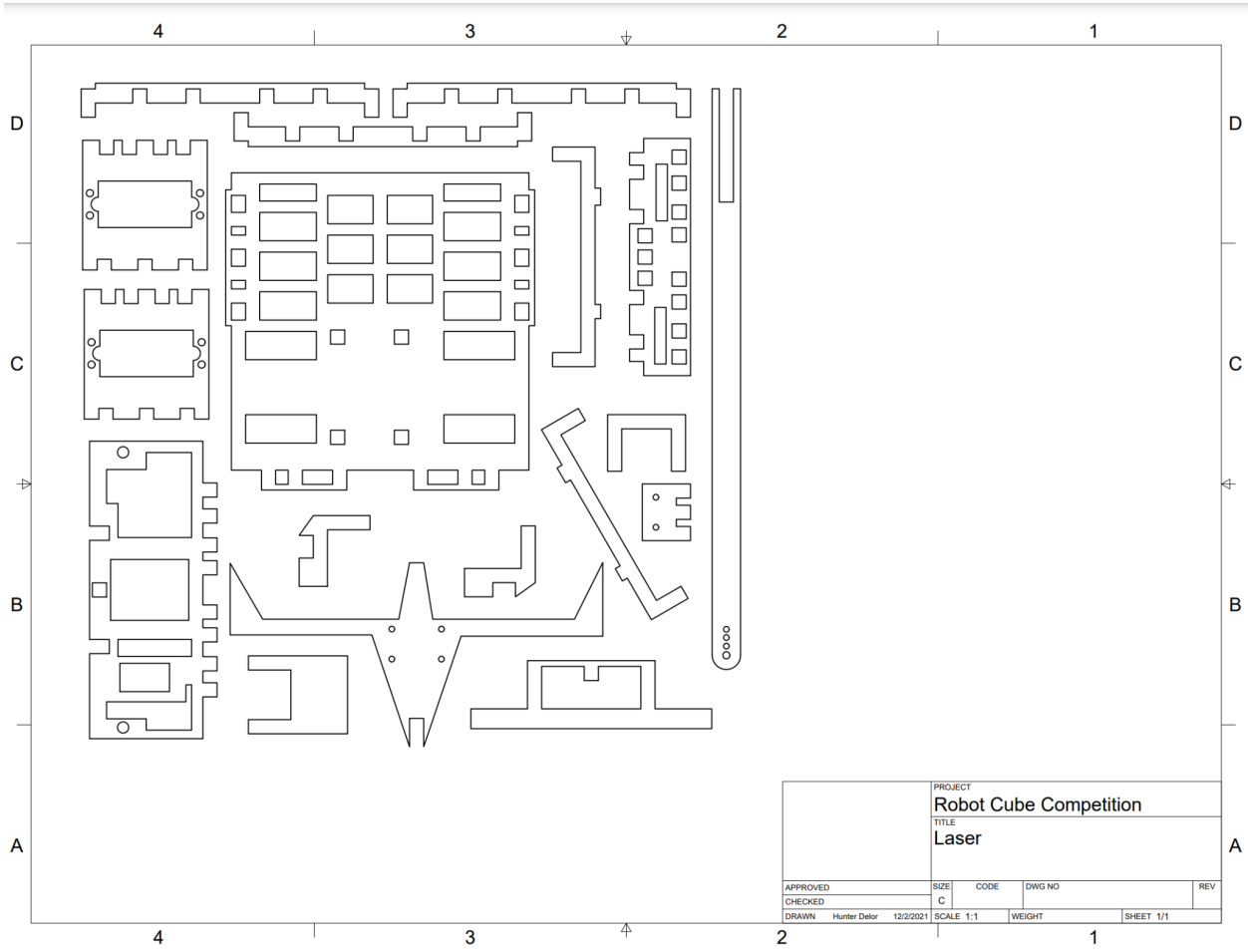

Figure 5: Drawing file of all the acrylic parts that make up the robot; laser cut by Cornell's Rapid Prototyping lab

5.4. Appendix D: Flowchart

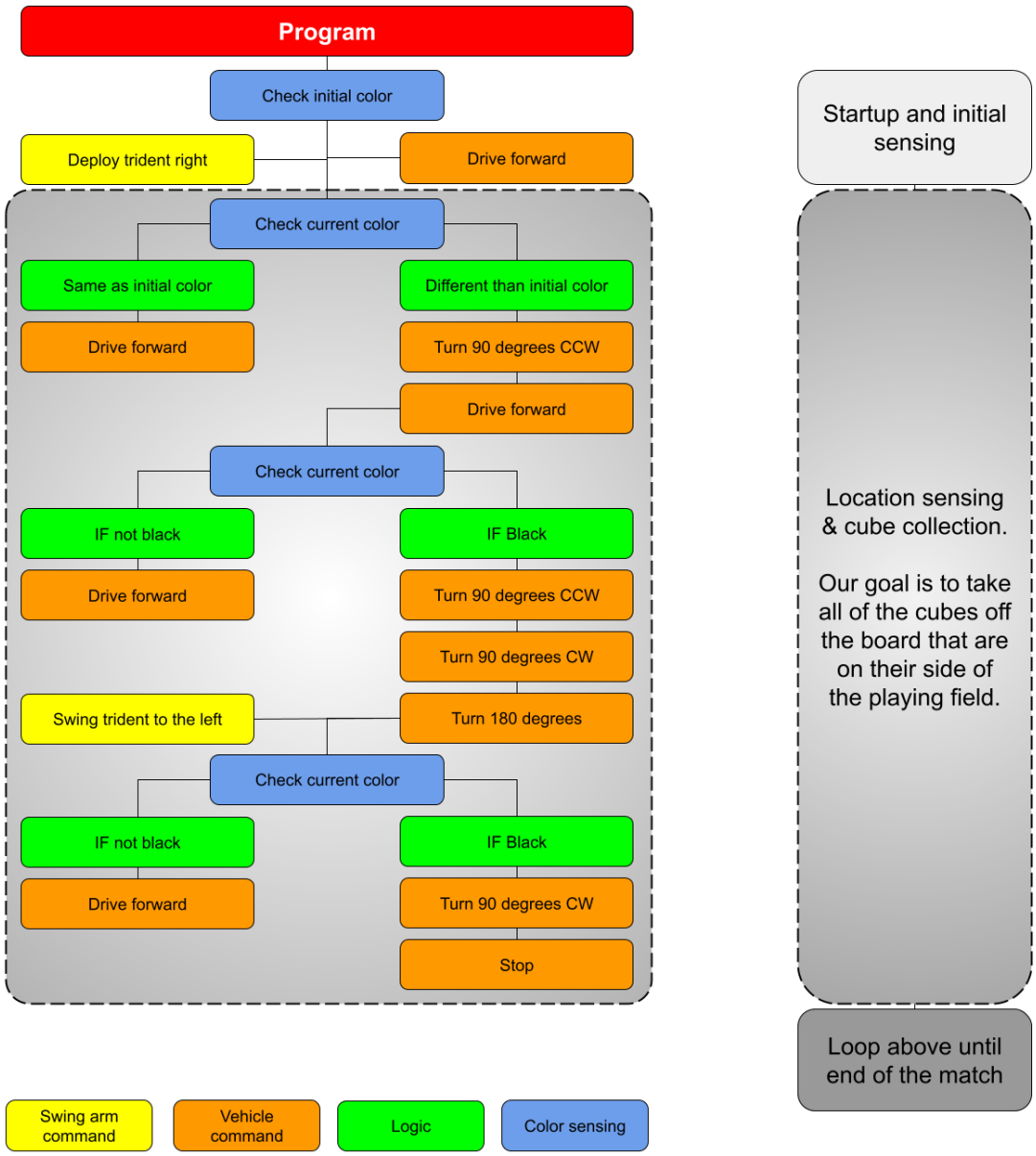

Figure 6: Flowchart showing the strategy of our robot.

5.5. Appendix E: Code

```
/*
  Right QTI connects to pin 5
  Left QTI connects to pin 6
```

```
    Color sensor connects to pin 7
    Arm Servo connects to 9

*  Color sensor variables set up
*/

volatile int period     ;
volatile int timer1     ;
volatile int S_Color    ;
volatile int Cur_Color  ;

/*
 *  Interrupt setup - interrupt vector for PCINT2
 */
ISR(PCINT2_vect)
{
  if (PIND & 0b10000000)
  { //if pin 7 (PD7) is high
    TCNT1=0           ; //reset timer
  }
  else
  {
    timer1=TCNT1      ; //store timer value in variable timer1
  }
}

void initColor(){
  //set up I/O for sensor

  //set up pin change interrupt on pin 7 (PCINT23)
  PCICR=0b00000100    ; //enable PCINT2 (datasheet pg 92)
  sei()               ; //enable all interrupts

  //set up timer 1
  TCCR1A=0b00000000   ; //set timer to normal mode (datasheet pg 171)
  TCCR1B=0b00000001   ; //set prescaler to 1 (datasheet pg 173)
}

/*
* Gets the period from the color sensor by triggering the interrupt for 5ms
*/

void getColor(){
  PCMSK2|=0b10000000; //enable PCINT23 (datasheet pg 94)
  _delay_ms(5); //delay 5 ms to give interrupt time to trigger
  PCMSK2&=0b01111111; //disable PCINT23 (datasheet pg 94)
```

```
  period=timer1*.0625*2; //convert ticks to microseconds
}

/*
 * Color sensor finished setup
 */

/*
 * Setup motors
 */
void setup() {

      // Set up the pins for the motors
        DDRD = 0b00011000   ;   // set pin 3 & 4 as an output
        DDRB = 0b00111100    ;   // set pin 10, 11, 12 & 13 as an output
  /*
   * Grab initial color
   */

   initColor()      ; //call initColor() to initialize all variables
   getColor()       ; //Get color at start
   S_Color = period ;

  adjust_stop()     ;
  _delay_ms(100)    ;//Need to have a delay after adjust stop=
  drive_forward()   ;
  _delay_ms(100)    ;
  swing90()         ;
  swing_right()     ;

}




/*
 * Competition Code start
 */


void loop() {

  initColor()         ;   //call initColor() to initialize all variables
  getColor()          ;
  Cur_Color = period  ;
```

```
DDRD = 0b00011000   ;   // set pin 3 & 4 as an output
DDRB = 0b00111100    ;   // set pin 10, 11, 12 & 13 as an output



if((Cur_Color > (S_Color + 12)) || (Cur_Color < (S_Color - 12)))
{
turn_left()        ;

/*
 * Drive forwards until it is at the edge
 */

while(!(Cur_Color>100))
{

initColor()        ;   //call initColor() to initialize all variables
getColor()         ;
Cur_Color = period  ;

DDRD = 0b00011000   ;   // set pin 3 & 4 as an output
DDRB = 0b00111100    ;   // set pin 10, 11, 12 & 13 as an output

  drive_forward()             ;
  _delay_ms(50)               ;
}

if((Cur_Color>100))
{
  turn_left()           ;
  _delay_ms(50)    ;
  turn_right()          ;
  _delay_ms(50)    ;
  turn_around()      ;
  _delay_ms(50)    ;
  drive_forward()   ;
  swing90()           ;
  swing_left()        ;

initColor()             ;   //call initColor() to initialize all variables
getColor()              ;
Cur_Color = period;

DDRD = 0b00011000   ;   // set pin 3 & 4 as an output
```

```
  DDRB = 0b00111100    ;   // set pin 10, 11, 12 & 13 as an output4

    while(!(Cur_Color>100))
      {

  initColor()         ;   //call initColor() to initialize all variables
  getColor()          ;
  Cur_Color = period  ;

  DDRD = 0b00011000   ;   // set pin 3 & 4 as an output
  DDRB = 0b00111100    ;   // set pin 10, 11, 12 & 13 as an output

        drive_forward()   ;
        _delay_ms(50)     ;
      }
    turn_right();
    adjust_stop();
  }

  }

  else
  {
    drive_forward()  ;
    _delay_ms(100) ;
  }


}

/*
 * Competition Code is done!
 */

/*
 * Driving functions
 */
int drive_forward(void){
  // spin both wheels forward

  PORTB = PORTB | 0b00100100  ;//
  PORTB = PORTB & 0b11100111  ;//
}
```

```
int turn_around(void){
 // Turn around 180 degrees

 PORTB = PORTB | 0b00010100 ;
 PORTB = PORTB & 0b11010111 ;

  _delay_ms(1400);
}

int turn_right(void){
 // Turn right 90 degrees

 PORTB = PORTB & 0b11010111 ;
 PORTB = PORTB | 0b00010100 ;
  _delay_ms(700);
}

int turn_left(void){
 // Turn left 90 degrees

 PORTB = PORTB & 0b11101011 ;
 PORTB = PORTB | 0b00101000 ;

  _delay_ms(700);
}

int adjust_stop(void){
 // Stop motors
 PORTB = PORTB & 0b11000011 ;
}

/*
 * Swing arm commands
 */

int swing90(void)
{
  DDRD = 0b00001000;
  TCCR2A = 0b00100001;
  TCCR2B = 0b00001111;
  OCR2A = 156;

if(OCR2B<11)
  {
  OCR2B = 3         ;
      while(OCR2B<11){
```

```
        OCR2B++       ;  //
        _delay_ms(10) ;
      }
  }
else if(OCR2B>11)
  {
  OCR2B = 21        ;
  while(OCR2B>11)
    {
    OCR2B--           ;  //
    _delay_ms(10)     ;
    }
  }
}

int swing_left(void)
{
  DDRD = 0b00001000;
  TCCR2A = 0b00100001;
  TCCR2B = 0b00001111;
  OCR2A = 156        ;
  OCR2B = 11         ;

    while(OCR2B<14 && OCR2B>3){
      OCR2B--       ;  //
      _delay_ms(10)  ;
    }
}

int swing_right(void)
{
  DDRD = 0b00001000;
  TCCR2A = 0b00100001;
  TCCR2B = 0b00001111;
  OCR2A = 156        ;
  OCR2B = 11          ;

    while(OCR2B>8 && OCR2B<21){
      OCR2B++       ;  //
      _delay_ms(10)  ;
    }
}
```