Automated Driving Using External Perception

Individual Lab Report - ILR01

February 9, 2023

Team E - Outersense

Author:

Jash Shah

Team Members:

Atharv Pulapaka
Dhanesh Pamnani
Ronit Hire
Shreyas Jha

Carnegie Mellon University

Contents

1 Individual Progress

1.1 Sensors and motor control lab

As for the sensors and motor control lab, my responsibility was to write code for the DC motor's position and speed control. Additionally, I was also responsible for the interfacing of the ultrasonic sensor with the Arduino Mega.

For the DC motor with encoder, I made the hardware connections and once this was complete, the number of encoder ticks per revolution had to be found. To do this, I manually rotated the motor shaft for one revolution while receiving feedback from the encoder. Our experiments showed that our encoder produced 100 ticks per revolution.

Next, I began writing the Arduino program to control the position and speed of the DC motor. Starting with velocity control, I first attached hardware interrupts on the encoder pins because every time there was a RISING edge, the interrupt service routine was called, which changed based on which encoder was leading. Once I was able to read the encoder count and convert it into an RPM (by counting how many ticks the encoder was providing every minute), I was able to use simple PID equations to control the speed of the motor. I then went on to calculate the position from the encoder count by multiplying the encoder count by 100 (tick count) and dividing by 360 (number of degrees). This gave me the current position of the encoder and again, I was able to use this feedback to control the position of the DC motor.

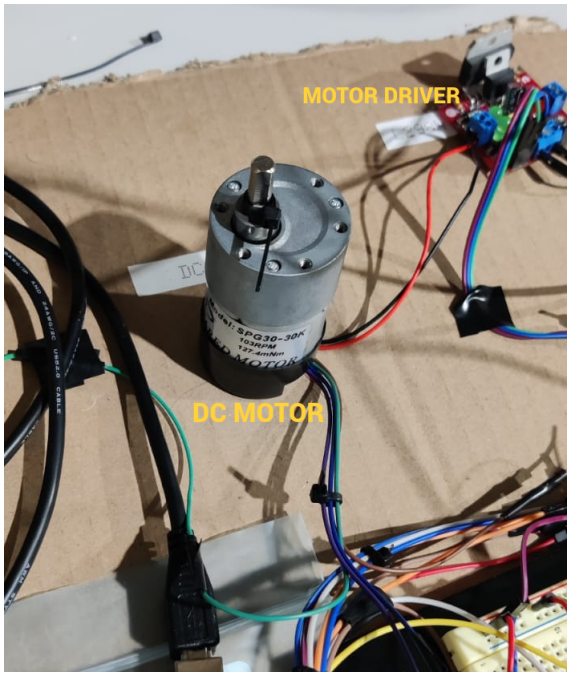

Figure 1: Motor and L298 driver

1.2 MRSD Project - OuterSense

As for the MRSD project, my main focus thus far has been in the perception domain. Vision is a core facet of our project because the car and lane need to be detected from a bird's eye view accurately with minimal lag. My responsibility as of now is to detect a non-fiducial marker on the car. For this, I have to implement three levels of filtering; shape, color, and area. I included trackbars in my code to change these thresholds in real-time. These parameters need to be tuned

to achieve the best performance. Next, I need to find the angle between the marker on the car and a fixed fiducial marker on the track with respect to the camera. This is extremely important again because I can use this angle to determine the yaw of the car with respect to the track and send a steering angle to the controller. Moreover, the camera can have slight vibrations, movements, and disturbances. To eliminate this threat, I do not want to hard code the transforms between the fixed fiducial marker on the track and the camera frame.

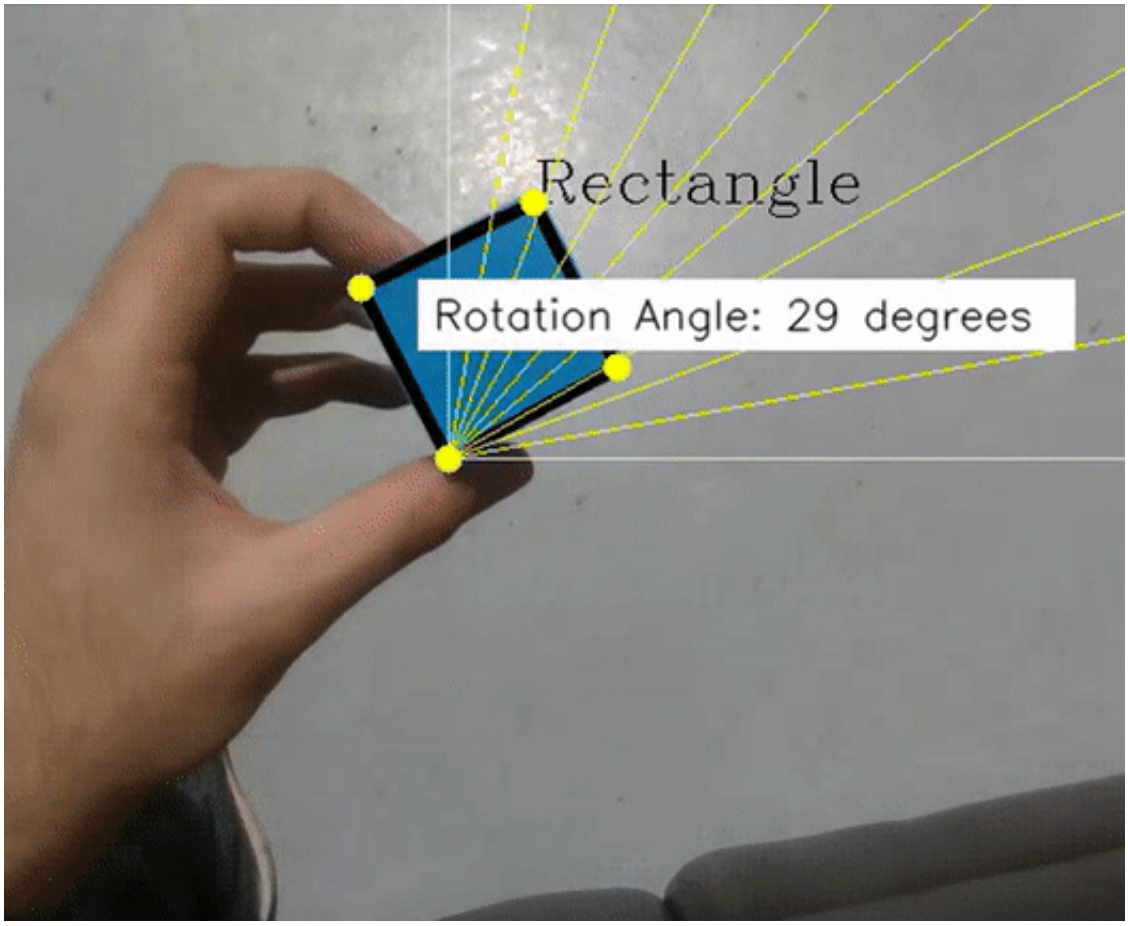

Figure 2: Table testing of marker yaw angle

Currently, I am working on using the corners detected in the above shapes to further threshold the shapes better because the car will be in motion and more so, at some point, the car will be at an obscure camera angle which might make it difficult to detect the marker. I am currently fine-tuning more parameters that will overcome this issue. Parallely, I am also working on detecting and tracking ArUco markers which will aid me in finding the angle between the marker on the car and the fixed marker on the track. Once I find the angle deviation of the center of the car with the track, I can use it to calculate yaw and send the steering command to the high-level controller.

2 Challenges

2.1 Sensors and Motors Lab

A problem I faced during the process of implementing position and speed control was during the integration of these two codes. It was difficult to combine these two individually functional codes because, for both, the interrupts were being called on the same pins. Hence, the code would not know whether the function call was for position control or speed control. To solve this issue, I used a feature called detach interrupt. Because speed control required both interrupts whereas position control could be achieved using a single RISING interrupt, I was able to detach the second interrupt during the position control function call. Also, in the ISR, I placed a flag to check which control was being called by the loop, position, or control. Using this, I was able to solve this issue effectively and make sure that the same code could be pushed for both position and speed control. This would make the high-level integration with the GUI much easier.

Another hurdle I faced was with respect to the hardware of the DC motor and its driver. During some tests, the same code would run on the same hardware, whereas, at other times, it would behave erratically. It took the team a while to debug this issue and it turned out that the DC motor driver, L298N was faulty.

2.2 MRSD project - OuterSense

During the MRSD project, the main issue I am facing currently is that of robustly detecting objects in cluttered scenes. If there are multiple objects in scenes with the car, the algorithm gets several garbage readings. The values for color, as well as shape detection, need to be tuned much more in order to eliminate this threat.

Moreover, another challenge I am currently facing is the detection of objects when they are at obscure angles. For instance, when the marker is used, if the marker is on the edge of the camera frame, it looks like a parallelogram rather than a rectangle which causes the algorithm to fail at some edge cases. This needs to be fixed and is a major challenge being faced right now.

3 Teamwork

- **Ronit Hire**: Ronit was responsible for the web-based GUI that he made on Flask. Moreover, he also helped me integrate the position and velocity control codes. Regarding the MRSD project, Ronit is mainly been involved in high-level systems architecture. Moreover, he is also looking after the logistics of the team including the budget, parts delivery, and documentation.

- **Shreyas Jha**: As for the sensors and motor control lab, Shreyas worked on the stepper motor and potentiometer. He interfaced these two components and also worked on integrating all the code for the lab. As for the project, Shreyas is currently working on modifications to the RC car.

- **Dhanesh Pamnani**: Dhanesh implemented the code for the push button and debouncing. Moreover, he also implemented the program to read data from the IR sensor, along with the computation of its transfer function. As for the MRSD project, Dhanesh is actively working with Tim on the hardware aspects to build the track and infrastructure units which are critical to our project's success.

- **Atharv Pulapaka**: As for the sensors and motor control lab, Atharv was involved in interfacing the servo motor with the Arduino Mega. Further, he was also actively involved in building the physical circuit and debugging hardware issues. As for the project, Atharv is currently attempting to implement Model Predictive Control for the RC car and is focusing on trying out different algorithms for lane keeping.

4 Future work

4.1 Sensors and motor control lab

As for the sensors and motor control lab, the only useful outcome that can be used is the interfacing of the IMU and servo with the Arduino. We can use this because we require a low-level controller on the car which implements closed-loop control of velocity and steering angle.

4.2 MRSD Project - OuterSense

Regarding the capstone project, my personal goal is to overcome the hurdles described in Section 2.2. I need to make sure that the shape is detected in all poses in the camera frame. It is critical that my algorithm achieves this. Moreover, I must make sure that my values are tuned such that the garbage values I receive in a cluttered environment are minimal.

From the team's perspective, we are looking to start testing on the track as soon as possible so that we can ascertain parameters that can only be found experimentally, such as the slip the vehicle is experiencing or the amount the camera vibrates when mounted at a height. It is essential that we take note of these factors because they can gravely affect the performance of our system.

5 Sensor and motor controls lab quiz

1. From ADXL335 accelerometer datasheet

   (a) What is the sensor's range?
   A. The range is between -3.6 to +3.6 g

   (b) What is the sensor's dynamic range?
   A. The dynamic range is 7.2g

   (c) What is the purpose of the capacitor CDC on the LHS of the functional block diagram on p. 1? How does it achieve this?
   A. The capacitor is used to prevent noise from the power supply. It can do this because capacitors' terminals do not change voltage with sudden changes in supply voltage.

   (d) Write an equation for the sensor's transfer function.
   A. $Vout = 1.5 + 0.3a$

   (e) What is the largest expected non-linearity error in g?
   A. The typical non-linearity is 0.3% of full scale reading, $0.3\% * 7.2 = 0.0216g$

   (f) What is the sensor's bandwidth for the X- and Y-axes?
   A. The bandwidth is 1600Hz for both directions.

   (g) How much noise do you expect in the X- and Y-axis sensor signals when your measurement bandwidth is 25 Hz?
   A. The noise is given by $rmsNoise = NoiseDensity * \sqrt{1.6 * BW}$
   This equates to 948.68 μg

   (h) If you didn't have the datasheet, how would you determine the RMS noise experimentally? State any assumptions and list the steps you would take.
   A. To determine the RMS noise experimentally, first we would have to place the sensor on a flat surface assuming there is no offset in the X or Y direction. Then, we would have to take readings at regular intervals to avoid aliasing. Next, we would have to calculate the mean, subtract the mean from each reading, square the residual signal, average this value and then finally take the square root of the average power in the noise. This is the RMS value of the noise.

2. Signal Conditioning

   - Filtering
   1. Name at least two problems you might have in using a moving average filter.
   A.
   i) One of the main problems with the moving average filter is its slow response time. This is because the filter takes into account the average of a large number of previous samples, which means it takes a long time to respond to changes in the input signal. This can lead to a lag in the output signal compared to the input signal, which can be problematic in real-time applications.
   ii) Another problem with the moving average filter is its sensitivity to outliers. This is because the filter takes into account all the previous samples, which means that a single outlier can have a significant impact on the output signal. This can cause the filter to produce incorrect results and can lead to inaccurate data analysis.

   2. Name at least two problems you might have in using a median filter.
   A.

i) One of the main problems with median filters is their computational complexity. This is because the filter needs to sort all the samples in the window in order to calculate the median value, which can be a time-consuming and computationally expensive process, especially for large windows.
ii) Another problem with median filters is their sensitivity to impulse noise. This is because median filters are not as effective as other filters, such as averaging filters, in removing impulse noise. This can lead to a degradation of the quality of the filtered signal and can result in inaccurate data analysis.

- Opamps
  In the following questions, you want to calibrate a linear sensor using the circuit in Fig. 1 so that its output range is 0 to 5V. Identify in each case: 1) which of V1 and V2 will be the input voltage and which is the reference voltage; 2) the values of the ratio Rf/Ri and the reference voltage. If the calibration can't be done with this circuit, explain why.

  (i) Your uncalibrated sensor has a range of -1.5 to 1.0V (-1.5V should give a 0V output and 1.0V should give a 5V output).
  A. (Case 1)
  Using the opamp equation: $\frac{V_1-V_2}{R_i} = \frac{V_2-V_{out}}{R_f}$
  $V_1 = V_{in}$ and $V_2 = V_{ref}$
  $\therefore \frac{R_f}{R_i} = -2000\Omega$
  This is not possible, so we can use $V_1 = V_{ref}$ and $V_2 = V_{in}$
  $\therefore \frac{R_f}{R_i} = 1000\Omega$
  So, we can use V_1 as a reference, V_2 as input which can be used to calibration.

  (ii) Your uncalibrated sensor has a range of -2.5 to 2.5V
  A. (Case 2) Given the opamp equation: $\frac{V_1-V_2}{R_i} = \frac{V_2-V_{out}}{R_f}$
  $V_1 = V_{in}$ and $V_2 = V_{out}$
  $\therefore \frac{R_f}{R_i} = -2000\Omega$
  This is not possible so we use $V_1 = V_{out}$ and $V_2 = V_{in}$
  $\therefore \frac{R_f}{R_i} = 0$
  So R_f has to be 0 or R_i has to be infinite. This is not possible so we cannot calibrate in this case.

3. Controls

   (a) If you want to control a DC motor to go to a desired position, describe how to form a digital input for each of the PID (Proportional, Integral, Derivative) terms.
   A. To form the digital input for the integral term, you can calculate the integral of the error over time using a numerical integration method and multiply it by a constant integral gain (Ki). The resulting value will be the integral input. To form the digital input for the derivative term, you can calculate the derivative of the error using a numerical differentiation method and multiply it by a constant derivative gain (Kd). The resulting value will be the derivative input.

   (b) If the system you want to control is sluggish, which PID term(s) will you use and why?

A. The proportional term generates an output that is proportional to the error between the desired position and the actual position. Increasing the proportional gain (Kp) will increase the response of the controller to the error, causing it to generate a stronger control signal that drives the system to the desired position more quickly.

(c) After applying the control in the previous question, if the system still has a significant steady-state error, which PID term(s) will you use and why?
A. The integral term helps to eliminate the steady-state error by continuously driving the system toward the desired position. The integral term accumulates the error over time and generates an output proportional to the accumulated error. By increasing the integral gain (Ki), the integral term will generate a stronger control signal that will correct for any steady-state error in the system.

(d) After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?
A. The derivative term generates an output proportional to the rate of change of the error. By increasing the derivative gain (Kd), the derivative term will generate a stronger control signal that will dampen the overshoot and stabilize the system.

6 Code for DC motor position and speed control

```
1
2  //FLAG
3  char flag = 'p';
4
5  // SPEED DEFS
6  float kp = 0.001;
7  float ki = 0.0005;
8  float kd = 0;
9  //float kp = 0.1;
10 //float ki = 0.005;
11 //float kd = 0;
12 unsigned long t;
13 unsigned long t_prev = 0;
14 const byte interruptPinA = 2;
15 const byte interruptPinB = 3;
16 //volatile long EncoderCount = 0;
17 long EncoderCount = 0;
18 const byte PWMPin = 4;
19 const byte DirPin1 = 5;
20 const byte DirPin2 = 12;
21 volatile unsigned long count = 0;
22 unsigned long count_prev = 0;
23 float Theta, RPM, RPM_d;
24 float Theta_prev = 0;
25 int dt;
26 float RPM_max = 230;
27 #define pi 3.1416
28 float Vmax = 11.7;
29 float Vmin = -11.7;
30 float V = 0.1;
31 float e, e_prev = 0, inte, inte_prev = 0;
32
33
34 //POSITION DEFS
35 #define ENCA 3
36 #define ENCB 2
37 #define PWM 4
38 #define IN2 5
39 #define IN1 12
40 int pos = 0;
41 long prevT = 0;
42 float eprev = 0;
43 float eintegral = 0;
44
45 //FUNCTIONS
46 void ISR_EncoderA() {
47
48
49   if (flag == 'v') {
50     bool PinB = digitalRead(interruptPinB);
51     bool PinA = digitalRead(interruptPinA);
52
53     if (PinB == LOW) {
54       if (PinA == HIGH) {
55         EncoderCount++;
```

```
56        }
57        else {
58          EncoderCount--;
59        }
60      }
61
62      else {
63        if (PinA == HIGH) {
64          EncoderCount--;
65        }
66        else {
67          EncoderCount++;
68        }
69      }
70    }
71
72    else {
73      int b = digitalRead(ENCB);
74      if (b > 0) {
75        pos++;
76      }
77      else {
78        pos--;
79      }
80    }
81  }
82
83  void ISR_EncoderB() {
84    bool PinB = digitalRead(interruptPinA);
85    bool PinA = digitalRead(interruptPinB);
86
87
88    if (flag == 'v') {
89
90      if (PinA == LOW) {
91        if (PinB == HIGH) {
92          EncoderCount--;
93        }
94        else {
95          EncoderCount++;
96        }
97      }
98
99      else {
100       if (PinB == HIGH) {
101         EncoderCount++;
102       }
103       else {
104         EncoderCount--;
105       }
106     }
107   }
108
109
110   //  else{
111   //    return ;
112   //  }
113 }
```

```
114
115
116 float sign(float x) {
117   if (x > 0) {
118     return 1;
119   } else if (x < 0) {
120     return -1;
121   } else {
122     return 0;
123   }
124 }
125
126 //***Motor Driver Functions*****
127
128 void WriteDriverVoltage(float V, float Vmax) {
129   int PWMval = int(255 * abs(V) / Vmax);
130   if (PWMval > 255) {
131     PWMval = 255;
132   }
133   if (V > 0) {
134     digitalWrite(DirPin1, HIGH);
135     digitalWrite(DirPin2, LOW);
136   }
137   else if (V < 0) {
138     digitalWrite(DirPin1, LOW);
139     digitalWrite(DirPin2, HIGH);
140   }
141   else {
142     digitalWrite(DirPin1, LOW);
143     digitalWrite(DirPin2, LOW);
144   }
145   analogWrite(PWMPin, PWMval);
146 }
147
148
149 int mode = 0 ;
150
151
152
153 void setup() {
154
155   if (flag == 'v') {
156
157     Serial.begin(9600);
158     pinMode(interruptPinA, INPUT_PULLUP);
159     pinMode(interruptPinB, INPUT_PULLUP);
160     //    attachInterrupt(digitalPinToInterrupt(interruptPinA),
      ISR_EncoderA, RISING);
161     //    attachInterrupt(digitalPinToInterrupt(interruptPinB),
      ISR_EncoderB, CHANGE);
162     pinMode(DirPin1, OUTPUT);
163     pinMode(DirPin2, OUTPUT);
164
165     cli();
166     TCCR1A = 0;
167     TCCR1B = 0;
168     TCNT1 = 0;
169     OCR1A = 12499; //Prescaler = 64
```

```
    TCCR1B |= (1 << WGM12);
    TCCR1B |= (1 << CS11 | 1 << CS10);
    TIMSK1 |= (1 << OCIE1A);
    sei();
  }



  else {
    Serial.begin(9600);
    pinMode(ENCA, INPUT);
    pinMode(ENCB, INPUT);
//    attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);
  }
}

int switch_flag = 0;
bool first_time = true;

void loop() {

  if (Serial.available() > 0) {
    delay(500);
    flag = Serial.read();
    switch_flag = 1;
    delay(500);
    Serial.println(flag);
  }



  if (flag == 'v') {


        if(switch_flag==1)
        {
          detachInterrupt(digitalPinToInterrupt(ENCA));
          delay(500);
          attachInterrupt(digitalPinToInterrupt(interruptPinA),
    ISR_EncoderA, RISING);
          attachInterrupt(digitalPinToInterrupt(interruptPinB),
    ISR_EncoderB, CHANGE);
          delay(2000);
          EncoderCount = 0;
          Theta_prev =0;
          t_prev = 0;
          count = 0;
          count_prev = 0;

//          delay(2000);
          switch_flag=0;
        }
    if (first_time)
    {
          attachInterrupt(digitalPinToInterrupt(interruptPinA),
    ISR_EncoderA, RISING);
```

```
224           attachInterrupt(digitalPinToInterrupt(interruptPinB),
      ISR_EncoderB, CHANGE);
225           first_time = false;
226     }
227 //    Serial.println("V control");
228
229
230     if (count > count_prev) {
231       t = millis();
232       Theta = EncoderCount / 900.0;
233       dt = (t - t_prev);
234       //    RPM_d = RPM_max * (sin(2 * pi * 0.005 * t / 1000.0))
235       //        * sign(sin(2 * pi * 0.05 * t / 1000.0));
236
237       RPM_d = 20;
238       //    RPM_d = 150;
239       if (t / 1000.0 > 100) {
240         RPM_d = 0;
241       }
242       RPM = (Theta - Theta_prev) / (dt / 1000.0) * 60;
243       e = RPM_d - RPM;
244 //      Serial.println(EncoderCount);
245       inte = inte_prev + (dt * (e + e_prev) / 2);
246       V = kp * e + ki * inte + (kd * (e - e_prev) / dt) ;
247       if (V > Vmax) {
248         V = Vmax;
249         inte = inte_prev;
250       }
251       if (V < Vmin) {
252         V = Vmin;
253         inte = inte_prev;
254       }
255
256
257       WriteDriverVoltage(V, Vmax);
258
259 //      Serial.print(RPM_d); Serial.print(" \t");
260 //      Serial.print(RPM); Serial.print(" \t ");
261       //    Serial.print(V); Serial.print("\t  ");
262       //    Serial.print(e); Serial.println("  ");
263 //      Serial.println(" ");
264
265       Theta_prev = Theta;
266       count_prev = count;
267       t_prev = t;
268       inte_prev = inte;
269       e_prev = e;
270     }
271
272   }
273
274   else if(flag=='p') {
275
276
277     if (switch_flag == 1)
278     { pinMode(ENCA, INPUT);
279       pinMode(ENCB, INPUT);
280       detachInterrupt(digitalPinToInterrupt(interruptPinA));
```

```
281       detachInterrupt(digitalPinToInterrupt(interruptPinB));
282       delay(500);
283       attachInterrupt(digitalPinToInterrupt(ENCA), readEncoder, RISING);
284
285       setMotor(-1, 0, PWM, IN1, IN2);
286       delay(2000);
287       switch_flag = 0;
288       pos = 0;
289     }
290     if (first_time)
291     {
292           attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING)
      ;
293           first_time = false;
294     }
295
296     int degree = 180;
297 //    Serial.println("P control");
298 //    Serial.println(pos);
299
300     // set target position
301     int target = degree * 100 / 360;
302     //    int target = 45;
303
304     //target = 250*sin(prevT/1e6);
305
306     // PID constants
307     float kp = 30;
308     float kd = 0.001;
309     float ki = 0.001;
310
311     // time difference
312     long currT = micros();
313     float deltaT = ((float) (currT - prevT)) / ( 1.0e6 );
314     prevT = currT;
315
316     // error
317     int e = pos - target;
318
319     // derivative
320     float dedt = (e - eprev) / (deltaT);
321
322     // integral
323     eintegral = eintegral + e * deltaT;
324
325     // control signal
326     float u = kp * e + kd * dedt + ki * eintegral;
327
328     // motor power
329     float pwr = fabs(u);
330     //  Serial.println(pwr);
331     if ( pwr > 255 ) {
332       pwr = 255;
333     }
334     //  if (pwr <200) {
335     //    pwr = 200;
336     //  }
337
```

```
338     // motor direction
339     int dir = 1;
340     if (u < 0) {
341       dir = -1;
342     }
343 
344     // signal the motor
345     setMotor(dir, pwr, PWM, IN1, IN2);
346 
347     // store previous error
348     eprev = e;
349 
350     //  Serial.print(target);
351     //  Serial.print(" ");
352     //  Serial.print(pos);
353     //  Serial.println();
354   }
355 }
356 
357 
358 ISR(TIMER1_COMPA_vect) {
359   count++;
360   //  Serial.print(count * 0.05); Serial.print(" \t");
361 }
362 
363 void setMotor(int dir, int pwmVal, int pwm, int in1, int in2) {
364   analogWrite(pwm, pwmVal);
365   if (dir == 1) {
366     digitalWrite(in1, HIGH);
367     digitalWrite(in2, LOW);
368   }
369   else if (dir == -1) {
370     digitalWrite(in1, LOW);
371     digitalWrite(in2, HIGH);
372   }
373   else {
374     digitalWrite(in1, LOW);
375     digitalWrite(in2, LOW);
376   }
377 }
378 
379 
380 void readEncoder() {
381 
382   int b = digitalRead(ENCB);
383   if (b > 0) {
384     pos++;
385   }
386   else {
387     pos--;
388   }
389 }
```