Experiment 2: Line Following Robot

Siddhant Batra (200070094)
Shivam Patel (200070077)
Harsh Lulla (200070024)

October 2022

1 Objective

The aim of this experiment was to create a PID controlled line following robot. The inputs to the robot are three sensors mounted on the bottom of the board. We can control the individual speed of the left and right wheels. We use a sample path the robot has to traverse within 30 seconds.

2 Control Algorithm

In the control algorithm we used PID controller implementation in discrete machine time (t 10us) where differential and integral are calculated at every step.

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{d}{dt}e(t)] \tag{1}$$

3 Design

The direction of movement of the motor is stored in pin PORTB, while the the pins PD4 and PD5 are for velocity of movement using PWM.
The three sensors on the base of the robot read the left, center and right values. When a sensor detects a black line under it, it outputs a high signal, otherwise outputs a low signal. We take analog inputs from these sensors and convert them into digital inputs(from 0 to 255)
We output on a LCD screen mounted on the topside. For this purpose, we initially

have to set all the LCD pins to output mode.

Setup code for timer circuit we used in this experiment:

```
void timer1_init(void)
{
 TCCR1B = 0x00; //stop
 TCNT1H = 0xFF; //setup
 TCNT1L = 0x01;
 OCR1AH = 0x00;
 OCR1AL = 0xFF;
 OCR1BH = 0x00;
 OCR1BL = 0xFF;
 ICR1H  = 0x00;
 ICR1L  = 0xFF;
 TCCR1A = 0xA1;
 TCCR1B = 0x0D; //start Timer
}
```

We define the error as

$$100 - \frac{(0 \cdot l + 100 \cdot c + 200 \cdot r)}{(l + c + r)}$$

We continously update the total and differential error at every time step by taking the readings from the sensors, then the PID control signal is sent to the control microchip which in turn controls the movement of motors.
A lot of trials were done to obtain the appropriate values of Kp, Kd and Ki which are as follows.

- $K_p = 43$
- $K_d = 0.78$
- $K_i = 0.06$

4 Challenges Faced

1. We didn't initially impose thresholds on the sensor input values. This gave erroneous results as even small input values caused the bot to turn in unexpected

directions. We experimented with various threshold values till the result was satisfactory.

2. At first, we didn't implement velocity control. This meant that the motor drive was at constant angular velocity. One of the undesirable consequences of this was that the bot would wander off at intersections. After implementing velocity control, the bot would speed up before approaching an intersection, and cross it with the inertia it had while entering it. Also this meant corners could be traversed more accurately.

3. While experimenting with one of the bots, the bot was unexpectedly turning which confused us whether the algorithm was incorrect. After continually trying to modify the code, we ran the bot with just the forward command and realized that one of the motors was jammed. We replaced the bot and continued.

5 Code Snippet

```
unsigned char ADC_Conversion(unsigned char);
unsigned char ADC_Value;
unsigned int l = 0;
unsigned int c = 0;
unsigned int r = 0;
unsigned char PortBRestore = 0;

const unsigned maxspeeda = 150;
const unsigned maxspeedb = 150;
const unsigned basespeeda = 100;
const unsigned basespeedb = 100;

int error=0;
int lastError = 0;
unsigned int position;

int P;
int I;
int D;
```

```
float Kp = 0;
float Ki = 0;
float Kd = 0;

unsigned max_l = 200;
unsigned max_r = 200;
unsigned max_c = 200;
unsigned min_l = 0;
unsigned min_r = 0;
unsigned min_c = 0;

//Function used for setting motor's direction
void motion_set (unsigned char Direction)
{
 unsigned char PortBRestore = 0;

 Direction &= 0x0F;
 PortBRestore = PORTB;
 PortBRestore &= 0xF0;
 PortBRestore |= Direction;
PORTB = PortBRestore;
}

void velocity (unsigned char left_motor, unsigned char right_motor) {
        OCR1AL = left_motor;
         OCR1BL = right_motor;
        }

void timer1_init(void)
{
 TCCR1B = 0x00; //stop
 TCNT1H = 0xFF; //setup
 TCNT1L = 0x01;
 OCR1AH = 0x00;
 OCR1AL = 0xFF;
 OCR1BH = 0x00;
 OCR1BL = 0xFF;
```

```
 ICR1H  = 0x00;
 ICR1L  = 0xFF;
 TCCR1A = 0xA1;
 TCCR1B = 0x0D; //start Timer
}

//Main Function
int main(void)
{
    init_devices();
    port_init();
    lcd_set_4bit();
    lcd_init();

    max_l = 255;
    max_r = 255;
    max_c = 255;
    min_l = 0;
    min_r = 0;
    min_c = 0;

    maxspeed = 150;

    basespeed = 100;

    Kp = 43;
    Ki = 0.06;
    Kd = 0.78;

    while(1)
    {
        l=ADC_Conversion(3);
        c=ADC_Conversion(4);
        r=ADC_Conversion(5);
        lcd_print(1, 1, l, 3);
        lcd_print(1, 5, c, 3);
        lcd_print(1, 9, r, 3);
        l = (l-min_l)/(max_l - min_l);
```

```
    l = l*100;
    r = (r-min_r)/(max_r - min_r);
    r = r*100;
    c = (c-min_c)/(max_c - min_c);
    c = c*100;
    position = (l*0 + c*100 + r*200)/(l+c+r);
    error = 100 - position;
    P = error;
    I = I + error;
    D = error - lastError;
    lastError = error;
    int motorspeed = P*Kp + I*Ki + D*Kd;

    int control_signal_left = basespeed + motorspeed;
    int control_signal_right = basespeed - motorspeed;

    if (control_signal_left > maxspeed) {
      control_signal_left = maxspeed;
    }
    if (control_signal_right > maxspeed) {
      control_signal_right = maxspeed;
    }
    if (control_signal_left < 0) {
      control_signal_left = 0;
    }
    if (control_signal_right < 0) {
      control_signal_right = 0;
    }
    velocity(control_signal_left, control_signal_right);
}
```

6 Results

The following are the experimental results.

- Kp = 43
- Kd = 0.78

- Ki = 0.06
- T_{LAP} = 29.5s

The effect of changing the constants is -

- Changing Kp - If Kp is changed, the speed of the bot changes. Increasing the Kp increases the speed of the bot and vice versa. This is attributed to the profounded correcting action of the bot PID controller
- Changing Kd - A change in Kd changes the corrective action based on the previous error. That is, if the Kd is large, then the bot takes sharper turns at edges, as the differential error becomes non zero at turns
- Changing Ki - Changing Ki does not have a profound impact on the bot, as the difference of right and left motor turns is almost zero for most of the times, when the bot is moving in a straight line