MEEG 667: Digital Control Project Report

Vaughn Chambers, Chrysostomos Karakasis, Sumanth Nakka

December 7, 2021

1 Introduction

The objective of this project was to build and program a line-following robot to track the black line trajectory shown in Fig. 1. We built our robot using the two-wheeled turtle bot chassis fitted with three line tracking sensors to detect the black line trajectory. We selected the RomeoV2 as our choice of microcontroller. The rest of this report provides further details about the individual components of our robot and the final configuration of the assembly. In addition to that, we also describe the control logic that we used to program the microcontroller, using Arduino, such that the robot reaches the destination by following the black line trajectory. Finally, we present the results regarding the robot's performance.

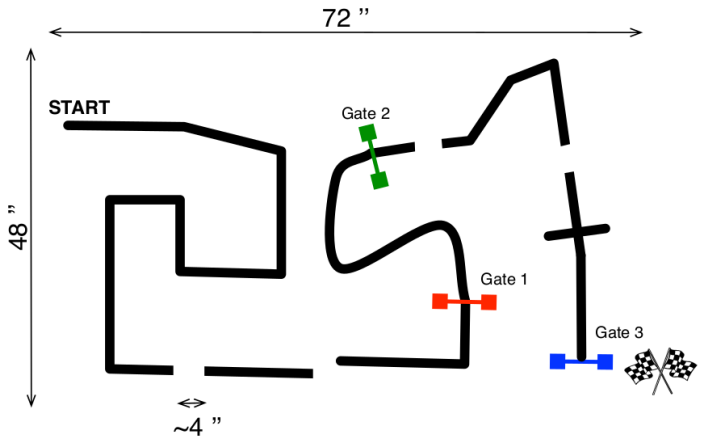

Figure 1: Track trajectory

2 Robot architecture

2.1 Robot chassis

The fundamental structure of our robot is made up of the turtle bot chassis that is shown in Fig. 2. The chassis itself provides the capability to install sensors like proximity or line tracking sensors along its periphery, as well as install control boards and batteries on the top plate. In addition to the chassis, the basic platform of our robot consists of two wheels with corresponding DC motors and a front caster ball.

2.2 Control board

We selected the RomeoV2 control board which is based on the Arduino Leonardo microcontroller as the brains of our project (Fig. 3). One key advantage of this board is that it has an integrated DC motor driver. In addition to that, the microcontroller is directly programmable using a USB. The control board has a total of 14 TTL (Transistor-Transistor Logic) 3-wire interfaces, three of which we

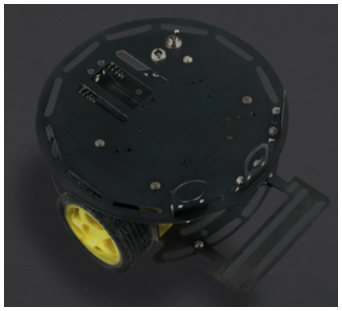

Figure 2: Turtle bot platform

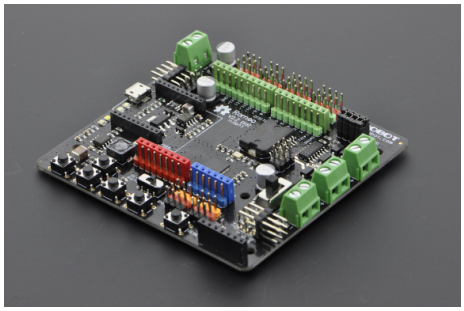

Figure 3: Robot control board

used for sensors. The control board also has two terminal blocks designed for motor connection, which we used both of.

2.3 Sensors

In order to design a robot to follow a black line pattern, we would need sensors that can detect the black lines. For this purpose, we chose the Gravity: Digital Line Tracking Sensor (Fig. 4) which can be used to detect either black or white colors. This sensor, with a supply voltage of 3.3-5.0V, outputs a TTL signal which is used to detect a black line on a white background or a white line on a dark background. If the signal voltage is low, surface under the sensor is black and if the voltage is high, the surface is white.

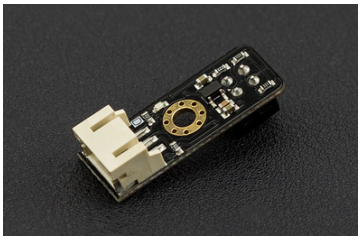

Figure 4: Gravity line tracking sensor

Component	Operating Voltage Range
Romeo V2 control board (input voltage)	5V (7-20V)
DC motor	3-7.5V
Gravity sensor	3.3-5V

Table 1: Operating voltages of the components.

3 Assembling the components

The components analyzed above were assembled together by mounting the hardware on the robot's chassis and connecting all parts to the control board (Figs. 5-6). Specifically, three line-tracking sensors were placed on the sensor mounting grill in the front of the robot, in a triangular shape. The left and right sensors were placed parallel to each other on the back of the grill, with a lateral gap between them. The center sensor was placed in the middle of the grill's front rail. The control board and battery holder were installed on the fuselage upper plate, at the top of the robot, using Velcro.

Regarding the connections of the components with the control board, all of them have been highlighted in Fig. 7. First, the battery holder was wired to a power switch, which was then wired to the corresponding terminal block that provides power to both the motors and the board itself. Next, the left and right motor were connected with the board via the terminal blocks *M1* and *M2*, respectively.

Moreover, the left, center and right line-tracking sensors were connected with the 0,1 and 2 TTL 3-wire interfaces of the board, respectively. It should be noted that five 1.5V AA batteries were used for the battery holder, while the power supplied to the board was controlled using the power switch discussed above. This power switch was mounted to the top plate of the chassis.

Figure 5: Front view of finalized robot design.

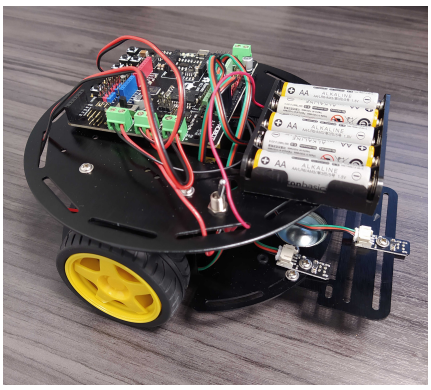

Figure 6: Side view of finalized robot design.

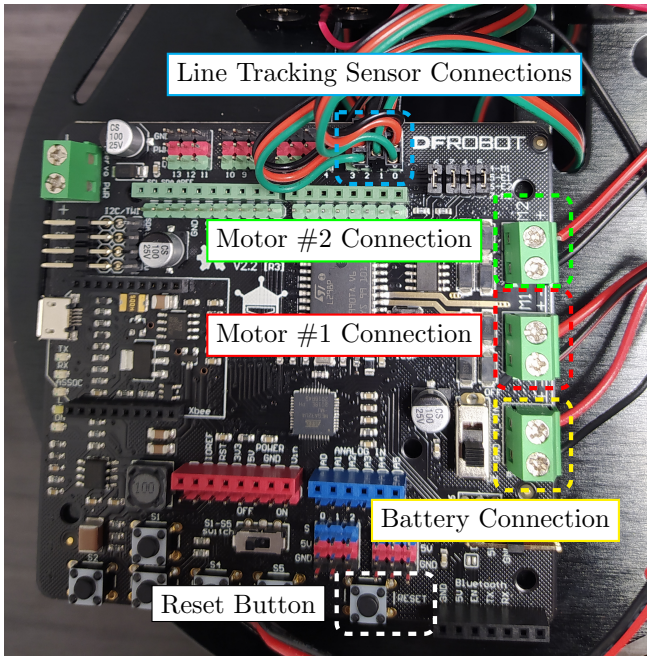

Figure 7: RomeoV2 control board with labelled connections.

4 Controlling the robot

Given the configuration of the robot and the placement of the sensors explained in the previous section, we used the Arduino IDE to program our robot such that it follows the track trajectory. The central idea of our control logic is based on the combinations of the three sensors detecting the black line. Just as a note of clarification, when a sensor detects the black line, the LED on the sensor turns off and when it detects white space, it turns back on. In the most simple scenario, if only the center sensor detects the black line, we command the robot to keep moving straight. Now, as shown in Fig. 8, when the track trajectory starts turning to the right, at some point the center sensor will no longer see the black line, but the right sensor does. In such a scenario, we command the robot to turn right (by only giving a voltage to the left motor) until only the center sensor detects the line again, after which it continues to move straight. In Fig. 9, we illustrate the same control logic, now using a hard left turn as an example.

If the track were simply a single black line from start to finish, the above control logic would be sufficient in controlling the robot. But since the track has both gaps and a crossroad, additional scenarios needed to be considered. To account for the crossroad, if all three sensors detect the line, we command the robot to simply continue moving forward. To account for the gaps, two concepts needed to be considered. First, if none of the three sensors detected the line, we again commanded the robot to continue forward. Second, if it just so happened that when the robot was making a turn (see paragraph above for details) and the center sensor went through the gap, the robot would then do a full 180° turn and begin traveling through the course backward. To account for this, we added a rotation limit to the robot. We command the robot, using delay functions, to only search for a line for about 150°. If the center sensor does not find the line in that time, we command the robot to turn back 60°, completing an overall turn of 90° with respect to it's original path, and then continue forward. The robot was programmed in this way to still allow for sharp turns greater than 90°, but also to add an extra level of safety to ensure the robot never turns completely around. Each of these scenarios are depicted in the flow chart in Fig. 10.

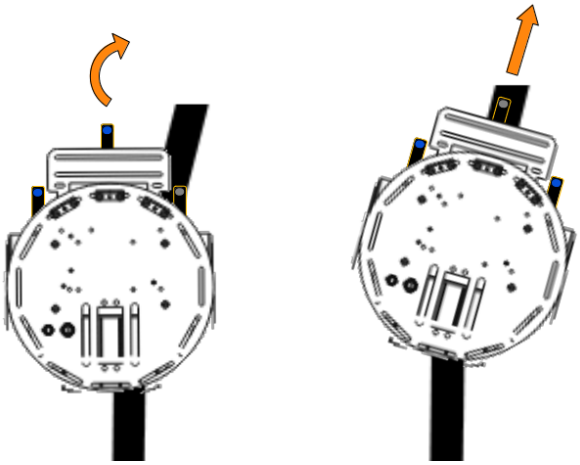

Figure 8: Illustration of slight right turn using control logic shown in Fig 10.

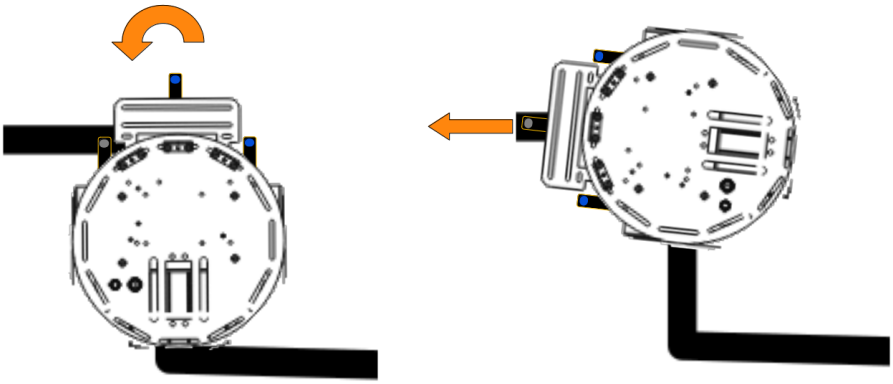

Figure 9: Illustration of hard left turn using control logic shown in Fig 10.

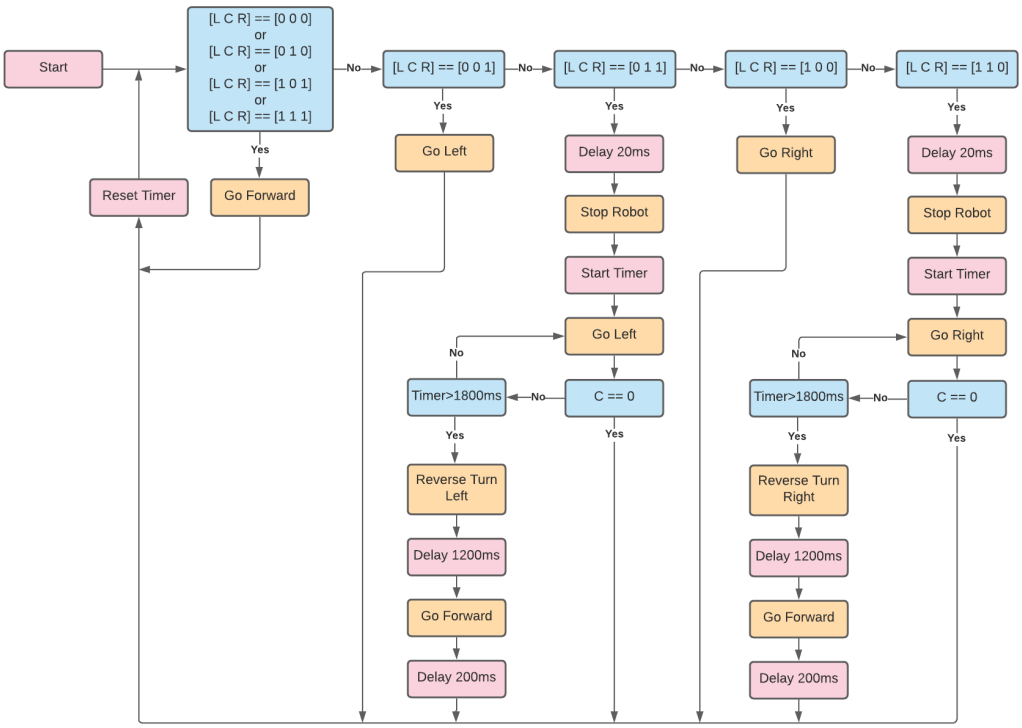

Figure 10: Robot control logic. Note that the format for this figure is [LeftSensor CenterSensor RightSensor]. Also, 0 corresponds to the sensor seeing the black line, and 1 corresponds to the sensor seeing the white paper.

5 Results

After completing both the assembly and the coding for the robot, several trial tests were performed, by printing the actual track. Here, the five best trials, with respect to duration, are listed, where in all of them the robot managed to track the black line trajectory closely and reach the finish line (Table 2). The average duration of the trials was found to be equal to 27.15 s. It should be noted that brand new batteries were used for all five trials, while the control board was reset before every trial using the reset button depicted in Fig. 7.

Trial Number	**Trial Duration (s)**
Trial #1	26.38
Trial #2	27.05
Trial #3	27.41
Trial #4	27.45
Trial #5	27.45
Average Trial	**27.15**

Table 2: Duration of five trials tested on the actual track.

6 Appendix

A Arduino code

```
#include <Servo.h>

// Definition of variables associated with motors based on board's schematic
int E1 = 5;     //M1 Torque Control (Left Motor)
int E2 = 6;     //M2 Torque Control (Right Motor)
int M1 = 4;     //M1 Direction Control (Left Motor)
int M2 = 7;     //M1 Direction Control (Right Motor)

// Initialization of useful variables
int RightSensor = 2;  // Variable associated with the line-tracking sensor placed on ...
    the left of the robot
int LeftSensor = 0;   // Variable associated with the line-tracking sensor placed on ...
    the right of the robot
int CenterSensor = 1; // Variable associated with the line-tracking sensor placed on ...
    the center of the robot

int sensor_left = 0;  // Variable to store the status of the line-tracking sensor ...
    placed on the left of the robot
int sensor_right = 0; // Variable to store the status of the line-tracking sensor ...
    placed on the right of the robot
int sensor_center = 0;// Variable to store the status of the line-tracking sensor ...
    placed on the center of the robot
int timer = 0;        // Variable to store time instants to avoid 360 turns

void stop(void)                   // Function that stops both motors
{
digitalWrite(E1,LOW); // Send zero torque to the left motor
digitalWrite(E2,LOW); // Send zero torque to the right motor
}
void back_off (char a,char b)     // Function that rotates motors backwards
{
analogWrite (E1,a);   // Apply user defined torque to the left motor
digitalWrite(M1,HIGH);// Set torque direction to HIGH for backwards rotation
analogWrite (E2,b);   // Apply user defined torque to the right motor
digitalWrite(M2,HIGH);// Set torque direction to HIGH for backwards rotation
}
void advance (char a,char b)     // Function that rotates motors forward
{
analogWrite (E1,a);   // Apply user defined torque to the left motor
digitalWrite(M1,LOW); // Set torque direction to LOW for forward rotation
analogWrite (E2,b);   // Apply user defined torque to the right motor
digitalWrite(M2,LOW); // Set torque direction to LOW for forward rotation
}
void turn_R (char a,char b)     // Function that turns the robot in-place towards to ...
    the right
{
analogWrite (E1,a);   // Apply user defined torque to the left motor
digitalWrite(M1,LOW); // Set torque direction to LOW for forward rotation
analogWrite (E2,b);   // Apply user defined torque to the right motor
digitalWrite(M2,HIGH);// Set torque direction to HIGH for backwards rotation
}
void turn_L (char a,char b)     // Function that turns the robot in-place towards to ...
    the left
{
analogWrite (E1,a);   // Apply user defined torque to the left motor
digitalWrite(M1,HIGH);// Set torque direction to HIGH for backwards rotation
analogWrite (E2,b);   // Apply user defined torque to the right motor
digitalWrite(M2,LOW); // Set torque direction to LOW for forward rotation
}

void setup(void)
{
}

```

```
57 void loop()
58 {
59 //Wait for button push
60   sensor_left = digitalRead(LeftSensor);                                    // Load ...
         the status of the line-tracking sensor on the left
61   sensor_center = digitalRead(CenterSensor);                                // Load ...
         the status of the line-tracking sensor on the center
62   sensor_right = digitalRead(RightSensor);                                  // Load ...
         the status of the line-tracking sensor on the right
63
64   timer = 0;                                                                // ...
         Initialization of timer variable to avoid 360 turns
65
66 // sensor_something = 0 => detect the line
67 // sensor_something = 1 => not detecting the line
68
69 // There is a total of 8 different cases (2^3)
70 if (sensor_left == 1 && sensor_center == 1 && sensor_right == 1) {         // None of ...
       the three sensors detect any lines, keep moving forward
71     advance (132,175);
72 } else if(sensor_left == 1 && sensor_center == 1 && sensor_right == 0) {  // Only ...
       the right sensor detects a line, turn right
73     delay(20);                                                            // Small ...
           delay to keep moving forward before turning for allignment purposes
74     stop ();                                                              // Stop ...
           the robot
75     timer = millis();                                                     // Store ...
           the timing that the robot initiated rotation
76     while (digitalRead(CenterSensor) == 1){                               // while ...
           not detecting center sensor
77       advance (150,0);                                                    // Turn ...
             right while moving forward
78       if (millis()≥timer+1800 || digitalRead(LeftSensor) == 0){          // Check ...
             whether the left sensor detected the line or if enough time for a full ...
             rotation has passed by
79         if (millis()≥timer+1800){                                         // Check ...
               whether the robot completed approx. a full rotation
80           back_off (150,0);                                               // Turn ...
                 left while moving backward
81           delay(1200);                                                    // Delay ...
                 to execute backward motion for enough time in order to return to the ...
                 initial position
82           advance(132,175);                                               // Move ...
                 staight forward (different torques between the motors were necessary ...
                 to maintain straight path)
83           delay(200);                                                     // Delay ...
                 to execute straight motion for enought time
84         }
85         break;                                                            // Break ...
               from rotation loop
86      }
87     }
88 } else if(sensor_left == 1 && sensor_center == 0 && sensor_right == 1) {  // Only ...
       the center sensor detects a line, move forward
89     advance (132,175);
90 } else if(sensor_left == 1 && sensor_center == 0 && sensor_right == 0) {  // Both ...
       the center and right sensors detect a line, turn to the right
91     advance (150,0);
92 } else if(sensor_left == 0 && sensor_center == 1 && sensor_right == 1) {  // Only ...
       the left sensor detects a line, turn left
93     delay(20);                                                            // Small ...
           delay to keep moving forward before turning for allignment purposes
94     stop ();                                                              // Stop ...
           the robot
95     timer = millis();                                                     // Store ...
           the timing that the robot initiated rotation
96     while (digitalRead(CenterSensor) == 1){                               // while ...
           not detecting center sensor
97       advance (0,150);                                                    // Turn ...
             left while moving forward
```

```
98        if (millis()≥timer+1800 || digitalRead(RightSensor) == 0){         // Check ...
              whether the right sensor detected the line or if enough time for a full ...
              rotation has passed by
99          if (millis()≥timer+1800){                                        // Check ...
                whether the robot completed approx. a full rotation
100           back_off (0,150);                                              // Turn ...
                  right while moving backward
101           delay(1200);                                                   // Delay ...
                  to execute backward motion for enough time in order to return to the ...
                  initial position
102           advance(132,175);                                              // Move ...
                  staight forward (different torques between the motors were necessary ...
                  to maintain straight path)
103           delay(200);                                                    // Delay ...
                  to execute straight motion for enought time
104         }
105         break;                                                           // Break ...
                from rotation loop
106       }
107     }
108   } else if(sensor_left == 0 && sensor_center == 1 && sensor_right == 0) {  // Both ...
        the left and right sensors detect a line, keep moving forward (it happens)
109     advance (132,175);                                                   // Move ...
            staight forward (different torques between the motors were necessary to ...
            maintain straight path)
110   } else if(sensor_left == 0 && sensor_center == 0 && sensor_right == 1) {  // Both ...
        the center and left sensors detect a line, turn to the left
111     advance (0,150);                                                     // Turn ...
            left while moving forward
112   } else if(sensor_left == 0 && sensor_center == 0 && sensor_right == 0) {  // All ...
        three sensors see lines, this is a crossroad, keep moving forward
113     advance (132,175);                                                   // Move ...
            staight forward (different torques between the motors were necessary to ...
            maintain straight path)
114   }
115
116   }
```