Checkpoint #2 Report

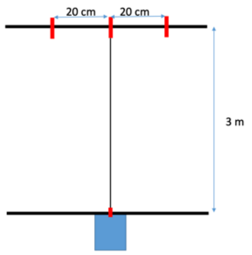
[EECN30169] Mobile Robot 2022

Student ID: 311605004 **Name:** 劉子齊 **Date: 2022.10.26**

1. Introduction:

For checkpoint 2, our goal is to implement our mobile robot and make it able to make corresponding motions according to the user’s control. We need to make motion control to the DC motor through the Raspberry Pi and Arduino by the L298N motor control module.

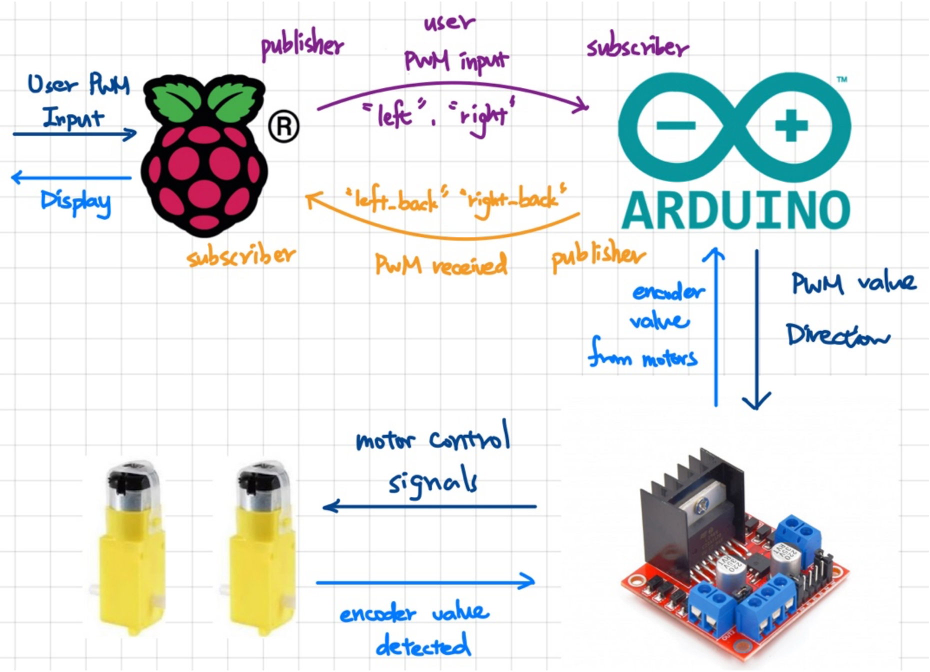
Besides having the robot to make corresponding actions according to the user’s control, we also need to make the robot go as straight as possible. To determine how straight our robot can go, we have to run on the court in the figure on the right.



As shown in the figure, the court basically looks like the character “I”. At the end of the court, which is the upper part of the court, we can see there is an intersection of the horizontal line and the vertical line. The closer the rear wheel of our mobile robot is to the intersection point, the better our robot performance is. In addition, the “going straight” task will be taken as failed if the rear wheel of our mobile robot went outside the 2 endpoints of the goal line of the court, which has the deviation of 20cm.

2. Description of Design:

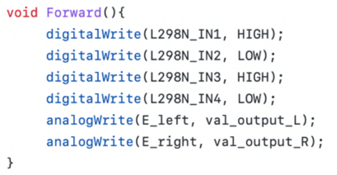
Below is the workflow of my implementation to the communication between the Raspberry Pie, the Arduino, the L298N motor control module, and the 2 DC motors.



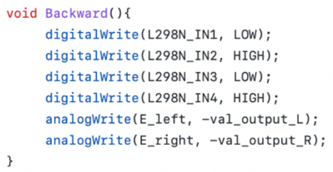
For the Raspberry Pie, initially, I had the Raspberry Pie request 2 inputs from the user as the PWM value of the left and the right DC motor, which would be typed on the terminal of the user’s computer connected to the Raspberry Pie through ssh connection.

After receiving the 2 PWM input values from the user, the Raspberry Pie would publish the 2 PWM values in the topics of “left” and “right”. Then the Arduino would subscribe to these topics which made the Arduino able to obtain the user input whenever the Raspberry Pie published a new number.

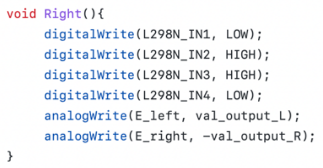
As the Arduino received the PWM values from the Raspberry Pie, the Arduino would make some simple determinations and send the corresponding values to the L298N motor control module according to different cases, which the different cases are shown in the following figures.



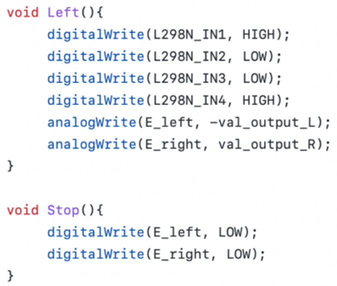
```
void Forward(){
    digitalWrite(L298N_IN1, HIGH);
    digitalWrite(L298N_IN2, LOW);
    digitalWrite(L298N_IN3, HIGH);
    digitalWrite(L298N_IN4, LOW);
    analogWrite(E_left, val_output_L);
    analogWrite(E_right, val_output_R);
}
```



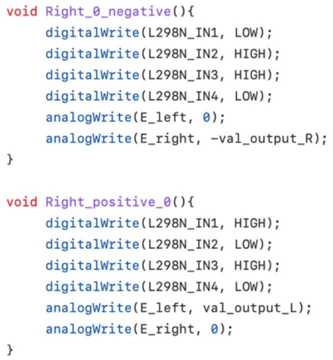
```
void Backward(){
    digitalWrite(L298N_IN1, LOW);
    digitalWrite(L298N_IN2, HIGH);
    digitalWrite(L298N_IN3, LOW);
    digitalWrite(L298N_IN4, HIGH);
    analogWrite(E_left, -val_output_L);
    analogWrite(E_right, -val_output_R);
}
```



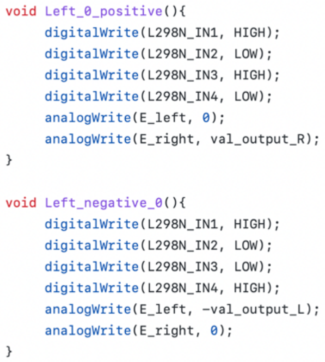
```
void Right(){
    digitalWrite(L298N_IN1, LOW);
    digitalWrite(L298N_IN2, HIGH);
    digitalWrite(L298N_IN3, HIGH);
    digitalWrite(L298N_IN4, LOW);
    analogWrite(E_left, val_output_L);
    analogWrite(E_right, -val_output_R);
}
```



```
void Left(){
      digitalWrite(L298N_IN1, HIGH);
      digitalWrite(L298N_IN2, LOW);
      digitalWrite(L298N_IN3, LOW);
      digitalWrite(L298N_IN4, HIGH);
      analogWrite(E_left, -val_output_L);
      analogWrite(E_right, val_output_R);
}

void Stop(){
      digitalWrite(E_left, LOW);
      digitalWrite(E_right, LOW);
}
```



```
void Right_0_negative(){
      digitalWrite(L298N_IN1, LOW);
      digitalWrite(L298N_IN2, HIGH);
      digitalWrite(L298N_IN3, HIGH);
      digitalWrite(L298N_IN4, LOW);
      analogWrite(E_left, 0);
      analogWrite(E_right, -val_output_R);
}

void Right_positive_0(){
      digitalWrite(L298N_IN1, HIGH);
      digitalWrite(L298N_IN2, LOW);
      digitalWrite(L298N_IN3, HIGH);
      digitalWrite(L298N_IN4, LOW);
      analogWrite(E_left, val_output_L);
      analogWrite(E_right, 0);
}
```

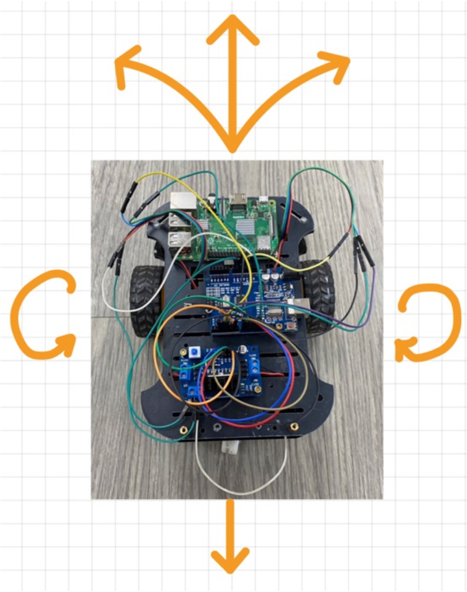


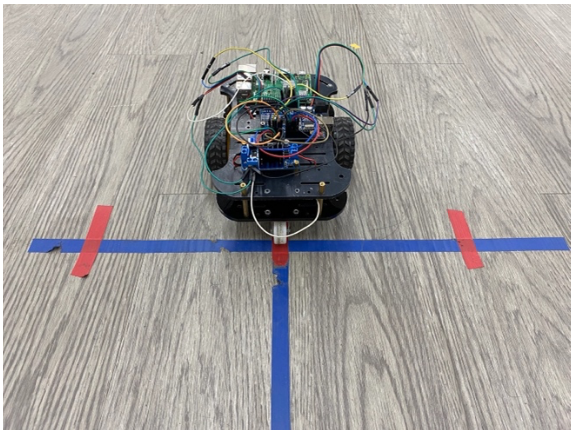
```
void Left_0_positive(){
      digitalWrite(L298N_IN1, HIGH);
      digitalWrite(L298N_IN2, LOW);
      digitalWrite(L298N_IN3, HIGH);
      digitalWrite(L298N_IN4, LOW);
      analogWrite(E_left, 0);
      analogWrite(E_right, val_output_R);
}

void Left_negative_0(){
      digitalWrite(L298N_IN1, HIGH);
      digitalWrite(L298N_IN2, LOW);
      digitalWrite(L298N_IN3, LOW);
      digitalWrite(L298N_IN4, HIGH);
      analogWrite(E_left, -val_output_L);
      analogWrite(E_right, 0);
}
```

As shown in the figures above, we separated all the situations into 9 cases, and each case had its own control values upon the two motors. After getting the corresponding value of each case, we send the values to the L298N motor control module, which acted as a controller that uses an H-Bridge to control the direction and speed of up to 2 DC motors here.

After receiving the control signals, the motor would react correspondingly to the control signals. At the same time, we had the Arduino would publish the PWM value received from the Raspberry Pie back in the topic named “left_back” and “right_back”. Then the Raspberry Pie would subscribe to this topic and would display the PWM value received, and this is the whole idea of the implementation to checkpoint 2.

3. Result





In the figures shown above, we can see how our mobile robot looks like. After our implementation, our robot can perform going forward, backward, right turn, left turn, turning clockwise, and turning counterclockwise. Although our robot didn’t go so straight in the formal demonstrate, through the figure on the right, which is the result of one of our testing results, it went straight perfectly, and I’ll make further discussion in the following session.

4. Discussion

Before programing, we had to deal with the electric circuits first. It slightly bothered me in the very beginning, since I didn’t consider that the electric potentials were different for the two batteries. After sharing their ground, the problem was solved.

About the problem we met on the “going as straight as possible” task, I think there were 2 main reasons causing us unable to perform stably. First, when controlling the DC motors, we didn’t made use of the PID controller which will take the encoder value returned by the DC motors and make the motor control to be more precise, but we controlled the motor by directly changing the PWM values. Second, the precision of the motor and the environment are too low and unstable. Hence, we could not perform as perfect as we did when testing on the formal demonstrate.

However, we will start applying sensors in the following checkpoints, which will give us more data and space to conduct correction on our mobile robot. I believe our team will have better performance in the next checkpoint.