ACTIVITY 5 - Arduino 5: Analog Output - Motor

Jason Harvey Lorenzo, March 9, 2024
ENGG 122.02, Department of Electronics, Computer, and Communications Engineering
School of Science and Engineering, Ateneo de Manila University
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—This project explores Arduino's analog output capabilities through motor control using a P-BOT R2 interface module in a mobile robot (mobot) kit. The source code orchestrates the mobot's movement to spell the letter "Z," utilizing dedicated functions for essential commands like forward/backward motion and left/right turns with PWM control. The Arduino setup involves connecting an Arduino Duemilanove Microcontroller to the P-BOT R2 Interface Module, powered by the mobot's lithium batteries.**

***Index Terms*—Arduino, motor control, P-BOT 32, mombile robot**

This project utilizes a mobile robot (mobot) kit, notably the P-BOT R2 interface module, to further apply Arduino's analog output capabilities through motor control. While the mobot kit offers various functionalities, this activity focuses solely on utilizing motor-related functions, as detailed in Table I.

TABLE I
MOTOR CONTROL

Pin ID	Description
M1DIR	Motor Driver1 Direction (Forward/Reverse)
M1RUN	Motor Driver1 Run
M2RUN	Motor Driver2 Run
M2DIR	Motor Driver2 Direction (Forward/Reverse)

The DIR function accepts boolean values, with HIGH directing the motion forward and LOW indicating reverse. Conversely, the RUN function accepts integer values ranging from 0 to 255, mirroring the analogWrite() range, translating these to approximately 0 to 8 volts in proportion.

I. Program Listing

This project involves programming a mobot unit to trace out a specific letter—in this instance, the letter "Z" as depicted in Fig. 1. To streamline the process, dedicated functions have been developed to execute essential movement commands for the mobot, including forward and backward motion, as well as left and right turns.

A. Source Code

```
int rightDIR = 8;
int leftDIR = 11;
int rightRUN = 9;
int leftRUN = 10;
```

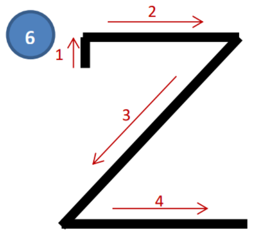

Fig. 1. Letter Z Mobot Movement Pattern

```
//adjust PWM depending on how strong
// motors are relative to each other
int leftPWM = 250;
int rightPWM = 180;


//adjust time delay to ensure 90
// degree turn
int delay90deg = 450;
int delay45deg = delay90deg/2;

void goForward(int delayT){
  digitalWrite(rightDIR, HIGH);
  digitalWrite(leftDIR, HIGH);
  analogWrite(rightRUN, rightPWM);
  analogWrite(leftRUN, leftPWM);
  delay(delayT);
  for(int i=8; i<=11; i++)
    digitalWrite(i, LOW);
}

void goBackward(int delayT){
  digitalWrite(rightDIR, LOW);
  digitalWrite(leftDIR, LOW);
  analogWrite(rightRUN, rightPWM);
  analogWrite(leftRUN, leftPWM);
  delay(delayT);
  for(int i=8; i<=11; i++)
    digitalWrite(i, LOW);
}

void turnRight(int delayT){
  digitalWrite(rightDIR, LOW);
  digitalWrite(leftDIR, HIGH);
  analogWrite(rightRUN, rightPWM);
  analogWrite(leftRUN, leftPWM);
  delay(delayT);
  for(int i=8; i<=11; i++)
    digitalWrite(i, LOW);
}

void turnLeft(int delayT){
  digitalWrite(rightDIR, HIGH);
  digitalWrite(leftDIR, LOW);
  analogWrite(rightRUN, rightPWM);
  analogWrite(leftRUN, leftPWM);
  delay(delayT);
  for(int i=8; i<=11; i++)
    digitalWrite(i, LOW);
}

void setup() {
  for(int i=8; i<=11; i++)
    pinMode(i, OUTPUT);

}

void loop() {
  //drawing pattern 6 (letter Z)
  //first step
  goForward(750);

  //second step
  turnRight(delay90deg);
  goForward(3000);

  //third step
  turnLeft(delay45deg);
  goBackward(3700);

  //fourth step
  turnRight(delay45deg);
  goForward(3000);

  while(1);
}
```

B. How It Works

Pins 8 and 11 controls the direction of the right and left motors, respectively, while pins 9 and 10 will regulate the voltage supplied to power these motors. Ideally, both motors should respond uniformly to identical voltage inputs, yet actual hardware disparities lead to varying responses. To address this, the variables leftPWM and rightPWM (PWM denoting Pulse Width Modulation) were introduced to manage the voltage outputs for each motor. This facilitated the process of aligning their responses through trial and error. Additionally, the time delay required for the motors to execute a 90-degree turn was determined also through trial and error. By halving this duration, the appropriate time delay for a 45-degree turn was derived.

As previously mentioned, dedicated functions were employed to define the movement commands of the mobot. The goForward() function configures both motors to move in the forward direction (1) and sets their power levels according to the leftPWM and rightPWM values. Following a specified time delay parameter, delayT, the motors are deactivated in preparation for subsequent commands. Similarly, the goBackward() function operates on the same principle but sets the direction to reverse (0). Meanwhile, turnRight() and turnLeft() functions adhere to a similar principle, albeit adjusting the direction of each motor differently. In turnRight(), the left motor moves forward while the right moves backward, whereas in turnLeft(), the left motor moves backward while the right moves forward. Once again, all these functions accept a time delay parameter, allowing for manipulation of their usage to achieve the desired outcomes by adjusting the delay duration.

In the setup() function, the Arduino assigns control to specific pins, as indicated in Table I. Subsequently, the loop() function orchestrates the mobot's movement commands to trace out the letter "Z," illustrated in Fig. 1. The sequence involves instructing the mobot to advance for 0.75 seconds, followed by a 90-degree right turn. Subsequently, it advances for 3 seconds, executing a 45-degree left turn, then reverses for 3.7 seconds. Finally, it turns right for 45 degrees before moving forward for 3 seconds, effectively spelling out the letter "Z" through mobot movement. The line while(1); captures the program's flow within an empty loop to halt the continuous execution of the loop() function.

II. Arduino Setup

As illustrated in Fig. 2 and 3, the Arduino setup features an Arduino Duemilanove Microcontroller linked to the P-BOT R2 Interface Module. Power is provided by the mobot's lithium batteries, while programming of the microcontroller is facilitated through a laptop using a USB connection. A video demonstration can be accessed here.

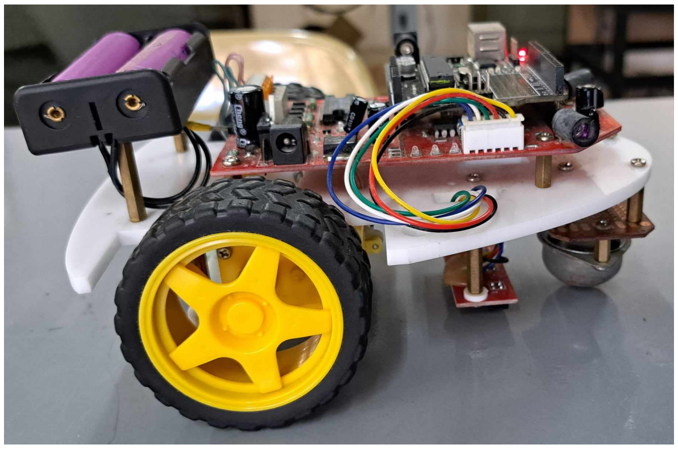

Fig. 2. Mobot Setup (Side View)

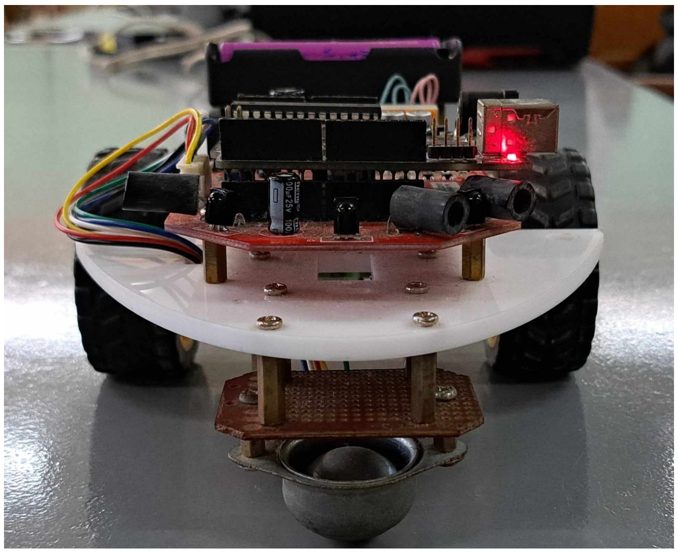

Fig. 3. Mobot Setup (Front View)