Part 1: Making the Car Move

In this part, I learned how to program an Arduino car to move and take turns (which will later help me in making a self-driving car). My goal was to move the car 1m forward, 1m left, and 1m right. A series of steps were taken that helped me achieve this goal. First, I connected the Arduino, breadboard, and motors using electrical cables that were both pre-stripped and that I stripped myself. This allowed the components to receive power and run the code. I edited the given code in order to make the robot move and turn based on my goal. This part took the most time because the time the motors ran for (delay time) determined how far the robot went and the angles at which it turned. Furthermore, whenever I tried to get the delay time values, they always changed after each trial because the voltage of the batteries decreased. To resolve this issue, I came up with the idea of using a portable battery bank as I was able to charge it before running the robot. As a result, I was able to get accurate delay value times. Although these steps helped me to get the car to move and turn fairly accurately, there were an additional 2 problems which were hardware related which I tried my best to mitigate with code. One of the encountered problems was that the motors were not very accurate. To expand, the right motor was less powerful than the left motor. Hence, I edited the code to decrease the power of the left motor in order to help the car move straight. The other problem was that the small back wheel caused the robot to turn in the beginning when it was not programmed to. To resolve this issue, I added a 6-second delay timer to the start of my code. This allowed me to set up the back wheel right before the robot started to move. As a result, this allowed the robot to begin fairly straight in the start. Overall, this project has allowed me to learn about how Arduinos work and the various things that you can do with them.

Code:

```
//identify pins
int ltPower = 6;
int ltFwd = 7;
int ltBwd = 8;
int rtPower = 11;
int rtFwd = 12;
int rtBwd = 13;

void setup() {
   Serial.begin(9600);
   pinMode(ltPower, OUTPUT);
   pinMode(ltFwd, OUTPUT);
   pinMode(ltBwd, OUTPUT);
   pinMode(rtPower, OUTPUT);
   pinMode(rtFwd, OUTPUT);
   pinMode(rtBwd, OUTPUT);
}
```

```
// move the robot
void loop() {

    // 6s to set up robot
    delay(6000);

    // drive forward
    lftMotor(200);
    rgtMotor(255);
    delay(4000);

    // stop
    lftMotor(0);
    rgtMotor(0);
    delay(3000);

    // turn right
    lftMotor(-200);
    rgtMotor(-255);
    delay(2000);

    // stop
    lftMotor(0);
    rgtMotor(0);
    delay(1000);

    // drive forward
    lftMotor(255);
    rgtMotor(200);
    delay(3000);

    // stop
    lftMotor(0);
    rgtMotor(0);
    delay(1000);

    // turn left
    lftMotor(-255);
    rgtMotor(-200);
    delay(3000);

    // drive forward
```

```
    lftMotor(255);
    rgtMotor(200);
    delay(3000);
}

void rgtMotor(int power)
{
  if (power>=0) // set to drive forward
  {
    digitalWrite(rtFwd, HIGH);
    digitalWrite(rtBwd, LOW);
  }
  else // set to drive in reverse
  {
    digitalWrite(rtFwd, LOW);
    digitalWrite(rtBwd, HIGH);
  }
  analogWrite(rtPower,abs(power));
}

void lftMotor(int power)
{
  if (power>=0) // set to drive forward
  {
    digitalWrite(ltFwd, HIGH);
    digitalWrite(ltBwd, LOW);
  }
  else // set to drive in reverse
  {
    digitalWrite(ltFwd, LOW);
    digitalWrite(ltBwd, HIGH);
  }
  analogWrite(ltPower,abs(power));
}
```

Part 2: Making a Self-Driving Car

In addition to programming the Arduino to make the car move according to the given guidelines, I added an ultrasonic sensor to the car to give it additional capabilities. This addition allowed the car to come to a stop and reverse when approaching an object placed in front of it. This works by the transmitter (trig pin) sending a high-frequency sound (signal) at 40 000 Hz at 8 cycle sonic bursts. This signal travels through the air, hits objects, and as a result, bounces back (reflects) to the echo pin which outputs the time the sound wave traveled. This technology is used where knowing the distance from an object to another object quickly is useful. An example would be emergency braking systems in a vehicle. A series of steps were taken in order to make this feature work. First, the ultrasonic sensor was connected to the Arduino and supplied power through the breadboard. In order to send and receive data from the ultrasonic sensor, I identified the trig and echo pins as pins 2 and 3 in the code respectively. The distance from an object was calculated using the time received from the ultrasonic sensor and the speed of sound. Although I was successful in achieving the goal, there were some hurdles I had to overcome. First of all, in order for the car to have a fast reaction time, the code had to be altered. The code that retrieves the distance using the ultrasonic sensor is run in the loop with very little delay. This is in contrast to the original code used to move the car which uses delays lasting multiple seconds. The code was changed so that the distance was repeatedly checked, and if it was less than a certain distance, the car would stop and quickly drive back. Additionally, some of the wires connecting to the left engine were not working properly and had to be replaced. Even though I was able to resolve these issues, there are still underlying problems within the ultrasonic sensor. For instance, an object that poorly reflects or absorbs ultrasound, a noisy environment, or unnecessary objects within a 160° angle can cause false readings. These issues can be fixed by placing a foam dome around the sensor, facing the car forward so it can only sense from a more narrow-angle, and keeping the area clear of unnecessary objects. Overall, this project has allowed us to develop an understanding of how Arduinos work, what they can do, and real-life applications of ultrasonic sensors.

Code:

```
//identify pins
int trigPin = 2;
int echoPin = 3;
int ltPower = 6;
int ltFwd = 7;
int ltBwd = 8;
int rtPower = 11;
int rtFwd = 12;
int rtBwd = 13;

int motorSpeedL = 255;
```

```
int motorSpeedR = 255;

boolean goingBack;
boolean doneGoingBack;

boolean first;

void setup() {
  Serial.begin(9600);
  pinMode(ltPower, OUTPUT);
  pinMode(ltFwd, OUTPUT);
  pinMode(ltBwd, OUTPUT);
  pinMode(rtPower, OUTPUT);
  pinMode(rtFwd, OUTPUT);
  pinMode(rtBwd, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  first = true;
  goingBack = false;
  doneGoingBack = false;
}

void loop() {
  if (first) {
    delay(6000);
    first = false;
  }

  boolean changed = false;

  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); // send a new pulse
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // time for sound pulse to go and
return
  distance = (duration / 2) * 0.034029; // speed of sound is 0.034029 cm
per microsecond

  Serial.print(distance);
```

```
  Serial.println(" cm");

  if (!goingBack) {
    // tell the car to stop when the distance is less than 20cm
    if (distance < 20) {
      goingBack = true;
    } else {
      motorSpeedL = 255;
      motorSpeedR = 255;
      changed = true;
    }

    if (changed) {
      lftMotor(motorSpeedL);
      rgtMotor(motorSpeedR);
    }
  } else {
    if (!doneGoingBack) {
      // stop the car
      lftMotor(0);
      rgtMotor(0);
      delay(1000);

      // drive backwards
      lftMotor(-255);
      rgtMotor(-255);
      delay(500);

      // stop the car
      lftMotor(0);
      rgtMotor(0);
      delay(500);

      doneGoingBack = true;
    }
  }

  delay(200);
}

void rgtMotor(int power)
{
  if (power >= 0) // set to drive forward
```

```
  {
    digitalWrite(rtFwd, HIGH);
    digitalWrite(rtBwd, LOW);
  }
  else // set to drive in reverse
  {
    digitalWrite(rtFwd, LOW);
    digitalWrite(rtBwd, HIGH);
  }
  analogWrite(rtPower, abs(power));
}

void lftMotor(int power)
{
  if (power >= 0) // set to drive forward
  {
    digitalWrite(ltFwd, HIGH);
    digitalWrite(ltBwd, LOW);
  }
  else // set to drive in reverse
  {
    digitalWrite(ltFwd, LOW);
    digitalWrite(ltBwd, HIGH);
  }
  analogWrite(ltPower, abs(power));
}
```