PAMANTASAN NG LUNGSOD NG MAYNILA
(University of the City of Manila)
Intramuros, Manila

Microprocessors (Laboratory)

Laboratory Activity No. 6
DC Motor

Score

Submitted by:
Leader: Palacio, Leticia Mae
Baldanzo, Raizza Marrie C.
Baltes, Billy Renz C.
Belmonte, Jhade Loui M.
Magnabihon, Michael Lorenz M.
Tiu. Joshua Miguel Yaj A.

S 10:00am-1:00pm / CPE 0412.1-1

Submitted to:

Engr. Maria Rizette H. Sayo

Date Submitted:

09/12/2023

I. Objectives

This laboratory activity aims to implement the principles and techniques of hardware programming using Arduino through:

- Creating an automated parking gate system that opens when detecting an object in proximity and closing when there is no object in proximity.
- Integrate an LCD I2C for real-time status updates, providing feedback on the gate's operation.

II. Methods

- Perform the tasks and problems presented in the presentation.
- Present a unique implementation representative of the objectives.

III. Results

In order to create the DC Motor circuit, the materials used were an Arduino Uno R3, 2 DC motors, an H-bridge Motor Driver, a 9V battery, and 1 ultrasonic distance sensor.

Name	Quantity	Component
U1	1	H-bridge Motor Driver
M1 M2	2	DC Motor
U2	1	Arduino Uno R3
BAT1	1	9V Battery
DIST2	1	Ultrasonic Distance Sensor

Table 1. List of Components for the Automated Parking Gate Circuit

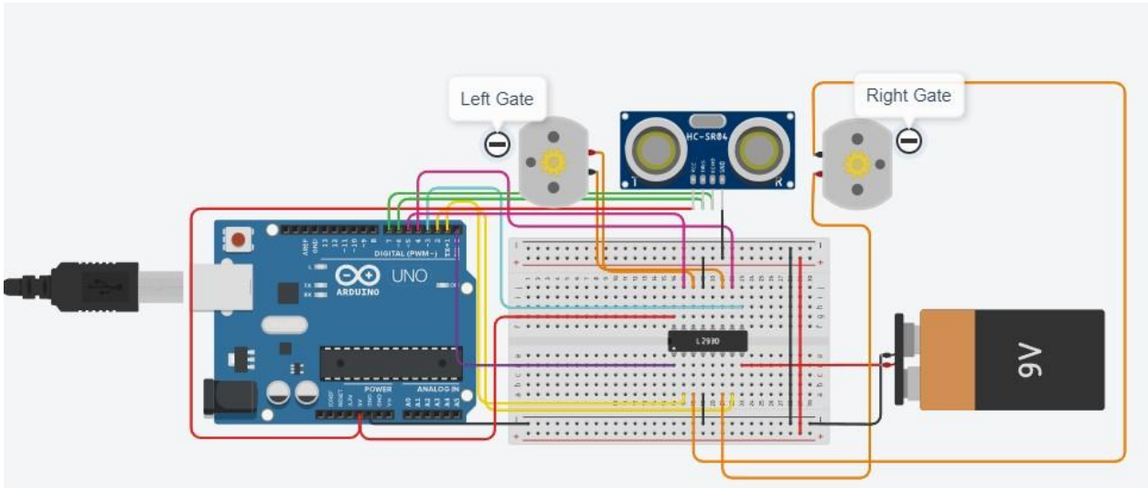

Figure 1. Circuit Diagram of the Automated Parking Gate Circuit

The circuit diagram is made by connecting 2 DC motors which represent a left and right gate in a parking area into an H-bridge motor driver IC (L293D). By connecting the output pins output 1, 2, 3, and 4 to each respective DC motor and connecting the input pins and enable pins to the Arduino. The IC is also connected to a 5v and 9v power source and ground. The ultrasonic sensor which acts as input for the whole circuit is then connected to the ground, into a 5v power source and its echo and trigger pins connected to the Arduino development board as input and output.

```
int ena1 = 0;//right motor
int in1 = 1; //right motor
int in2 = 2; //right motor
int ena2 = 3;//left motor
int in3 = 4; //left motor
int in4 = 5; //left motor
int trigPin = 6; //ultrasonic distance sensor
int echoPin = 7; //ultrasonic distance sensor

int counter = 1;

void setup()
{
    pinMode(ena1, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(ena2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);


}
```

Figure 2. First Part of the Program Code of the Automated Parking Gate Circuit

The first part of the program code initializes all the components as well as the pins used in the Arduino Uno R3 including its outputs which are connected the motor driver and the sensor. The right and left motors are also initialized as low which means it doesn't turn in any direction at the start of the program.

```
void gateopen()
{
  analogWrite(ena1, 255); //right motor speed
  digitalWrite(in1, HIGH); //rotate clockwise
  digitalWrite(in2, LOW); //rotate clockwise
  analogWrite(ena2, 255); //left motor speed
  digitalWrite(in3, LOW); //rotate counterclockwise
  digitalWrite(in4, HIGH); //rotate counterclockwise

  delay(1000); //simulated duration of gates opening
  counter=0; //used to control how many times a function will run
}

void gateclose()
{
  analogWrite(ena1, 255); //right motor speed
  digitalWrite(in1, LOW); //rotate counterclockwise
  digitalWrite(in2, HIGH); //rotate counterclockwise
  analogWrite(ena2, 255); //left motor speed
  digitalWrite(in3, HIGH); //rotate clockwise
  digitalWrite(in4, LOW); //rotate clockwise

  delay(1000);//simulated duration of gates opening
  counter=1; //used to control how many times a function will run
}

void gatestop()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

Figure 3. Second Part of the Program Code of the Automated Parking Gate Circuit

The second part of the program code includes the functions created within the program which is the gate open wherein the right motor rotates clockwise and the left motor rotates counterclockwise to simulate the gate opening. The rotation speed is also set in the function which is the analog value 255. The gate close function on the other hand rotates the right motor counterclockwise and the left motor clockwise to simulate the gate closing with the rotation speed for each motor set at analog value 255. Lastly, a gate stop function is created to stop the motors from spinning.

```
void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); //sending a pulse
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  int duration = pulseIn(echoPin, HIGH);
  //calculating the time it took to receive the pulse
  int distance= duration*0.034/2;
  //converting the duration to distance
  if (distance <= 300) //if object is within range
  {
   if (counter==1)
   {
     gateopen(); //open the gates
   }
   else
   {
     gatestop(); //stop rotation
   }
  }
  else
  {
   if (counter==0) //if object is not within range
   {
     gateclose(); //close the gates
   }
   else
   {
     gatestop(); //stop rotation
   }
  }
}
```

Figure 4. Third Part of the Program Code of the Automated Parking Gate Circuit

The Arduino performs the program by sending a pulse and calculating the duration before the pulse is received. The duration is then converted to distance which determines whether the object is within or not within the range set. An if-else condition is used to control the opening and closing of the gates. If the object is within 300 cm (simulated range) then the gates will open setting the counter to 0 in order to stop the rotation of the motors after some delay. If the object on the other hand is gets out of range or is not within range, the gates will close, and the counter will be set to 0 which stops the rotation of the motors after some delay.

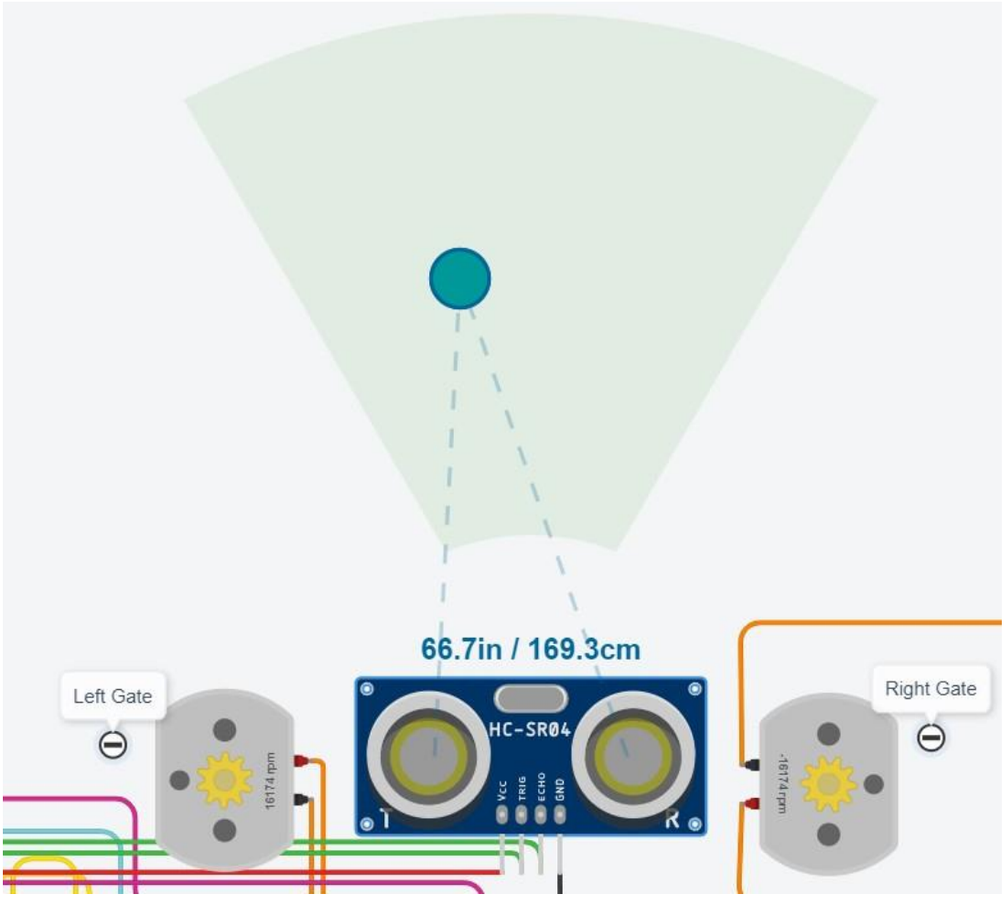

Figure 5. Simulation of the Parking Gates Opening when an Object is Within Range

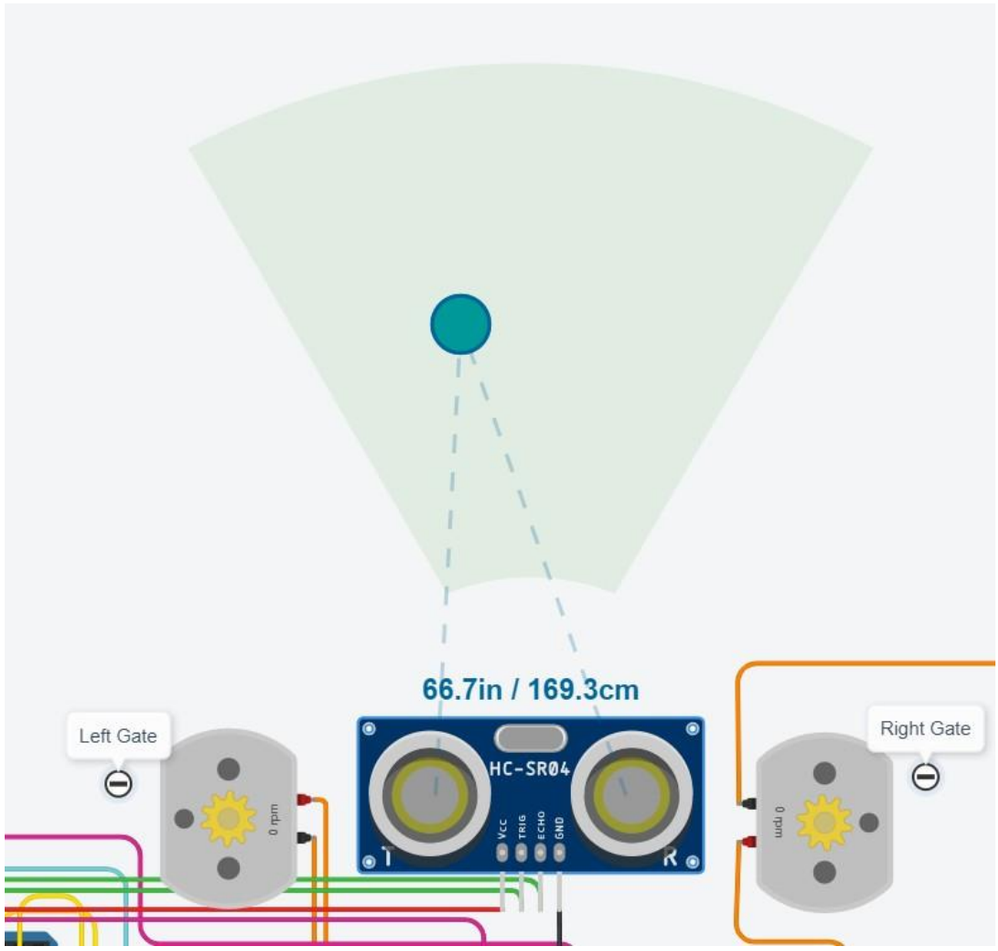

Figure 6. Motor Rotation Stopping After Opening the Gates

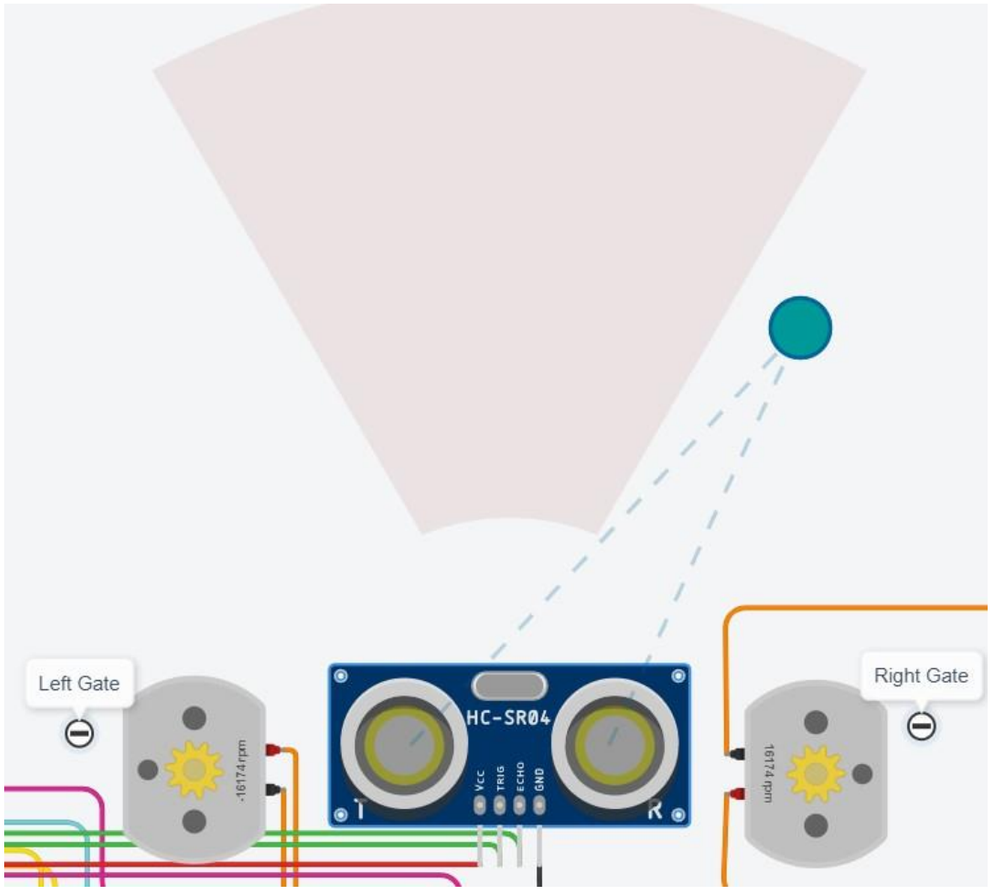

Figure 7. Simulation of the Parking Gates Closing when an Object is Not Within Range

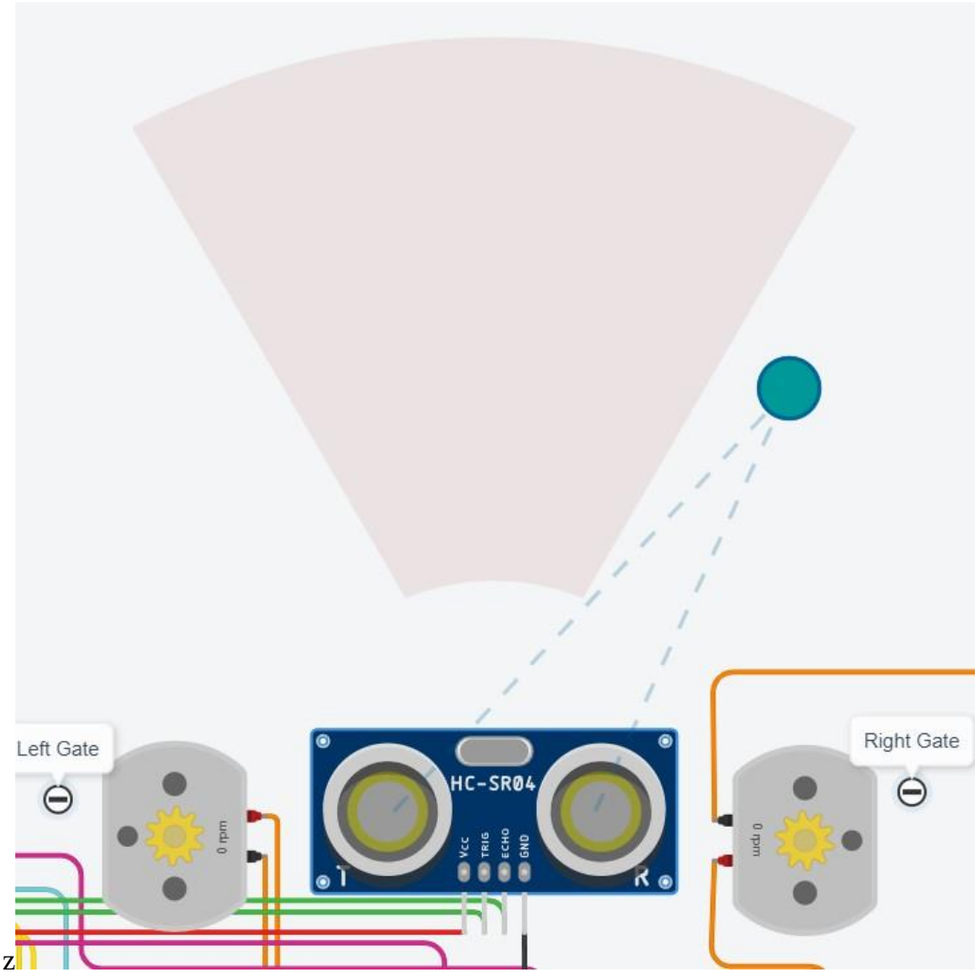

Figure 8. Motor Rotation Stopping After Closing the Gates

IV. Modification

Circuit Diagram:

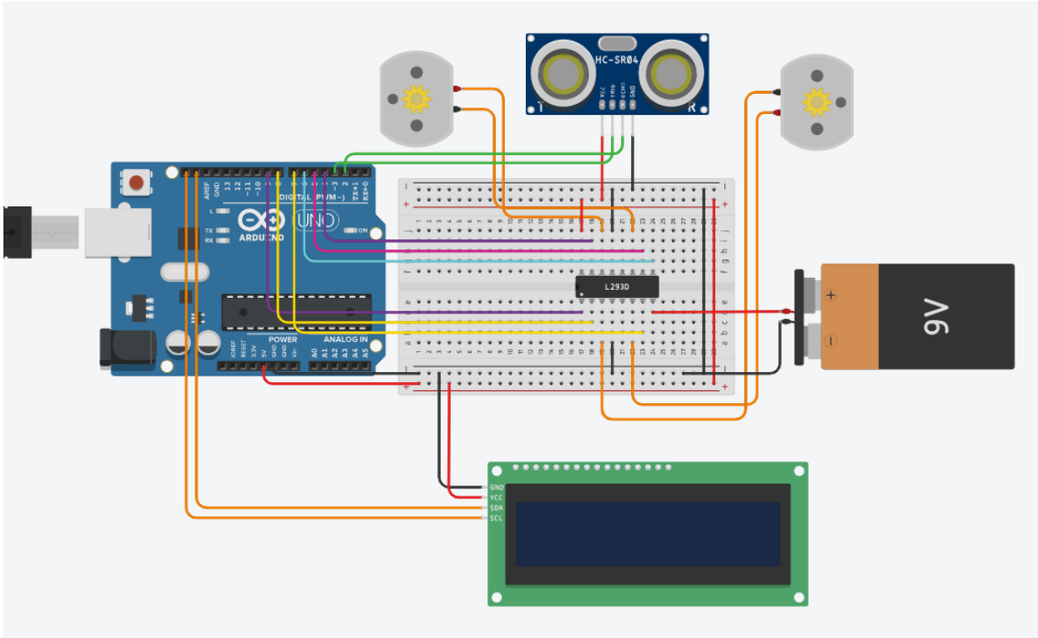

Figure 9. Circuit Diagram of Modified Automated Parking Gate with using LCD I2C

Actual Implementation:

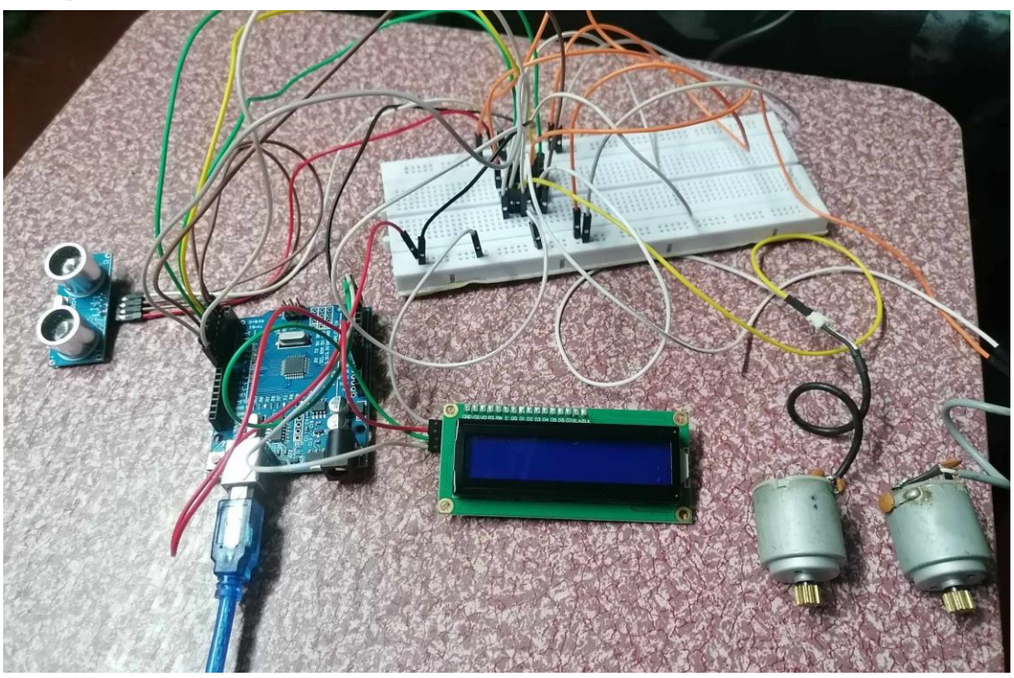

Figure 10. Actual Implementation of Modified Automated Parking Gate with using LCD I2C

Modified Code:

```
// DC motor control using ultrasonic sensor w/ LCD i2c

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin configurations
const int rightMotorEnablePin = 9;   // Right motor
const int rightMotorIn1 = 8;         // Right motor
const int rightMotorIn2 = 7;         // Right motor
const int leftMotorEnablePin = 6;    // Left motor
const int leftMotorIn3 = 5;          // Left motor
```

```
const int leftMotorIn4 = 4;         // Left motor
const int trigPin = 3;              // Ultrasonic distance sensor
const int echoPin = 2;              // Ultrasonic distance sensor
const int sdaPin = A5;              // LCD pin
const int sclPin = A4;              // LCD pin

int counter = 0; // Initialize to 0 if the gate is initially open
bool gateClosed = false; // Flag to track gate closure

// LCD configuration
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 columns and 2 rows

// Function declarations
void setup();
void openGate();
void closeGate();
void stopMotors();
void updateLCD(const char* message);
void loop();

void setup() {
  // Pin Setup
  pinMode(rightMotorEnablePin, OUTPUT);
  pinMode(rightMotorIn1, OUTPUT);
  pinMode(rightMotorIn2, OUTPUT);
  pinMode(leftMotorEnablePin, OUTPUT);
  pinMode(leftMotorIn3, OUTPUT);
  pinMode(leftMotorIn4, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // LCD Setup
  lcd.begin(16, 2);
  lcd.backlight();

  // Initial Motor States
  stopMotors();

  // Serial Communication Initialization
  Serial.begin(9600);
}

void openGate() {
  // Motor Control for Gate Opening
  analogWrite(rightMotorEnablePin, 255);
  digitalWrite(rightMotorIn1, HIGH);
  digitalWrite(rightMotorIn2, LOW);
  analogWrite(leftMotorEnablePin, 255);
  digitalWrite(leftMotorIn3, LOW);
  digitalWrite(leftMotorIn4, HIGH);
  delay(1000);
  counter = 0;
  gateClosed = false;
  updateLCD("Gate opened");
}

void closeGate() {
  // Motor Control for Gate Closing
  analogWrite(rightMotorEnablePin, 255);
  digitalWrite(rightMotorIn1, LOW);
  digitalWrite(rightMotorIn2, HIGH);
```

```
  analogWrite(leftMotorEnablePin, 255);
  digitalWrite(leftMotorIn3, HIGH);
  digitalWrite(leftMotorIn4, LOW);
  delay(1000);
  counter = 1;
  gateClosed = true;
  updateLCD("Gate closed");
}

void stopMotors() {
  // Stop Motors
  analogWrite(rightMotorEnablePin, 0);
  analogWrite(leftMotorEnablePin, 0);
}

void updateLCD(const char* message) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(message);
}

void loop() {
  // Ultrasonic Sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2; // Use integers for better accuracy

  // Display Distance
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance <= 10) {
    if (counter == 1) {
      openGate();
      Serial.println("Object detected - Gate opening");
    } else {
      stopMotors();
    }
  } else {
    if (gateClosed && counter == 0) {
      stopMotors();
    } else if (counter == 0) {
      closeGate();
      Serial.println("No object detected - Gate closing");
    } else {
      stopMotors();
    }
  }
  delay(500); // Adjust the delay for smoother operation
}
```

Tinkercad Simulation:

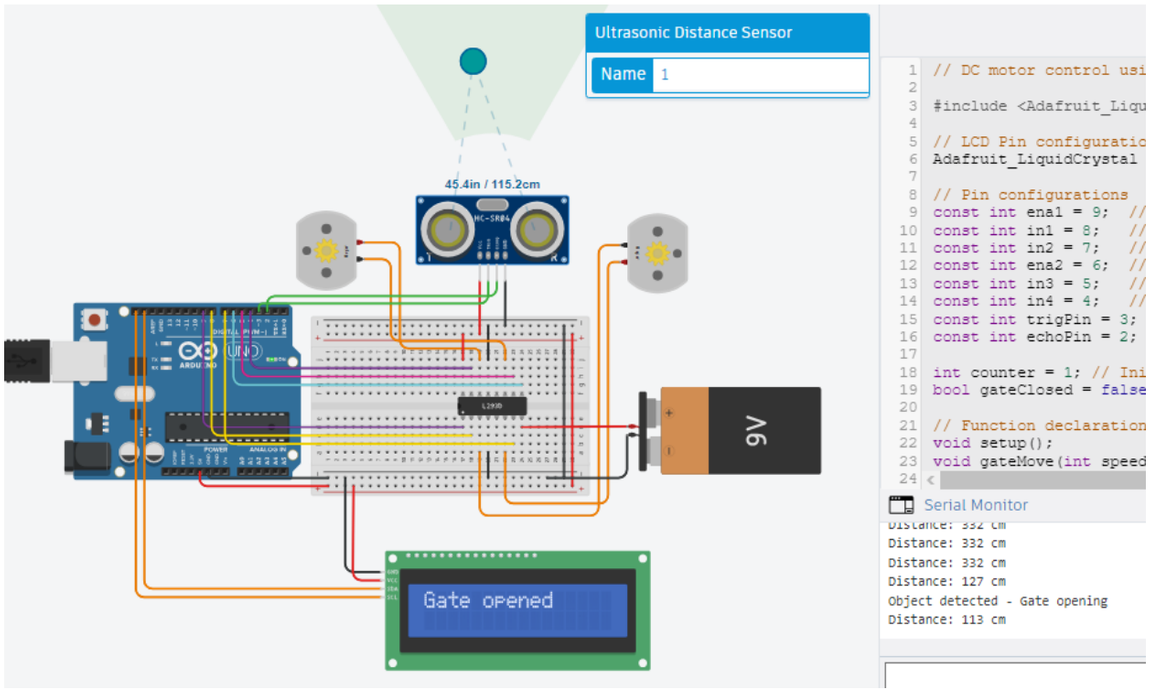

Figure 11. Simulation of the Parking Gates Opening when an Object is Within Range with using LCD I2C and Motor Rotation Stopping After Opening the Gates

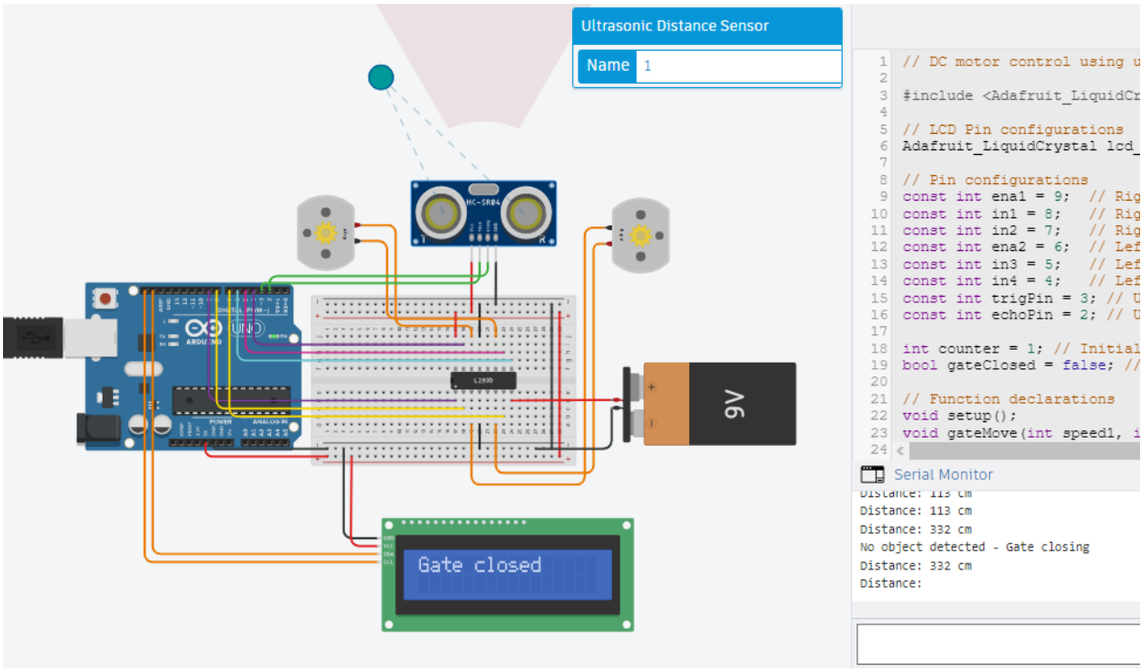

Figure 12. Simulation of the Parking Gates Closing when an Object is Not Within Range with using LCD I2C and Motor Rotation Stopping After Closing the Gates

V. Conclusion

In this laboratory activity, we successfully applied Arduino-based control for a gate system using DC motors and an ultrasonic distance sensor. The system is designed to automatically open and close the gate based on the proximity of an object detected by the ultrasonic sensor.

The system's key components include two DC motors responsible for opening and closing the gate, an ultrasonic sensor to detect the presence of an object in the gate's area, and an LCD for real-time feedback on the gate's status.

The modified code demonstrates effective motor control logic, utilizing pulse width modulation (PWM) for speed control and digital signals for direction control. The ultrasonic sensor measures the distance to an object, and the gate responds accordingly.

The LCD provides a user-friendly interface, updating the gate's status with "Gate opened" or "Gate closed." This feedback is crucial for monitoring the system's behavior and ensuring it operates as intended.

The code also includes appropriate delays to prevent rapid and unintended changes in motor states, contributing to the stability and reliability of the system.

Throughout the laboratory activity, we gained hands-on experience in interfacing various components with an Arduino microcontroller, understanding motor control mechanisms, and implementing a responsive and automated system using sensor input.

References

[1] Microdigisoft. (2023, October 22). DC Motor Control with Ultrasonic Sensors: An Arduino-Based Guide. microdigisoft.com - Tutorials and Projects. https://microdigisoft.com/control-a-motor-using-ultrasonic-distance-sensors/

[2] A.W.Azad. (2023, September 14). Ultrasonic sensor DC motor Arduino Code: important example - electrical hub. azadtechhub.com. https://azadtechhub.com/ultrasonic-sensor-dc-motor-arduino-code/