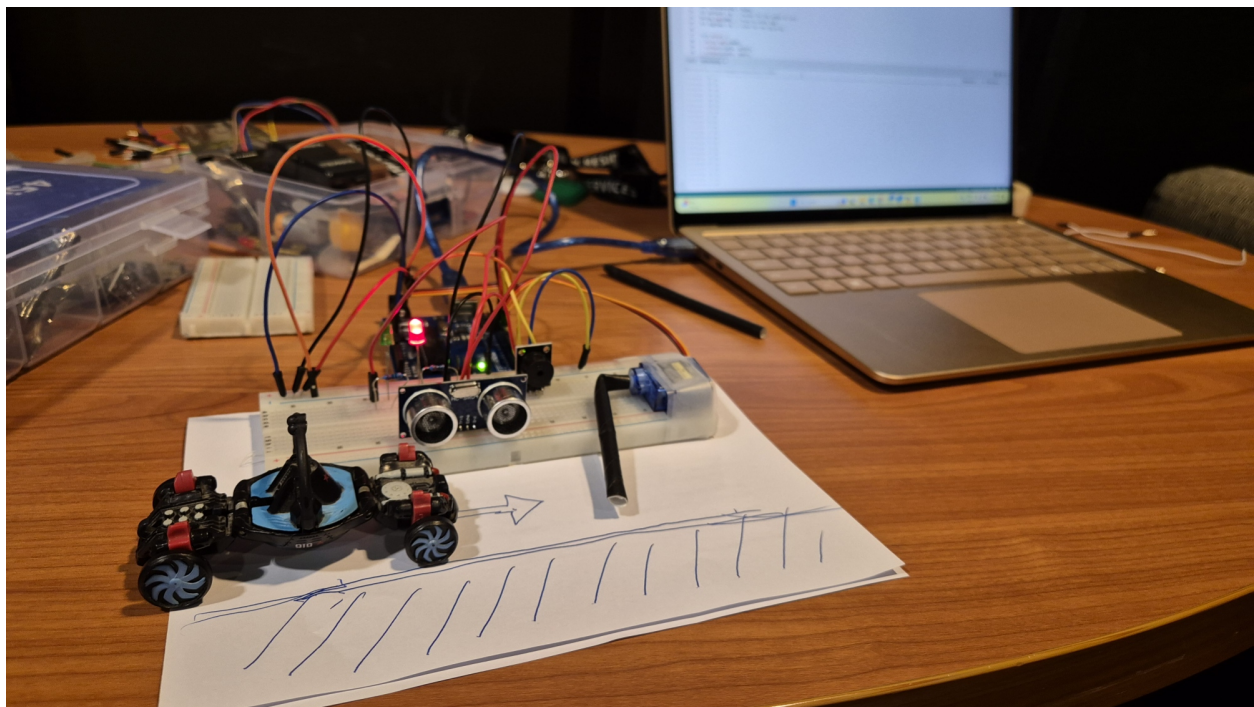


# SIT 111 3.8 HD

## Report - Automated Parking Toll System

The Automated Parking Toll System is a comprehensive project designed to manage vehicle access in a parking facility using an Arduino-based setup. This project went through several iterations to refine its functionality and ensure it met all the desired objectives. Here, I will document the project, detailing the components used, the system design, and the various stages of development.



## Project Overview and Objectives

The primary goal of this project was to create an automated system that could detect approaching vehicles, control a barrier to allow or deny access, and provide visual and auditory feedback. Additionally, the system needed to log the number of vehicles that passed through and interact with users through the Serial Monitor. The final design aimed to notify security personnel about incoming cars using a buzzer and represent traffic signals for the drivers using LEDs.

# Components and Initial Setup

The project utilized the following components:

- **Arduino Uno:** The microcontroller that serves as the brain of the system.
- **Ultrasonic Sensor (HC-SR04):** Detects the presence of vehicles by measuring distance.
- **Servo Motor:** Controls the barrier to allow or deny vehicle access.
- **Red and Green LEDs:** Represent traffic signals for the incoming cars.
- **Buzzer:** Notifies security personnel about incoming cars.
- **Resistors (220Ω):** Used with LEDs to limit current.
- **Jumper Wires and Breadboard:** For making connections between components.

The initial setup involved connecting the ultrasonic sensor, servo motor, LEDs, and buzzer to the Arduino. The ultrasonic sensor was used to detect the presence of vehicles by measuring the distance to an object. If the distance was below a certain threshold, the system considered a vehicle to be present.

## Iterations and Refinements

### First Iteration: Basic Detection and Barrier Control

In the first iteration, the focus was on basic vehicle detection and barrier control. The ultrasonic sensor was connected to the Arduino, and the servo motor was used to control the barrier. The system was programmed to lift the barrier when a vehicle was detected and lower it when the vehicle moved away. This iteration successfully demonstrated the core functionality of detecting a vehicle and controlling the barrier.

### Second Iteration: Adding Visual Indicators

The next step was to add visual indicators using LEDs. A red LED was used to indicate that the barrier was down, and a green LED was used to indicate that the barrier was lifted. This provided a clear visual signal to drivers about whether they could proceed. The LEDs were connected to the Arduino, and the code was updated to control the LEDs based on the barrier's position.

### Third Iteration: User Interaction and Logging

To enhance the system's interactivity, the third iteration introduced user interaction through the Serial Monitor. When a vehicle was detected, the system prompted the user (simulating security personnel) to grant or deny access by entering "yes" or "no". If "yes" was entered, the barrier was lifted, and the green LED was turned on. If "no" was entered, the barrier remained down, and the red LED stayed on. Additionally, the system logged each vehicle that passed through, storing the logs in an array.

#### **Fourth Iteration: Adding Auditory Notification**

In this iteration, a buzzer was added to notify security personnel about incoming cars. The buzzer sounded when a vehicle was detected, alerting the security personnel to check the Serial Monitor and grant or deny access. This added an important auditory element to the system, ensuring that security personnel were promptly notified of incoming vehicles.

#### **Final Iteration: Delay and Log Display**

The final iteration included a 2-second delay after the barrier was lifted to allow the car time to pass. This ensured that the barrier did not lower too quickly, providing a safe passage for the vehicle. Additionally, the system was updated to display the total car count and all log entries when the user entered "quit". This provided a comprehensive log of all vehicles that had passed through the barrier.

## **Arduino Code Sketch**

```
#include <Servo.h>

const int trigPin = 9;
const int echoPin = 10;
const int servoPin = 11;
const int redLEDPin = 12;
const int greenLEDPin = 13;
const int buzzerPin = 8;
const int distanceThreshold = 10; // Distance threshold in cm

Servo barrierServo;
bool accessGranted = false;
```

```

int carCount = 0; // Counter for the number of cars
String logs[100]; // Array to store logs
int logIndex = 0; // Index for the log array

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redLEDPin, OUTPUT);
  pinMode(greenLEDPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  barrierServo.attach(servoPin);
  barrierServo.write(0); // Start with the barrier down
  digitalWrite(redLEDPin, HIGH); // Red LED on initially
  digitalWrite(greenLEDPin, LOW); // Green LED off initially
  digitalWrite(buzzerPin, LOW); // Buzzer off initially

  Serial.println("Type quit to end counter for today.");
}

void loop() {
  long duration, distance;

  String input = Serial.readStringUntil('\n');
  if (input.equalsIgnoreCase("quit")) {
    Serial.print("Total cars passed: ");
    Serial.println(carCount);
    Serial.println("Log entries:");
    for (int i = 0; i < logIndex; i++) {
      Serial.println(logs[i]);
    }
    delay(2000);
    return;
  }

  // Send a 10us pulse to trigger the ultrasonic sensor

```

```

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Read the echo pin and calculate the distance
duration = pulseIn(echoPin, HIGH);
distance = (duration / 2) / 29.1; // Convert to cm

// Print the distance to the Serial Monitor
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");

// Check if the distance is less than the threshold
if (distance < distanceThreshold && !accessGranted) {
    digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
    Serial.println("Vehicle detected. Grant access? (yes/no/quit)");

    // Clear the serial buffer
    while (Serial.available() > 0) {
        Serial.read();
    }

    // Wait for user input with a timeout
    unsigned long startTime = millis();
    while (Serial.available() == 0) {
        if (millis() - startTime > 10000) { // 10 seconds timeout
            Serial.println("No response. Access denied.");
            digitalWrite(buzzerPin, LOW); // Turn off the buzzer
            delay(2000);
            return;
        }
    }
}

```

```

String input = Serial.readStringUntil('\n');
input.trim(); // Remove any trailing whitespace
if (input.equalsIgnoreCase("yes")) {
    accessGranted = true;
    barrierServo.write(90); // Lift the barrier
    digitalWrite(redLEDPin, LOW); // Turn off red LED
    digitalWrite(greenLEDPin, HIGH); // Turn on green LED
    carCount++; // Increment the car counter
    String logEntry = "Car passed at: " + String(millis()) + '
    logs[logIndex++] = logEntry; // Save log entry to array
    Serial.println("Access granted. Barrier lifted!");
    Serial.println(logEntry);
    delay(2000); // Allow time for the car to pass
} else if (input.equalsIgnoreCase("no")) {
    Serial.println("Access denied.");
    delay(2000);
}
digitalWrite(buzzerPin, LOW); // Turn off the buzzer
}

// If the vehicle moves away, reset the system
if (distance >= distanceThreshold && accessGranted) {
    accessGranted = false;
    barrierServo.write(0); // Lower the barrier
    digitalWrite(redLEDPin, HIGH); // Turn on red LED
    digitalWrite(greenLEDPin, LOW); // Turn off green LED
    Serial.println("Barrier lowered.");
}

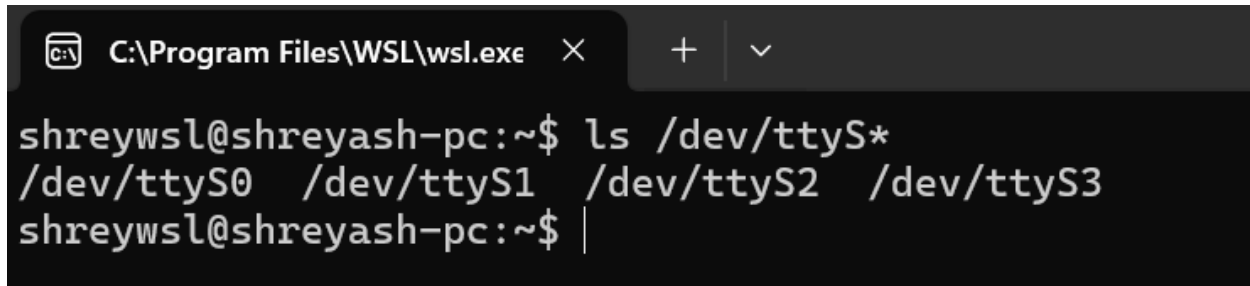
delay(500); // Wait for 500ms before the next measurement
}

```

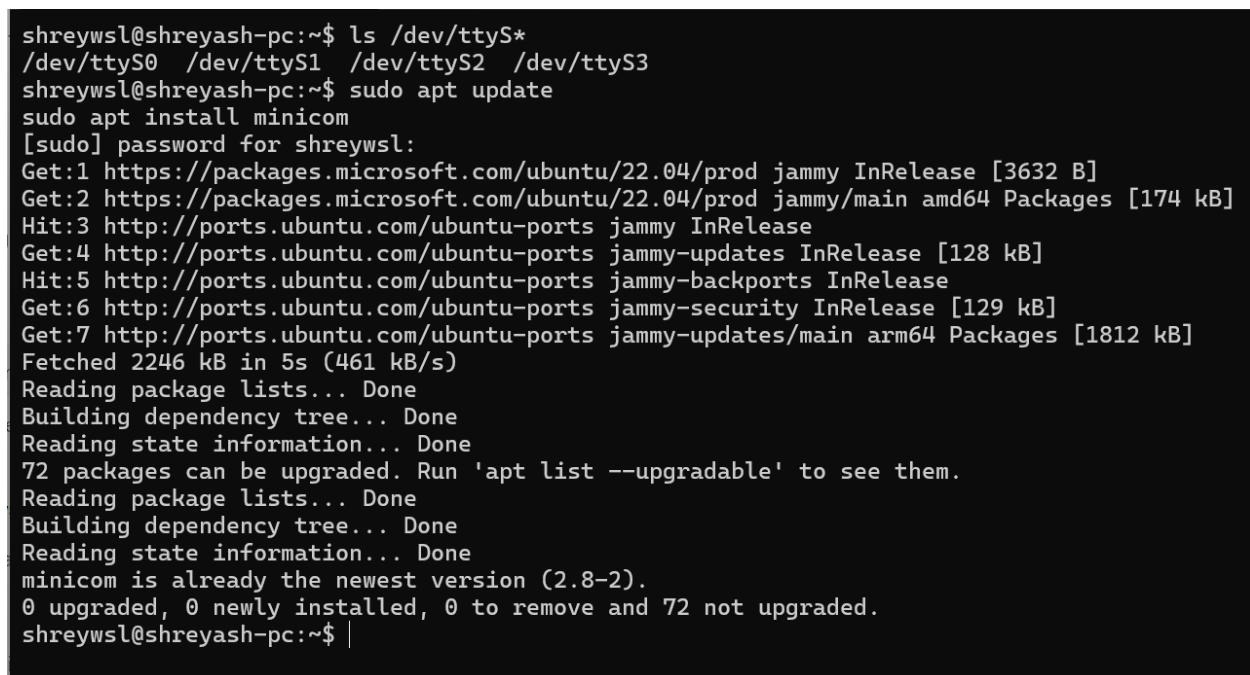
## Sending Linux commands via Terminal

I was able to set up a serial connection between by arduino and the ubuntu linux command. I have used minicom interface to communicate serially with the arduino.

### Finding correct port for connection

A terminal window titled 'C:\Program Files\WSL\wsl.exe' with a dark background. The prompt is 'shreywsl@shreyash-pc:~\$'. The command 'ls /dev/ttyS\*' is entered, and the output is '/dev/ttyS0 /dev/ttyS1 /dev/ttyS2 /dev/ttyS3'. The prompt is now 'shreywsl@shreyash-pc:~\$ |'.

### Installing Minicom

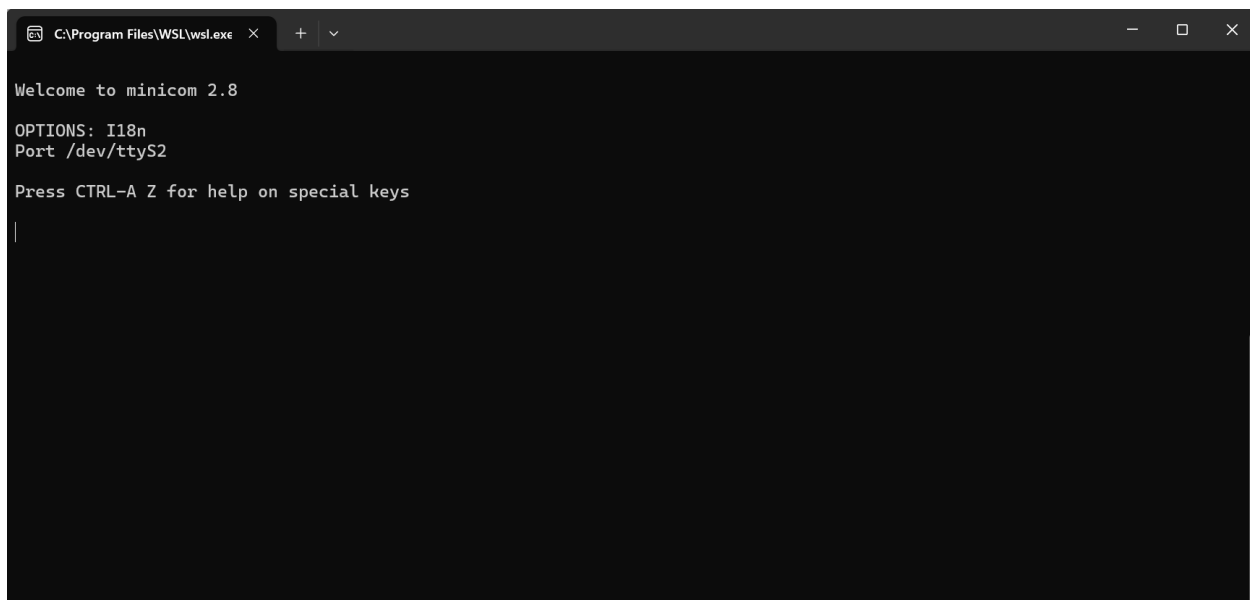
A terminal window with a dark background showing the installation of minicom. The prompt is 'shreywsl@shreyash-pc:~\$'. The command 'ls /dev/ttyS\*' is entered, and the output is '/dev/ttyS0 /dev/ttyS1 /dev/ttyS2 /dev/ttyS3'. The prompt is now 'shreywsl@shreyash-pc:~\$'. The command 'sudo apt update' is entered, and the output is 'sudo apt install minicom'. The prompt is now '[sudo] password for shreywsl:'. The command 'sudo apt install minicom' is entered, and the output is 'Get:1 https://packages.microsoft.com/ubuntu/22.04/prod jammy InRelease [3632 B] Get:2 https://packages.microsoft.com/ubuntu/22.04/prod jammy/main amd64 Packages [174 kB] Hit:3 http://ports.ubuntu.com/ubuntu-ports jammy InRelease Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [128 kB] Hit:5 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease Get:6 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [129 kB] Get:7 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 Packages [1812 kB] Fetched 2246 kB in 5s (461 kB/s) Reading package lists... Done Building dependency tree... Done Reading state information... Done 72 packages can be upgraded. Run 'apt list --upgradable' to see them. Reading package lists... Done Building dependency tree... Done Reading state information... Done minicom is already the newest version (2.8-2). 0 upgraded, 0 newly installed, 0 to remove and 72 not upgraded. shreywsl@shreyash-pc:~\$ |'.

### Configuring Minicom

I have ensured that the baud rate and serial port configuration in minicom match those used in the Arduino code.

```
+-----+
| A -   Serial Device       : /dev/ttyS2
| B - Lockfile Location    : /var/lock
| C -   Callin Program      :
| D -   Callout Program     :
| E -   Bps/Par/Bits        : 9600 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
| H -   RS485 Enable        : No
| I -   RS485 Rts On Send   : No
| J -   RS485 Rts After Send : No
| K -   RS485 Rx During Tx  : No
| L -   RS485 Terminate Bus : No
| M - RS485 Delay Rts Before: 0
| N - RS485 Delay Rts After : 0
|
| Change which setting? |
+-----+
```

## Using minicom to receive messages and send serial commands via terminal



```
C:\Program Files\WSL\wsl.exe
Welcome to minicom 2.8
OPTIONS: I18n
Port /dev/ttyS2
Press CTRL-A Z for help on special keys
|
```



```
Distance: 5 cm
Vehicle detected. Grant access? (yes/no/quit)
Access granted. Barrier lifted!
Car passed at: 56603 ms. Total cars: 1
Distance: 346 cm
Barrier lowered.
Distance: 346 cm
Distance: 346 cm
Distance: 346 cm
Distance: 4 cm
Vehicle detected. Grant access? (yes/no/quit)
Access granted. Barrier lifted!
Car passed at: 68652 ms. Total cars: 2
Distance: 346 cm
Barrier lowered.
Distance: 10 cm
Distance: 8 cm
Vehicle detected. Grant access? (yes/no/quit)
Access granted. Barrier lifted!
Car passed at: 78895 ms. Total cars: 3
Distance: 346 cm
Barrier lowered.
Distance: 346 cm
Distance: 346 cm
Total cars passed: 3
Log entries:
```

## Video link:

[https://youtu.be/Pv\\_plqxakyM](https://youtu.be/Pv_plqxakyM)

[https://youtu.be/Pv\\_plqxakyM](https://youtu.be/Pv_plqxakyM)