Smart Seatbelt Reminder System

Introduction

In an era where road safety is paramount, seatbelts stand as the most fundamental and effective protective measure in vehicles. Despite widespread awareness, instances of unbuckled occupants persist, leading to preventable injuries and fatalities in accidents. This project, the "Smart Seatbelt Reminder System," aims to address this critical issue by providing an automated, real-time visual alert to occupants who are seated but have not fastened their seatbelts. This system serves as a proactive safety mechanism, promoting better buckling habits and ultimately contributing to enhanced road safety.

Problem Statement

Human error and forgetfulness can lead to occupants failing to fasten their seatbelts before or during a journey. Current vehicle warning systems may sometimes be overlooked or activated too late. There is a need for a clear, immediate, and intuitive reminder system that specifically addresses the scenario where a seat is occupied, but the seatbelt is not engaged.

Solution Overview

The Smart Seatbelt Reminder System is an embedded system designed to monitor the occupancy status of a seat and the fastening status of its corresponding seatbelt. Utilizing an Arduino microcontroller as its central processing unit, the system integrates a custom-built Force Sensitive Resistor (FSR) to detect seat occupancy and a push-button to simulate the seatbelt buckle status. A Light Emitting Diode (LED) serves as the visual alert mechanism. The system's logic is straightforward: if a seat is detected as occupied AND the seatbelt is determined to be unfastened, the LED will illuminate, reminding the occupant to buckle up.

System Components

The project utilizes the following key components:

1. Arduino Uno Microcontroller:
    - **Function:** Serves as the brain of the system, processing sensor inputs and controlling the output LED. It's programmed to continuously read sensor data, apply the defined logic, and activate the alert as necessary.

2. DIY Force Sensitive Resistor (FSR):
    - **Function:** Detects the presence of an occupant in the seat.
    - **Innovation:** A custom FSR was built from sponge and aluminium foil due to commercial unavailability. Applied pressure changes its electrical resistance, which the Arduino's analogRead() function measures as a digital value proportional to the pressure.

3. Push Button:
    - **Function**: Simulates the seatbelt buckle's fastening/unfastening status.
    - **Connection**: The button is connected to digital Pin 2 with Arduino's internal INPUT_PULLUP resistor enabled. This means the pin reads HIGH when the button is not pressed, and LOW when pressed (connected to Ground).

4. Light Emitting Diode (LED):
    - **Function:** Provides the visual alert when the seatbelt is unfastened while the seat is occupied.
    - **Connection:** Connected to a digital output pin (Pin 9).
    - **Critical Safety Component:** A 220 Ohm resistor is connected in series with the LED. This resistor is crucial for limiting the current, preventing the LED from burning out due to excessive current from the Arduino's 5V output.

5. Connecting Wires:
    - For establishing electrical connections between components.

6. Breadboard:
    - For prototyping and temporary circuit assembly.

7. USB Cable:
    - For programming and powering the Arduino.

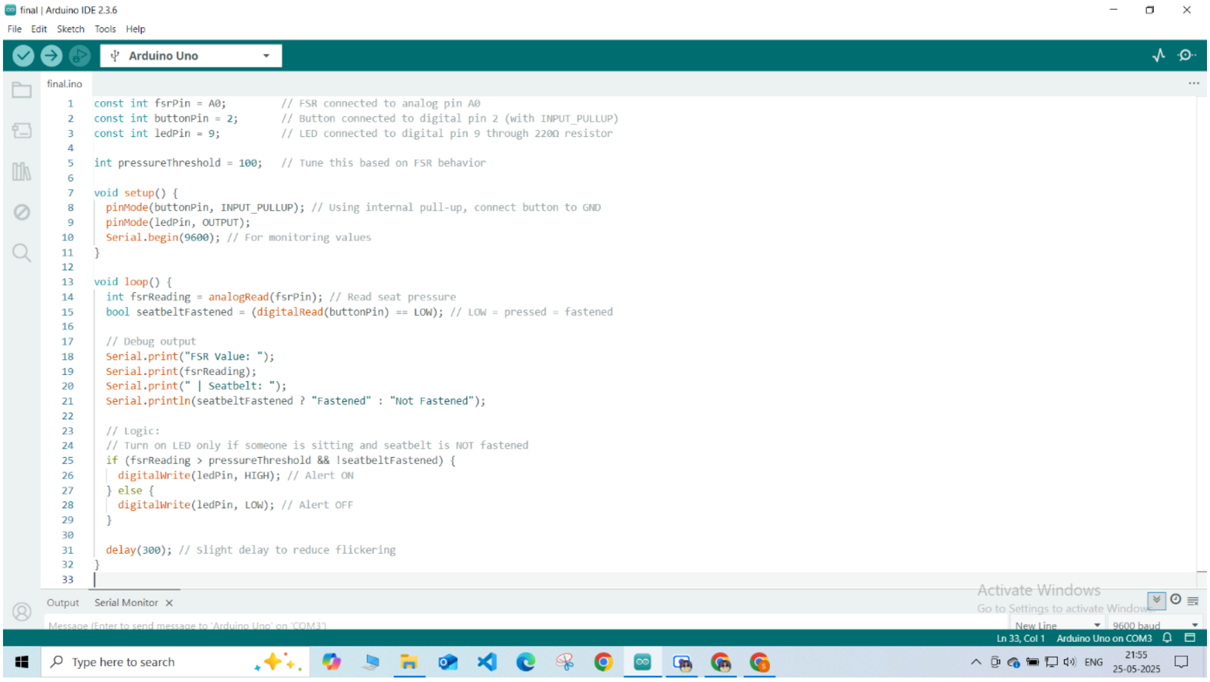
System Operation (Code Logic)

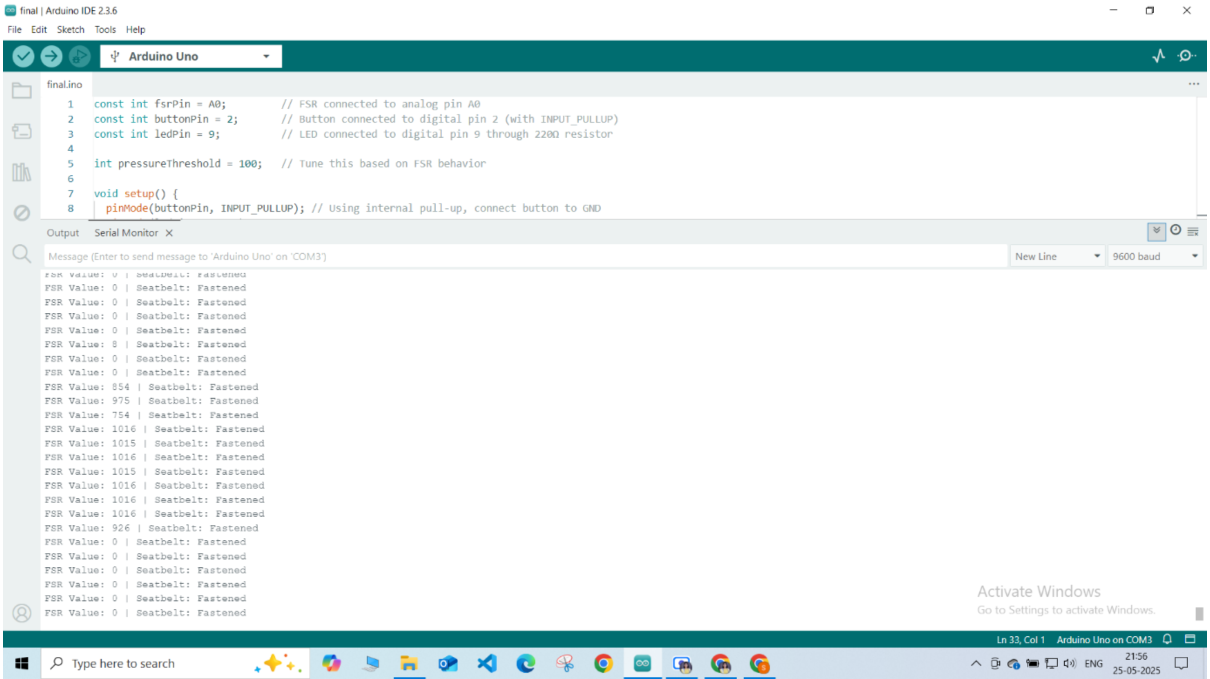
The system's behavior is defined by the following code uploaded to the Arduino:

```
const int fsrPin = A0;          // FSR connected to analog pin A0
const int buttonPin = 2;        // Button connected to digital pin
2
const int ledPin = 9;           // LED connected to digital pin 9
through 220Ω resistor

int pressureThreshold = 100;    // Adjust this value based on your
FSR reading
// int pressureThreshold = 10;  // for DIY FSR

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
// Using internal pull-up, so button connects to GND
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
// Open serial monitor to see FSR values and status
}

void loop() {
  int fsrReading = analogRead(fsrPin);
// Read FSR analog value, i.e, seat pressure
  bool seatbeltFastened = (digitalRead(buttonPin) == LOW);
// LOW = pressed = fastened

  // Debug output
  Serial.print("FSR Value: ");
  Serial.print(fsrReading);
  Serial.print(" | Seatbelt: ");
  Serial.println(seatbeltFastened ? "Fastened" : "Not Fastened");

  // Turn on LED if person is sitting and seatbelt is NOT fastened
  if (fsrReading > pressureThreshold && !seatbeltFastened) {
    digitalWrite(ledPin, HIGH); // Alert ON
  } else {
    digitalWrite(ledPin, LOW); // Alert OFF
  }

  delay(300); // Slight delay for stability(reduces flickering)
}
```

Code Explanation:

- **setup() Function:**
    - Initializes the buttonPin as an input with an internal pull-up resistor.
    - Sets the ledPin as an output.
    - Starts Serial.begin(9600) for debugging, allowing sensor values to be viewed on a computer.
- **loop() Function:**
    - **fsrReading = analogRead(fsrPin);**: Reads the analog value from the FSR. A higher value indicates more pressure.
    - **seatbeltFastened = (digitalRead(buttonPin) == LOW);**: Reads the state of the button. Since INPUT_PULLUP is used, LOW means the button is pressed (simulating a fastened seatbelt), and HIGH means it's not pressed (simulating an unfastened seatbelt).
    - **Serial.print(...)**: These lines are for debugging, displaying the current FSR value and seatbelt status on the Serial Monitor.
    - **if (fsrReading > pressureThreshold && !seatbeltFastened)**: This is the core decision-making logic:
        - fsrReading > pressureThreshold: Checks if the detected pressure from the FSR exceeds a predefined threshold, indicating a person is sitting.
        - !seatbeltFastened: Checks if the seatbeltFastened variable is false (i.e., the button is *not* pressed, indicating the seatbelt is unfastened).
        - If both conditions are true, the system proceeds to the next line.
    - **digitalWrite(ledPin, HIGH);**: If both conditions in the if statement are met, the LED connected to ledPin is turned ON, providing the visual alert.
    - **else { digitalWrite(ledPin, LOW); }**: If either the seat is not occupied or the seatbelt is fastened, the LED is turned OFF.
    - **delay(300);**: A brief pause is introduced to ensure stable readings and prevent rapid flickering of the LED.

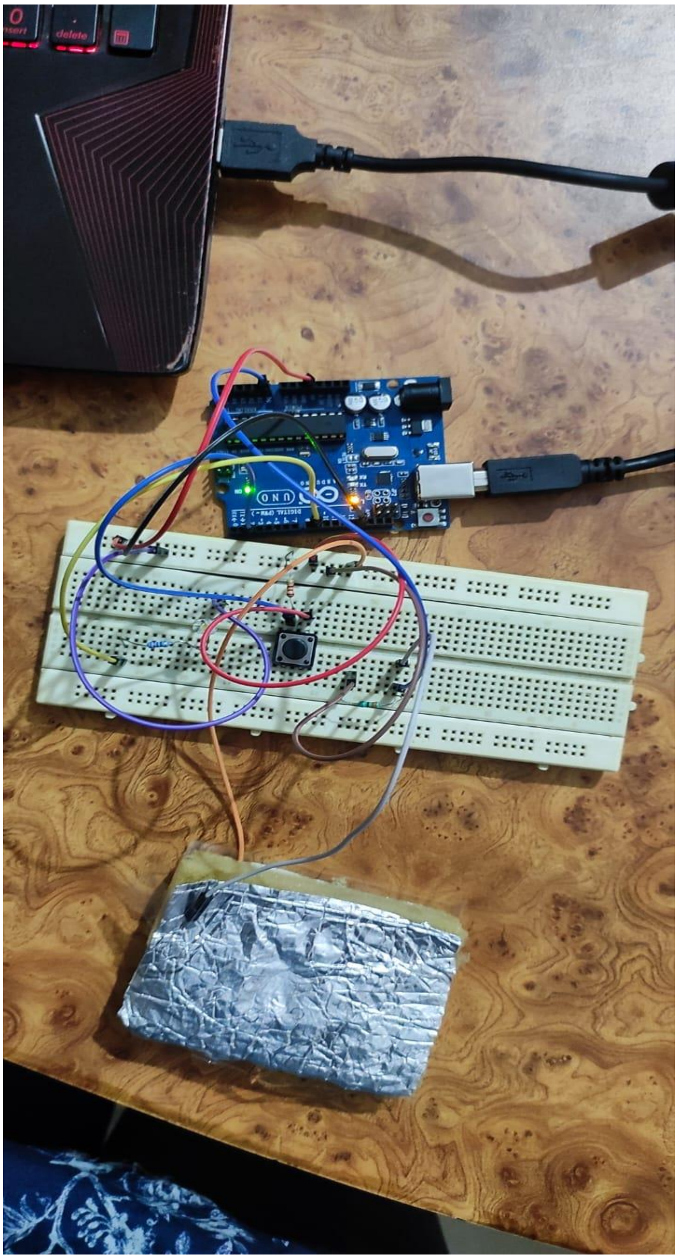

final | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Uno
final.ino
const int fsrPin = A0; // FSR connected to analog pin A0
const int buttonPin = 2; // Button connected to digital pin 2 (with INPUT_PULLUP)
const int ledPin = 9; // LED connected to digital pin 9 through 220Ω resistor
int pressureThreshold = 100; // Tune this based on FSR behavior
void setup() {
pinMode(buttonPin, INPUT_PULLUP); // Using internal pull-up, connect button to GND
pinMode(ledPin, OUTPUT);
Serial.begin(9600); // For monitoring values
}
void loop() {
int fsrReading = analogRead(fsrPin); // Read seat pressure
bool seatbeltFastened = (digitalRead(buttonPin) == LOW); // LOW = pressed = fastened
// Debug output
Serial.print("FSR Value: ");
Serial.print(fsrReading);
Serial.print(" | Seatbelt: ");
Serial.println(seatbeltFastened ? "Fastened" : "Not Fastened");
// Logic:
// Turn on LED only if someone is sitting and seatbelt is NOT fastened
if (fsrReading > pressureThreshold && !seatbeltFastened) {
digitalWrite(ledPin, HIGH); // Alert ON
} else {
digitalWrite(ledPin, LOW); // Alert OFF
}
delay(300); // Slight delay to reduce flickering
}
Output
Serial Monitor
Activate Windows
Go to Settings to activate Windows.
Ln 33, Col 1 Arduino Uno on COM3
Type here to search
ENG
21:55
25-05-2025


final | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Uno
final.ino
const int fsrPin = A0; // FSR connected to analog pin A0
const int buttonPin = 2; // Button connected to digital pin 2 (with INPUT_PULLUP)
const int ledPin = 9; // LED connected to digital pin 9 through 220Ω resistor
int pressureThreshold = 100; // Tune this based on FSR behavior
void setup() {
pinMode(buttonPin, INPUT_PULLUP); // Using internal pull-up, connect button to GND
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM3')
New Line
9600 baud
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 8 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 854 | Seatbelt: Fastened
FSR Value: 975 | Seatbelt: Fastened
FSR Value: 754 | Seatbelt: Fastened
FSR Value: 1016 | Seatbelt: Fastened
FSR Value: 1015 | Seatbelt: Fastened
FSR Value: 1016 | Seatbelt: Fastened
FSR Value: 1015 | Seatbelt: Fastened
FSR Value: 1016 | Seatbelt: Fastened
FSR Value: 1016 | Seatbelt: Fastened
FSR Value: 1016 | Seatbelt: Fastened
FSR Value: 926 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
FSR Value: 0 | Seatbelt: Fastened
Activate Windows
Go to Settings to activate Windows.
Ln 33, Col 1 Arduino Uno on COM3
Type here to search
ENG
21:56
25-05-2025

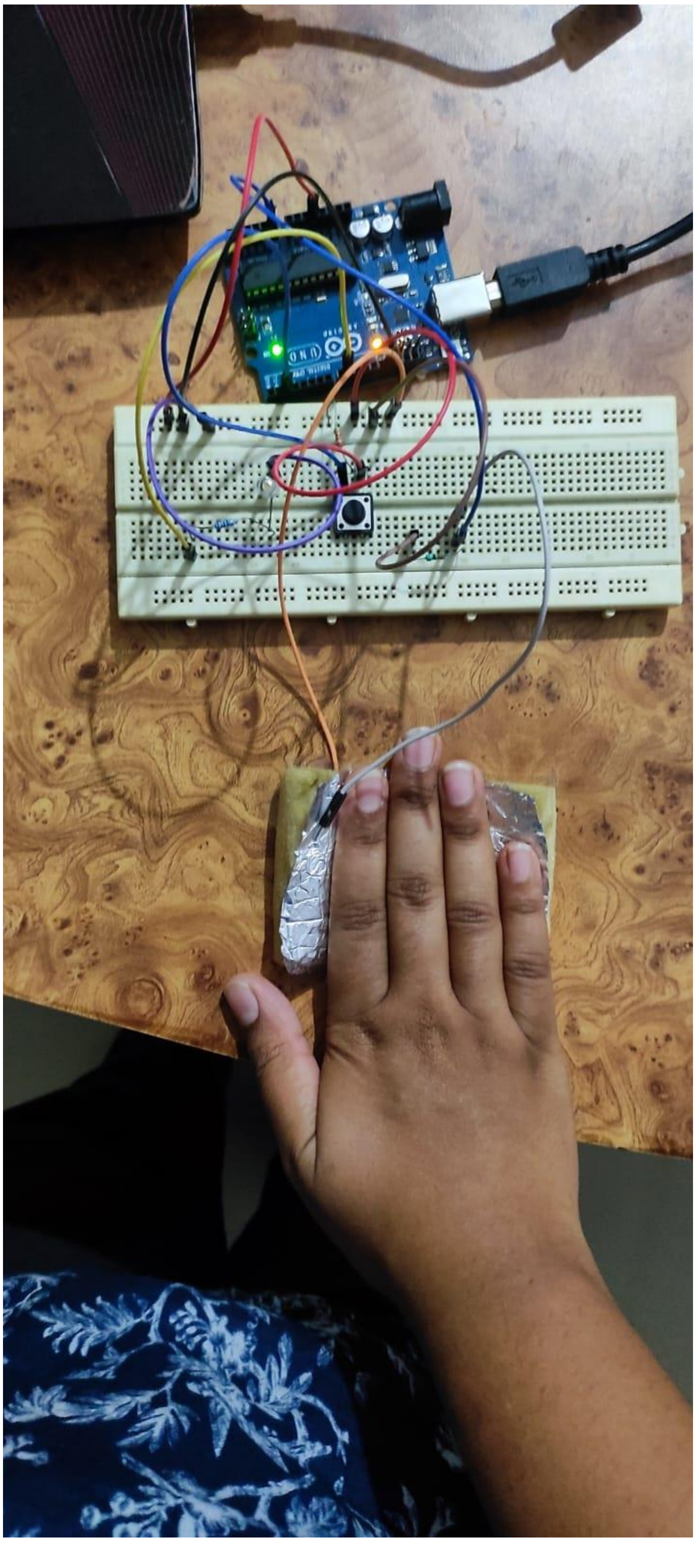
Practical Demonstration Scenarios

When the system is operational, the following behaviors would be observed:

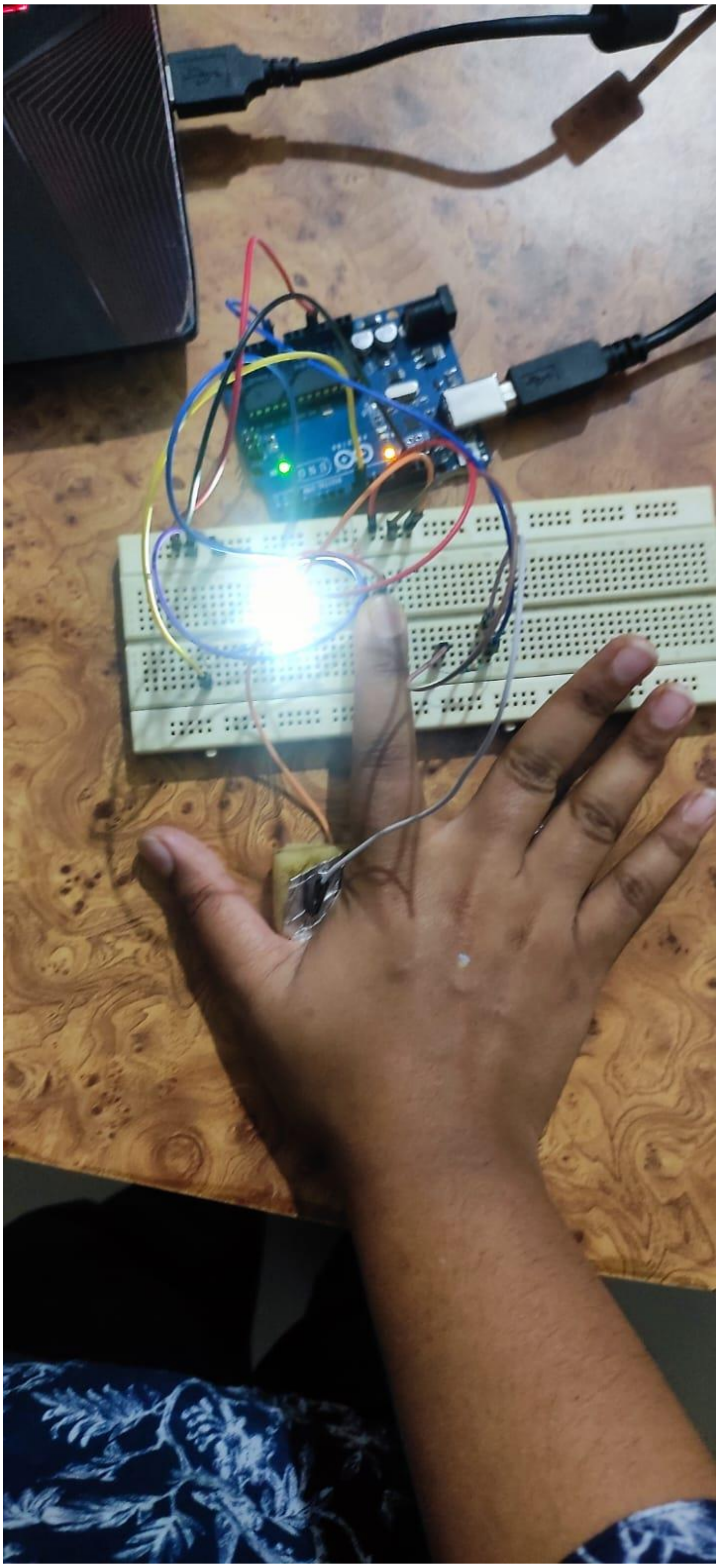
1. **Empty Seat & Seatbelt Fastened (Default State):**
    - FSR Value: Low (below pressureThreshold)
    - Seatbelt State: Fastened (button is LOW or not pressed)
    - LED: OFF



2. **Occupied Seat & Seatbelt Fastened:**
    - FSR Value: High (above pressureThreshold)
    - Seatbelt State: Fastened (button is LOW or not pressed)
    - LED: OFF



3. **Occupied Seat & Seatbelt Unfastened (Alert Condition):**
    - FSR Value: High (above pressureThreshold)
    - Seatbelt State: Unfastened (button is HIGH or pressed)
    - LED: **ON** (Alerting the occupant to buckle up)



Challenges Faced & Lessons Learned

Developing this project presented several valuable learning opportunities:

- **DIY FSR Fabrication and Calibration:** Designing and building a functional FSR from scratch required experimentation with materials (sponge, aluminum foil) and careful calibration of the pressureThreshold value to ensure accurate and consistent seat occupancy detection. This highlighted the importance of empirical testing in sensor integration.
- **Current Limiting for LEDs:** Initially, connecting the LED directly to the Arduino resulted in an overcurrent warning. This experience underscored the critical necessity of using a current-limiting resistor (like the 220 Ohm resistor) to protect sensitive components and ensure their longevity. This is a fundamental concept in basic electronics.
- **Arduino IDE Upload Issues:** Encountering "Access is denied" errors with COM ports during code upload taught me essential troubleshooting steps, such as closing the Serial Monitor, unplugging/replugging the board, checking port selections, and understanding driver implications. These are common hurdles for beginners in physical computing.
- **Resourcefulness:** The inability to procure a commercial FSR locally pushed me to think creatively and utilize readily available materials, demonstrating the importance of adaptability in project development.

Future Enhancements

While this prototype effectively demonstrates the core concept, several enhancements could be implemented in future iterations:

- **Audible Alert:** Integrating a buzzer to provide an auditory warning in addition to the visual LED alert.
- **Improved Seatbelt Sensor:** Replacing the simple button with a more robust sensor, such as a magnetic reed switch or a physical latch sensor, for more realistic seatbelt detection.
- **Multi-Seat Monitoring:** Expanding the system to monitor multiple passenger seats, each with its own FSR and seatbelt sensor.
- **Integration with Vehicle Systems:** Exploring potential integration with actual vehicle systems for more advanced warning displays or data logging.
- **Power Management:** Implementing power-saving modes for battery-powered applications.

Conclusion

The Smart Seatbelt Reminder System is a practical and effective demonstration of how basic electronics and programming can be applied to enhance real-world safety. Through the development of this project, I gained hands-on experience with microcontrollers, sensor integration, circuit design principles (especially the crucial role of resistors), and problem-solving through troubleshooting. The successful creation of a DIY FSR further showcased adaptability and resourcefulness in overcoming component availability challenges. This project serves as a foundational step towards more sophisticated safety systems and highlights the impactful potential of embedded electronics in everyday applications.