

CHAPTER THREE



METHODOLOGY



3.1 Preamble


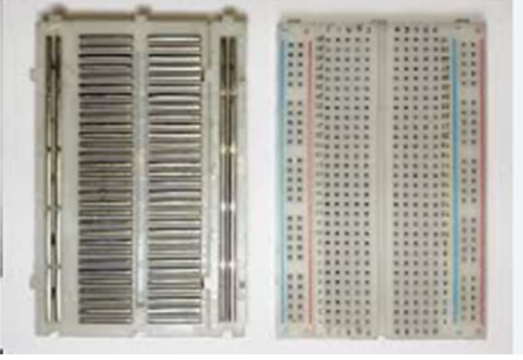

This research project was designed and implemented in two phases: hardware development and system simulation. The methodology includes sourcing components locally in Ilorin, Kwara State, Nigeria, assembling the circuit, programming the Arduino, and testing the system in a simulated environment.

3.2 Materials and Their Sources

Table 3.1 List of Materials and their sources

Component	Specification	Source in Ilorin
Arduino Uno	ATmega328P microcontroller board 	Kwaratech Electronics, Tanke Junction
Ultrasonic Sensors (2x)	HC-SR04, range: 2cm – 400cm 	JKK Electronics, Challenge Bookshop Complex

Servo Motors (2x)	SG90 Micro Servo 	TechCity Hub, Unity Road
LEDs (Red & Green)	5mm, 2V 	Ilorin Electronics Market, Taiwo Isale
Buzzer	5V Piezoelectric Buzzer 	JustElectronics, GRA Junction
Jumper Wires	Male-to-Male, Male-to-Female	Unity Road Market

		
Breadboard	830-point 	Kwaratech Electronics, Tanke Junction
Resistors	220 Ω , 330 Ω 	JKK Electronics
Power Supply	9V Battery or 5V USB Adapter	TechCity Hub or Spar Mall

		
Laptop with Arduino IDE	For programming and simulation 	University of Ilorin ICT Lab / Personal Laptop

3.3 Research Procedure

Step 1: Component Acquisition

Electronic components were identified based on compatibility with Arduino. Vendors across Ilorin (Tanke, Taiwo, Unity Road) provided reliable access to affordable and tested hardware.

Step 2: Circuit Design and Assembly

- i. A schematic was drawn with Arduino Uno as the main controller.
- ii. The entry ultrasonic sensor was placed before the crossing, and the exit sensor after it, to detect train approach and departure.

- iii. Two servo motors controlled mock barrier gates.
- iv. LEDs represented traffic lights (Red and Green).
- v. A buzzer provided audible warnings.

All connections were made on a breadboard using jumper wires, with current-limiting resistors for the LEDs.

Step 3: Arduino Programming (C/C++)

- i. Programming was done using Arduino IDE in C/C++ language. The logic flow followed:
- ii. When the entry sensor detects a train (distance < 30 cm):
- iii. Buzzer is activated.
- iv. Red light turns ON.
- v. Servo motor lowers the gate.
- vi. When the exit sensor detects the train has left:
- vii. Buzzer turns OFF.
- viii. Green light turns ON.
- ix. Servo motor raises the gate.

Sample Code Snippet:

```
#include <Servo.h>
```

```
Servo gate;
```

```
const int triggerPin = 9;
```

```
const int echoPin = 10;
```

```
const int redLed = 6;

const int greenLed = 7;

const int buzzer = 5;

void setup() {

    pinMode(triggerPin, OUTPUT);

    pinMode(echoPin, INPUT);

    pinMode(redLed, OUTPUT);

    pinMode(greenLed, OUTPUT);

    pinMode(buzzer, OUTPUT);

    gate.attach(3); // Servo connected to pin 3

    Serial.begin(9600);

}

void loop() {

    long duration;

    int distance;

    digitalWrite(triggerPin, LOW);

    delayMicroseconds(2);

    digitalWrite(triggerPin, HIGH);

    delayMicroseconds(10);

    digitalWrite(triggerPin, LOW);
```

```

duration = pulseIn(echoPin, HIGH);

distance = duration * 0.034 / 2;

if (distance < 30) {

    digitalWrite(redLed, HIGH);

    digitalWrite(greenLed, LOW);

    digitalWrite(buzzer, HIGH);

    gate.write(0); // Close gate

} else {

    digitalWrite(redLed, LOW);

    digitalWrite(greenLed, HIGH);

    digitalWrite(buzzer, LOW);

    gate.write(90); // Open gate

}

delay(100);

}

```

Step 4: System Testing and Calibration

- i. Sensors were tested for accurate readings under different object distances.
- ii. Trigger distances were calibrated to detect model trains at 20–30 cm.
- iii. The servo motor's closed and open angles were set to 0° and 90°, respectively.
- iv. The buzzer's timing was adjusted to avoid unnecessary noise post-departure.

Step 5: Simulation on Tinkercad

- i. Autodesk Tinkercad was used to simulate the entire circuit virtually.
- ii. Components were placed in a simulated environment.
- iii. The same Arduino code was uploaded to test barrier operations and sensor responses.
- iv. Real-time debugging was done using the Serial Monitor and visual observations.

Step 6: Evaluation and Troubleshooting

- i. The system was tested repeatedly for consistency.
- ii. Any false triggers from sensors were addressed by: Adjusting sensor alignment, adding debounce delays in the code, using software averaging for distance measurements.

Step 7: Documentation and Analysis

- i. All findings and behaviors were documented.
- ii. Observations confirmed that the system met its intended objectives.
- iii. Identified areas for improvement (e.g., sensor sensitivity to light and sound interference).