University of Ghana

Computer Engineering Department

PIR Motion Sensor Light Control with STM32H743

Project Repository: https://github.com/AWESOME04/PIR-Motion-Sensor-Light-Control-with-STM32H743

GROUP 11

EVANS ACHEAMPONG – 10987644

EDWARD AYIREBI ACQUAH- 10986982

AMOAH OFORI DARKWAH- 10949533

BANI BERES ETORNAM – 10948391

EYRAM AHETO - 10987509

Contents

Abstract

This guide outlines the process of interfacing an **HC-SR501 Pyroelectric PIR infrared motion sensor** with an **STM32H743 Nucleo-144** to enable motion detection capabilities in projects. The HC-SR501 sensor detects temperature changes caused by movement and sends signals to the STM32. The STM32 then interprets these signals, determining if motion has occurred and triggering corresponding actions, such as activating lights or sending notifications. The hardware setup involves connecting the sensor to the STM32 using specified pins and configuring the Arduino IDE with the necessary libraries. Code snippets are provided to initialise serial communication, read sensor output, and display motion status on the serial monitor. The tutorial also emphasises the versatility of this setup for various applications including home automation, security systems, and robotics. Additionally, a schematic diagram is provided for easy reference in making the necessary connections.

Keywords: HC-SR501, PIR infrared motion sensor, STM32H743 Nucleo-144, Motion detection, Temperature changes

Introduction

Interfacing an HC-SR501 Pyroelectric PIR infrared motion sensor with an STM32 is a powerful way to add motion detection capabilities to your projects. The HC-SR501 PIR motion sensor is a small, low-cost device that utilizes infrared technology to detect changes in temperature caused by the movement of people or objects in front of the sensor. The sensor can detect motion within a certain range and angle, and sends a signal to the microcontroller indicating whether motion has been detected or not.

The STM32 then reads this signal and performs the necessary logic to determine if motion has been detected, and this information can be used to trigger an action such as turning on a light or sending a notification. The status of the motion detection can also be posted on the serial monitor for debugging or for displaying the status. This process allows for the creation of interactive projects and systems that can react to movement in the environment, such as home automation, security systems, and robotics.

Hardware Components

You will require the following hardware for Interfacing PIR Motion Sensor with Arduino.

S.no	Component	Value	Qty
1.	STM32H743 Nucleo-144	–	1
2.	USB Cable Type A to B	–	1
3.	PIR Motion Sensor	HC-SR501	1
4.	LED	–	1
5.	Jumper Wires	–	1
6.	Resistor	_	1

Motion Sensor with STM32H743

1. Connect the HC-SR501 PIR motion sensor to the STM32 using the VCC, GND, and OUT pins specified in the sensor's documentation.
2. In the Arduino IDE, include the necessary libraries for the HC-SR501 sensor.

```
#include <Arduino.h>
```

3. In the setup() function, initialize the serial communication and the pin mode for the sensor:

```
Serial.begin(9600);

pinMode(sensorPin, INPUT);
```

4. In the loop() function, read the sensor's output value:

```
int sensorValue = digitalRead(sensorPin);
```

5. Use an if statement to check if motion is detected by checking if the sensor value is high and post the status on the serial monitor using the Serial.println() function:
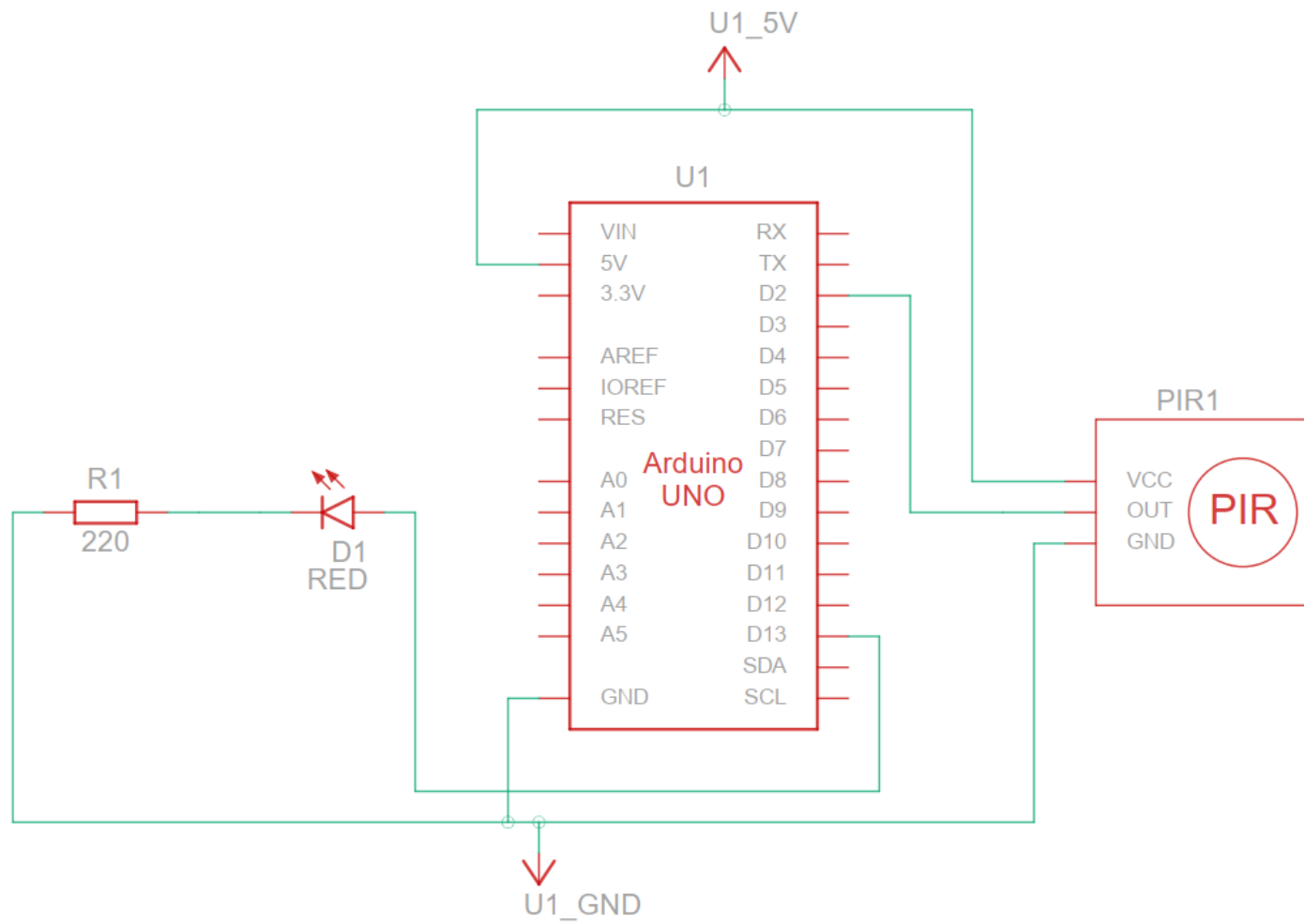
```
if(sensorValue == HIGH){

  Serial.println("Motion detected");

}else{

  Serial.println("No motion detected");
```

6. You can add any other action you want to take when motion is detected.
7. Upload the code to the Arduino Uno and open the serial monitor to see the sensor's output values.

Schematic Diagram

Make connections according to the circuit diagram given below.
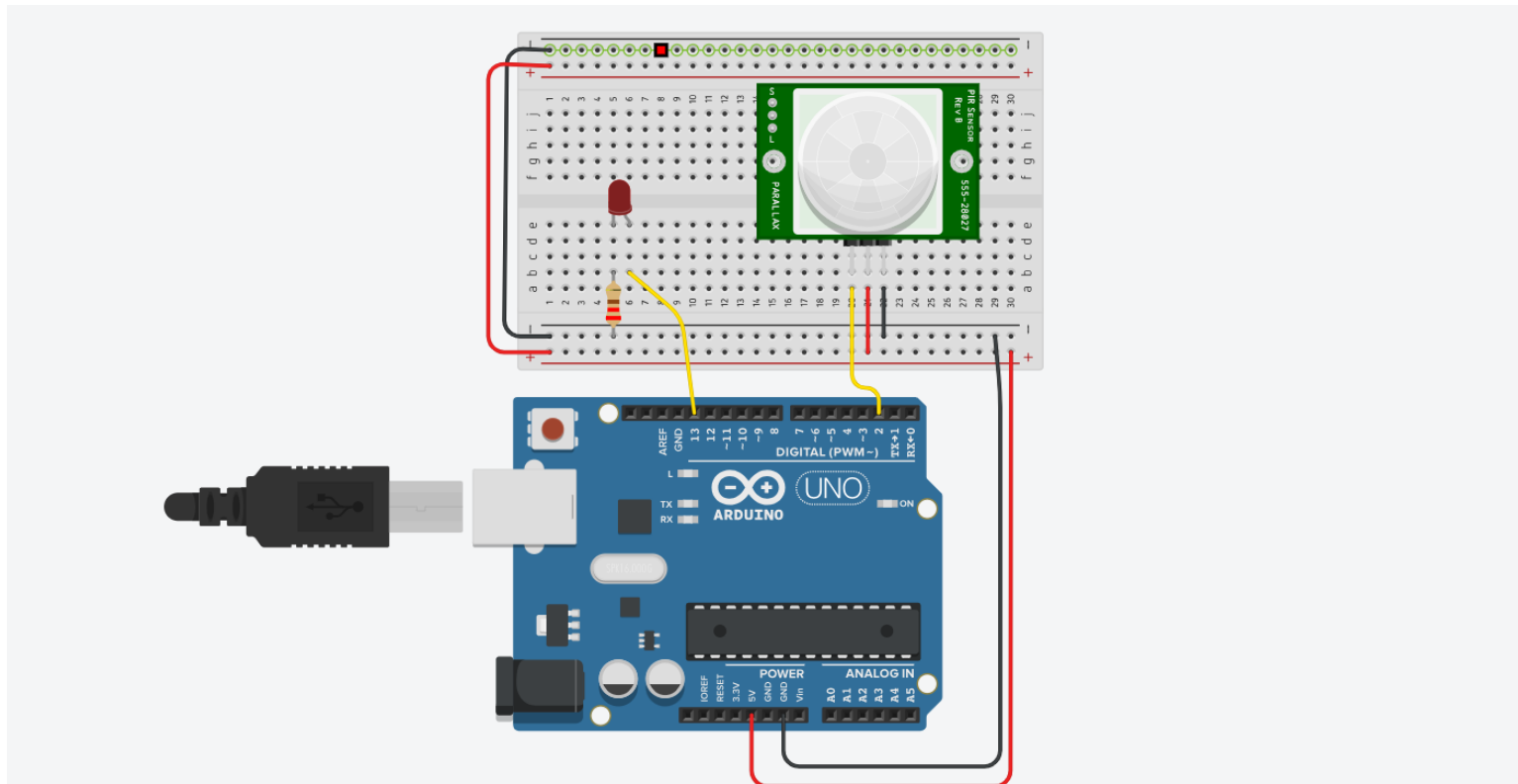
Wiring / Connections

STM32	PIR Motion Sensor
5V	VCC
GND	GND
D2	OUT

Installing Arduino IDE

First, you need to install Arduino IDE Software from its official website Arduino. Here is a simple step-by-step guide on "How to install Arduino IDE".

Arduino Code

```cpp
#include <Arduino.h>

#define PIR_PIN PA1

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Configure PIR pin as input
  pinMode(PIR_PIN, INPUT);

  // Configure LED pin as output
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  // Read PIR sensor output
  int pirState = digitalRead(PIR_PIN);

  if (pirState == HIGH) {
    Serial.println("Motion Detected");
    digitalWrite(LED_BUILTIN, HIGH); // Turn on the LED
  } else {
    Serial.println("No Motion Detected");
    digitalWrite(LED_BUILTIN, LOW); // Turn off the LED
  }

  delay(1000); // Adjust delay as needed
```

Serial Monitor Output


```
12    // Configure LED pin as output
13    pinMode(LED_BUILTIN, OUTPUT);
14  }
```

Output   Serial Monitor X

Message (Enter to send message to 'Nucleo-144' on 'COM10')

Motion detected!
Motion detected!
No motion detected.
No motion detected.
Motion detected!
Motion detected!
Motion detected!

Code for Python Script

```python
import serial
import pyttsx3
import time

ser = serial.Serial('COM10', 9600)
# text-to-speech engine
engine = pyttsx3.init()

while True:
    # Read data from serial port
    data = ser.readline().decode().strip()

    # Print received data for debugging
    print("Received data:", data)

    # Speak the received message
    engine.say(data)
    engine.runAndWait()
```
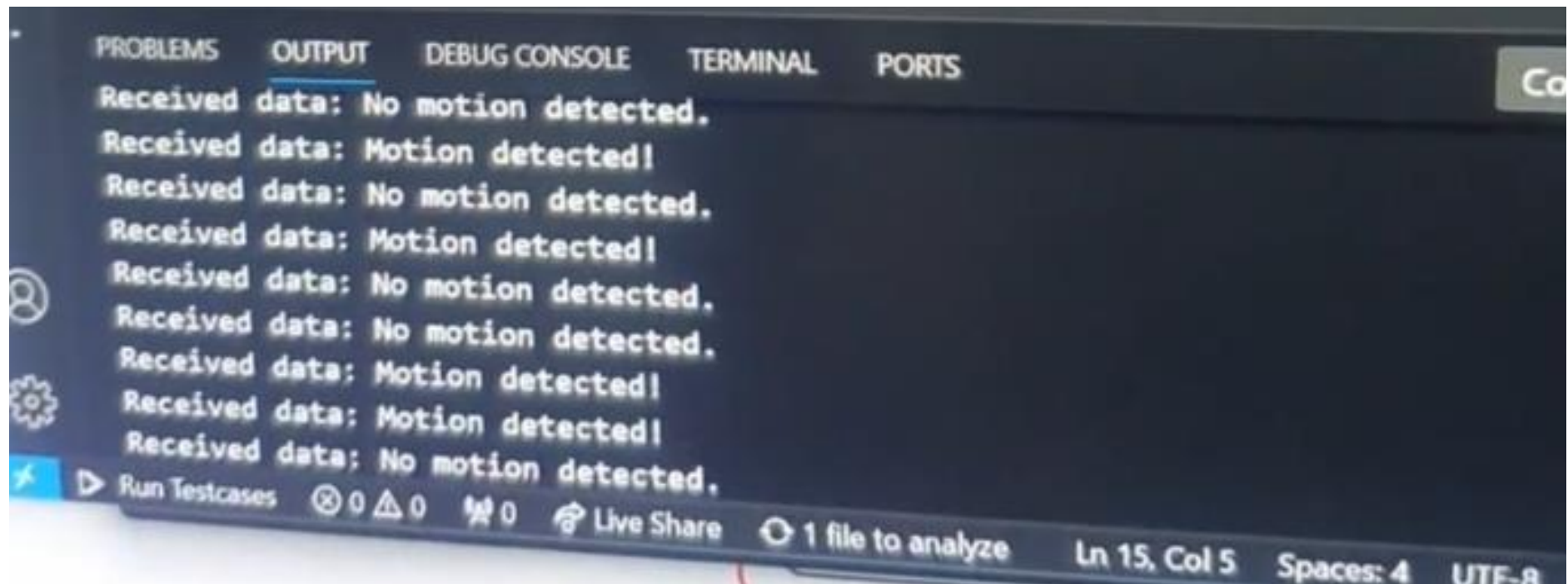
This Python script demonstrates a basic implementation for communicating with an external device via a serial port and converting received text data into speech using the pyttsx3 library. Here's a brief explanation of each part:

1. **Importing Libraries**: The script imports the necessary libraries: `serial` for serial communication and `pyttsx3` for text-to-speech conversion.

2. **Serial Port Configuration**: The `serial.Serial()` function initializes a serial connection. It takes two arguments: the port name (`COM10` in this case) and the baud rate (`9600`).

3. **Text-to-Speech Engine Initialization**: The `pyttsx3.init()` function initializes the text-to-speech engine.

4. **Main Loop (Infinite Loop):** The `while True:` loop runs indefinitely, continuously listening for data from the serial port.

5. **Reading Data**: Inside the loop, `ser.readline()` reads a line of text from the serial port. `.decode()` converts the received bytes to a string, and `.strip()` removes any leading or trailing whitespace.

6. **Printing Data (for Debugging)**: The received data is printed to the console for debugging purposes.

7. **Text-to-Speech Conversion**: The `engine.say()` function converts the received text data to speech. The `engine.runAndWait()` function ensures that the speech is spoken synchronously, blocking further execution until the speech is complete.

Python Script Output


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
Received data: No motion detected.
Received data: Motion detected!
Received data: No motion detected.
Received data: Motion detected!
Received data: No motion detected.
Received data: No motion detected.
Received data: Motion detected!
Received data: Motion detected!
Received data: No motion detected.
```

Working Explanation

In the Arduino IDE, the necessary libraries for the HC-SR501 sensor are included. In the setup() function, the serial communication and the pin mode for the sensor is initialized. The baud rate is set to 9600 bps.

In the loop() function, the sensor's output value is read using the digitalRead() function, which returns either a HIGH or LOW value. The value is then compared to check if motion is detected by using an if statement. If the sensorValue is HIGH, it means motion is detected and the message "Motion detected" is sent to the serial monitor. If the sensorValue is LOW, it means no motion is detected and the message "No motion detected" is sent to the serial monitor.
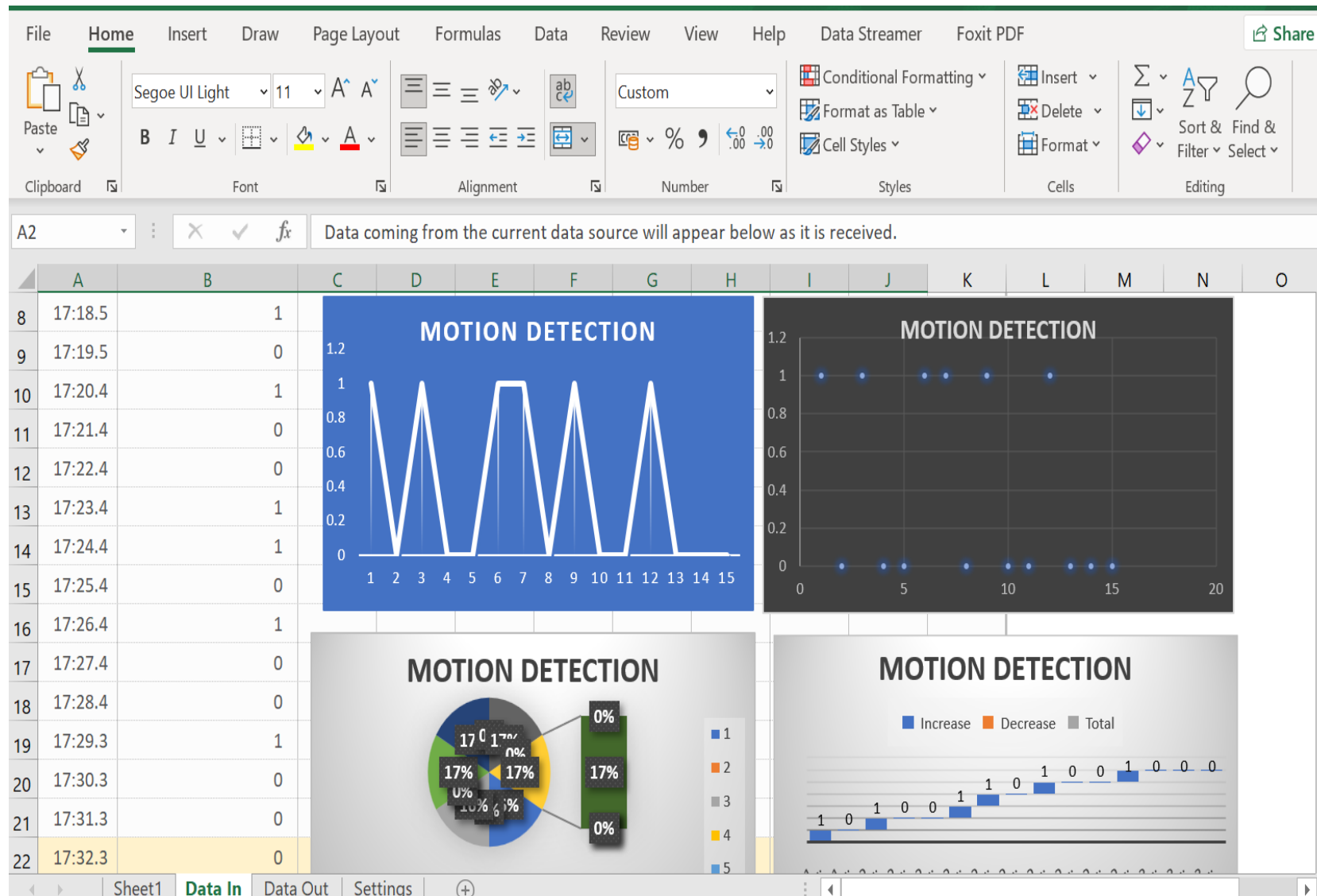
Applications

- Security Systems
- Robotics
- Industrial Automation
- Automotive
- Medical equipment
- Proximity Detection
- Gaming and interactive projects

Final Results
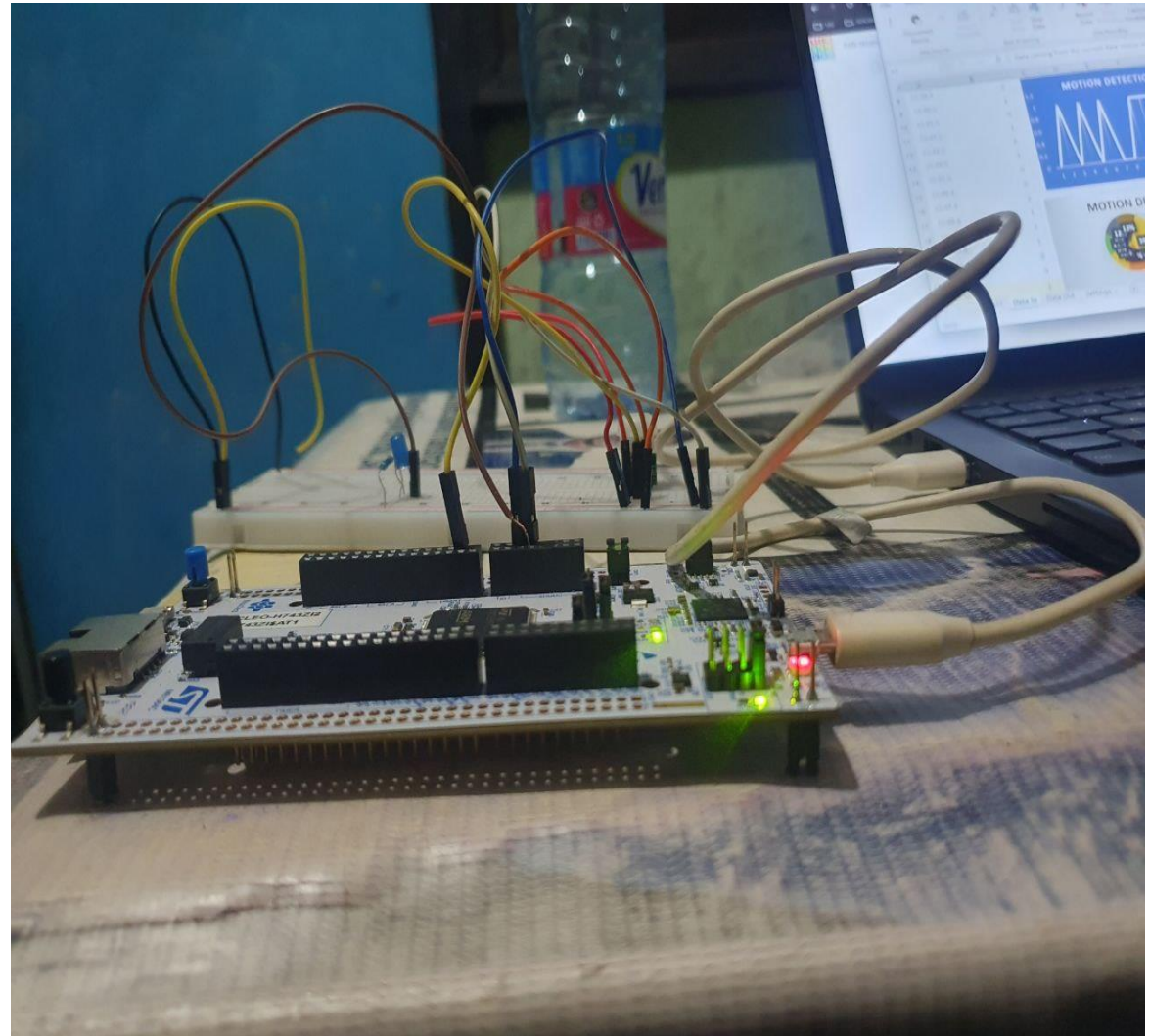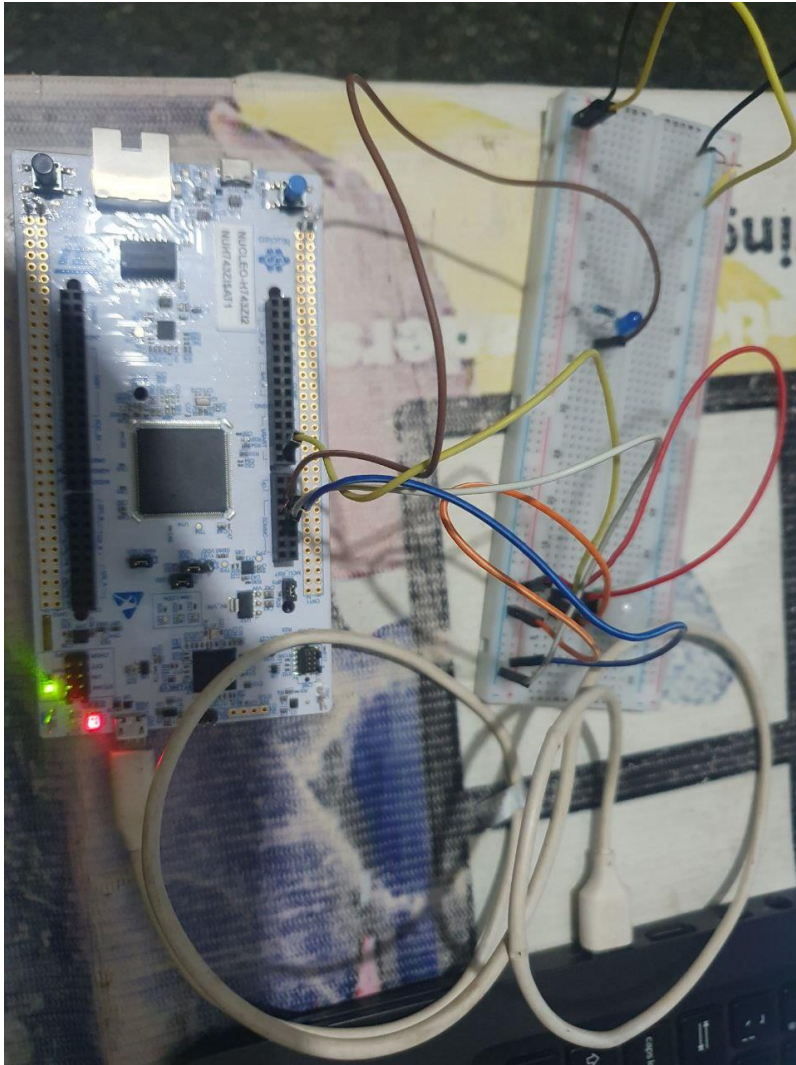
Data Visualization in Excel Using Data Streamer

Conclusion

Interfacing the HC-SR501 Pyroelectric PIR infrared motion sensor with an STM32 offers a cost-effective and efficient solution for implementing motion detection capabilities in projects. By following the outlined steps, users can easily set up the hardware and write the necessary code to detect motion and trigger actions accordingly. This versatile system can be applied across a wide range of applications, from simple home automation tasks to more complex security systems and robotics projects. With its ability to detect motion within a specified range and angle, the HC-SR501 sensor provides a reliable solution for enhancing the interactivity and functionality of various projects. By leveraging the capabilities of the Arduino platform, users can create responsive and intelligent systems that adapt to changes in their environment, making this a valuable addition to any DIY enthusiast or hobbyist's toolkit.

References

[1] https://arduinogetstarted.com/tutorials/arduino-motion-sensor

[2] https://circuitdigest.com/microcontroller-projects/interface-pir-sensor-with-arduino

[3] https://www.instructables.com/How-to-Use-a-PIR-Motion-Sensor-With-Arduino/

[4] https://www.circuits-diy.com/pir-motion-sensor-arduino-tutorial/

[5] https://www.phippselectronics.com/controlling-arduino-by-voice-with-python/