Motion Detection System Using Python

By
Akhil R Nair

Under the guidance of
Dr Chanthini Bhasker

FOSSEE Project
Indian Institute of Technology Bombay

June 2021

Contents

List of Figures

List of Tables

List of Arduino Code

List of Python Code

List of Acronyms

ACM	Abstract Control Model
ADC	Analog to Digital Converter
ADK	Accessory Development Kit
ALU	Arithmetic and Logic Unit
ARM	Advanced RISC Machines
BIOS	Basic Input/ Output System
CD	Compact Disc
CNES	National Centre for Space Studies
COM Port	Communication Port
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DC	Direct Current
DIY	Do It Yourself
DVD	Digital Versatile Disc
EEPROM	Electronically Erasable Programmable Read-Only Memory
FPGA	Field-programmable Gate Array
GNU	GNU's Not Unix
GPS	Global Positioning System
GPL	General Public License
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
LAPACK	Linear Algebra Package
LCD	Liquid Crystal Display
LDR	Light Dependent Resistor
LED	Light Emitting Diode

MRI	Magnetic Resonance Imaging
MISO	Master Input, Slave output
MOSI	Master out, Slave input
NTC	Negative Temperature Coefficient
OGP	Open Graphics Project
OS	Operating System
OSHW	Open Source Hardware
PCB	Printed Circuit Board
PTC	Positive Temperature Coefficient
PWM	Pulse width modulation
RAM	Random-access Memory
ROM	Read Only Memory
RS	Recommended Standard
RTC	Real Time Clock
Rx	Receiver
SD Card	Secure Digital Card
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCL	Tool Command Language
Tx	Transmitter
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Chapter 1

Interfacing Passive Infrared Sensor-PIR

The PIR sensor is responsible for detecting the change in infrared radiation levels when an intruder or human is passed through the system or space where it is arranged. Depending on the change in radiation levels the change in voltages occurs and then with this voltage the signal is amplified and hence the sound will be produced. Thus, it is helpful in various applications and areas. This type of system has many advantages compared to the existing system.

When motion is detected, it is displayed on the screen and in the next phase of this project, when motion is detected, it is displayed in an LCD.

1.1 Preliminary

This system can be broadly divided into 2 small parts namely: Motion detection using PIR sensor, Motion detection using PIR sensor and displaying on LCD.

- **Motion detection using PIR sensor**: - In this part a PIR sensor is used to detect motion. PIR becomes HIGH (gives output 1) when motion is detected and becomes LOW when no motion is detected or when the motion ends. So in this part using Arduino and python, we will receive output "No Motion" initially until any motion is detected. The output "Motion detected!" will be displayed when we the sensor encounters motion and the output "Motion ended!" will be displayed when the motion ends. The Ground pin of PIR sensor is connected to GND of Arduino, The VCC of PIR sensor is connected to 5V and the data pin is connected to digital pin 6 of Arduino.

- **Motion detection using PIR sensor and displaying on LCD**: - In this

part PIR and LCD is used. Working is same as previous part. But the output will be displayed in the LCD than in the monitor. The connections of PIR is same as previous part.

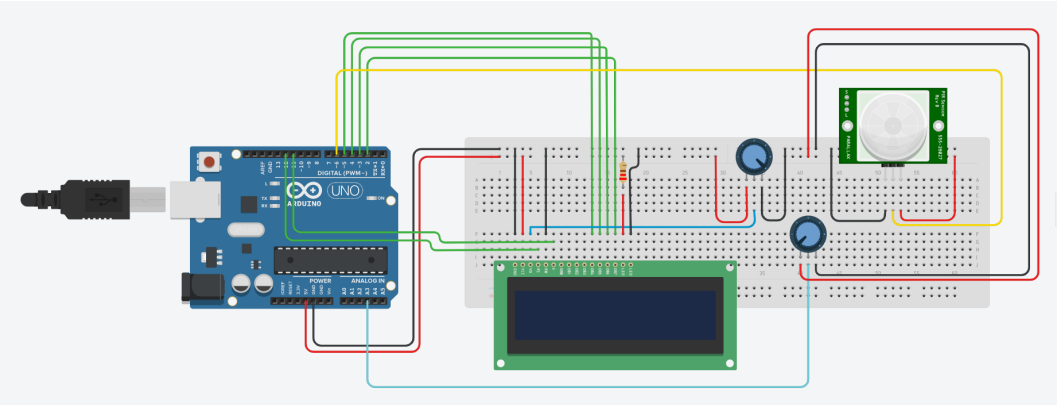


Figure 1.1: Proposed system for motion detection for PIR Interfacing

1.2 PIR Sensor HC-SR501

A passive infrared sensor is an electronic sensor that measures infrared light radiating from objects. PIR sensors mostly used in PIR-based motion detectors. PIR sensor is specially designed to detect such levels of infrared radiation. It basically consists of two main parts: A Pyroelectric Sensor and A special lens called Fresnel lens which focuses the infrared signals onto the pyroelectric sensor. A Pyroelectric Sensor actually has two rectangular slots in it made of a material that allows the infrared radiation to pass. Behind these, are two separate infrared sensor electrodes, one responsible for producing a positive output and the other a negative output. The reason for that is that we are looking for a change in IR levels and not ambient IR levels. The two electrodes are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

When the sensor is idle, i.e. there is no movement around the sensor; both slots detect the same amount of infrared radiation, resulting in a zero output signal. But when a warm body like a human or animal passes by; it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. The corresponding pulse of signals results in the sensor setting its output pin high.

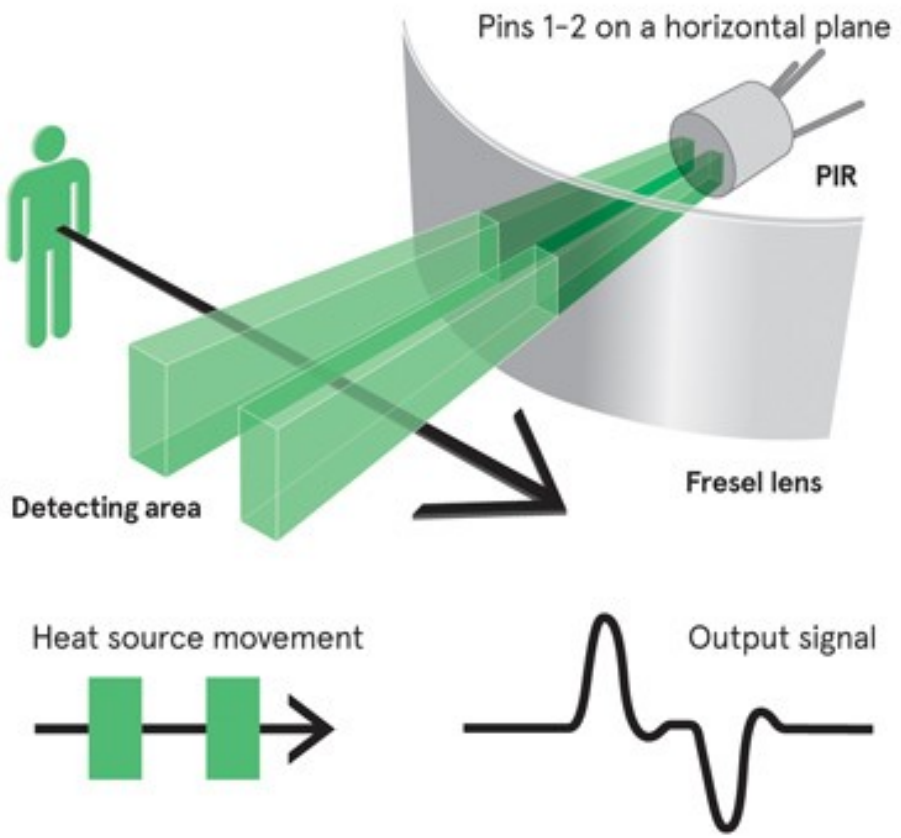


Figure 1.2: Pyroelectric Sensor

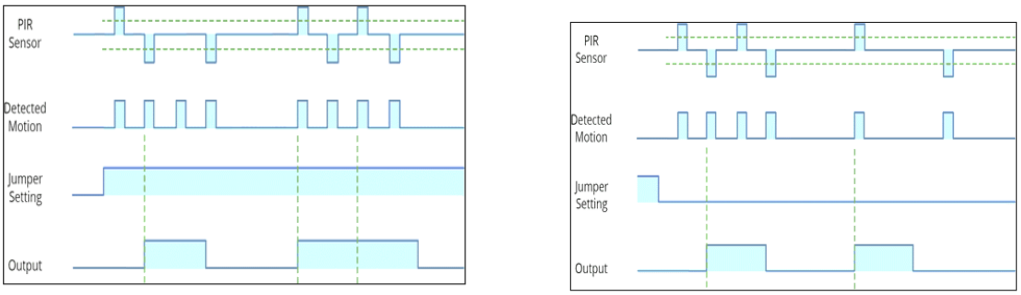
There are two potentiometers on the board to adjust a couple of parameters:

- **Sensitivity**– This sets the maximum distance that motion can be detected. It ranges from 3 meters to approximately 7 meters. The topology of your room can affect the actual range you achieve.
- **Time**– This sets how long that the output will remain HIGH after detection. At minimum it is 3 seconds, at maximum it is 300 seconds or 5 minutes.

It has two settings:

- H– This is the Hold/Repeat/Retriggering. In this position the HC-SR501 will continue to output a HIGH signal as long as it continues to detect movement.

- L– This is the Intermittent or No-Repeat/Non- Retriggering In this position the output will stay HIGH for the period set by the TIME potentiometer adjustment.



(a) No-Repeat/Non- Retriggering charactiristics

(b) Symbolic representation of an LDR

Figure 1.3: Settings of PIR Sensor

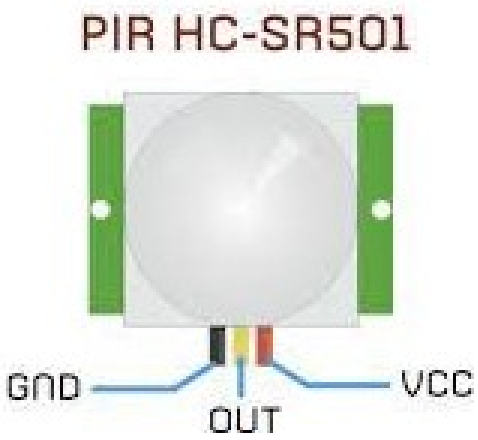


Figure 1.4: Pinout of PIR sensor

1.2.1 PIR sensor PINOUT

- **VCC** is the power supply for HC-SR501 PIR sensor which we connect the 5V pin on the Arduino.
- **Output** pin is a 3.3V TTL logic output. LOW indicates no motion is detected, HIGH means some motion has been detected.

- **GND** should be connected to the ground of Arduino.

1.3 Interfacing the PIR through the Arduino IDE

1.3.1 Interfacing the PIR

In this section, we shall describe how to read the voltage values from a HC-SR501 PIR Sensor connected to the digital pin 6 of the Arduino Uno board. The HC-SR501 PIR Sensor has to be connected to the Arduino Uno board before doing these experiments and the Arduino Uno needs to be connected to the computer with a USB cable, as shown in Fig 1.1.

1. Read the digital value of PIR sensor as mentioned above using digitalRead function.

```
1 void loop(){
```

2. The value received by digitalRead will be 0/1 (HIGH/LOW) depending upon whether motion is detected or not detected by PIR sensor. The received value is printed in serial monitor.

```
1         Serial.println("Motion detected!"); // print on output
      change
```

3. Delay is added to slow down the rate of output being printed in serial monitor.

```
1         delay(500);
```

   The functions are written inside void loop and the operations will keep on working till the Arduino code executes.

1.3.2 Arduino Code

Arduino Code 1.1 Read and display the PIR values. Available at `Origin/user-code/pir/arduino/pir/pir.ino`.

```
1 int inputPin = 6;                 // input pin for pir sensor
2 int pirState = 0;                // we start, assuming no motion detected
3 int val = 0;                      // variable for reading the pin status
4
5 void setup() {
6   pinMode(ledPin, OUTPUT);
7   pinMode(inputPin, INPUT);
8
9   Serial.begin(9600);
```

```
10 }
11
12 void loop(){
13   val = digitalRead(inputPin);  // read input value
14
15   if (val == HIGH)  // check if the input is HIGH
16   {
17       Serial.println("Motion detected!"); // print on output change
18       pirState = 1;
19       delay(500);
20   }
21   else if(val == LOW && pirstste==1)
22   {
23       Serial.println("Motion ended!");  // print on output change
24       delay(500);
25   }
26   else if(val == LOW && pirstste==0)
27   {
28       Serial.println("No Motion!");  // print on output change
29       delay(500);
30   }
31 }
```

1.4 Interfacing the PIR through Python

1.4.1 Interfacing the PIR

In this section, we discuss how to carry out the experiments of the section Sec. 1.3 using Python. We will list the same experiment. The HC-SR501 should be attached to the Arduino Uno board before doing these experiments and the Arduino Uno needs to be connected to the computer with a USB cable, as shown in Fig 1.1 .The Firmware code given in appendix should be uploaded to the Arduino board.

1. Declare the pin used for the HC-SR501 using a variable and creating a counter variable.

```
1          self.pir = 6
2          #Declaring digital 6 to connect PIR
3          pirstate = 0
```

2. The Python command used for digitalRead function is,

```
1              val = self.obj_arduino.cmd_digital_in(1, self.pir)
```

3. The output Motion detected, Motion ended and No Motion are printed based on the input received from PIR sensor and the pirstate variable. Initially

pirstate is 0, and until the input from the Sensor is 1(High), the output printed in python screen will be "No Motion". Once the Sensor detects motion(give input 1), pirstate is changed to 1 and the output "Motion detected" is printed. Then once the input becomes 0(LOW), the output printed on python screen will be "Motion Ended".

```
            if val=="1":
               pirstate=1
               print("Motion Detected!")
               sleep(0.5)
            elif val=="0"and pirstate==1:
                print("Motion Ended!")
                sleep(0.5)
            else:
                print("No Motion!")
```

1.4.2 Python Code

Python Code 1.1 Read and display the LDR values. Available at `Origin/user-code/pir/python/PIR-FOSSEE.py`.

```
import os
import sys
cwd = os.getcwd()
(setpath, Examples) = os.path.split(cwd)
sys.path.append(setpath)

from Arduino.Arduino import Arduino
from time import sleep

class PIR:
    def __init__(self,baudrate):
        self.baudrate = baudrate
        self.setup()
        self.run()
        self.exit()

    def setup(self):
        self.obj_arduino = Arduino()
        self.port = self.obj_arduino.locateport()
        self.obj_arduino.open_serial(1, self.port, self.baudrate)

    def run(self):
        self.pir = 6
        #Declaring digital 6 to connect PIR
        pirstate = 0
        for i in range(100):
            val = self.obj_arduino.cmd_digital_in(1, self.pir)
```

```
28              #function to get digital input
29              if val=="1":
30                 pirstate=1
31                 print("Motion Detected!")
32                 sleep(0.5)
33              elif val=="0"and pirstate==1:
34                  print("Motion Ended!")
35                  sleep(0.5)
36              else:
37                  print("No Motion!")
38                  sleep(0.5)
39
40
41
42      def exit(self):
43          self.obj_arduino.close_serial()
44
45  def main():
46      obj_pir = PIR(115200)
47
48  if __name__=='__main__':
49      main()
```

1.5 16x2 LCD

- **GND** should be connected to the ground of Arduino.
- **VCC** is the power supply for the LCD which we connect the 5 volts pin on the Arduino
- **Vo (LCD Contrast)** controls the contrast and brightness of the LCD. Using a simple voltage divider with a potentiometer, we can make fine adjustments to the contrast.
- **RS (Register Select)** pin lets the Arduino tell the LCD whether it is sending commands or the data. Basically, this pin is used to differentiate commands from the data. For example, when RS pin is set to LOW, then we are sending commands to the LCD (like set the cursor to a specific location, clear the display, scroll the display to the right and so on). And when RS pin is set on HIGH we are sending data/characters to the LCD.
- **R/W (Read/Write)** pin on the LCD is to control whether or not you're reading data from the LCD or writing data to the LCD. Since we're just using this LCD as an OUTPUT device, we're going to tie this pin LOW. This forces it into the WRITE mode.

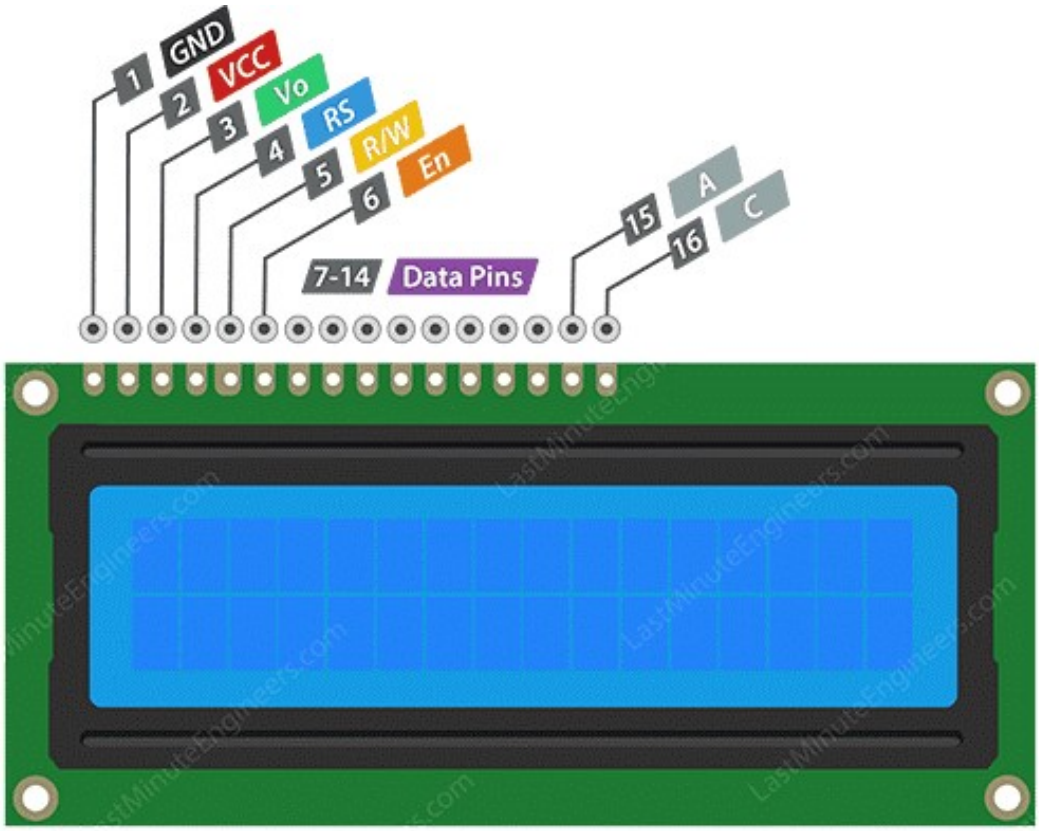


Figure 1.5: Pinout of 16x2 LCD

- **E (Enable)** pin is used to enable the display. Meaning, when this pin is set to LOW, the LCD does not care what is happening with R/W, RS, and the data bus lines; when this pin is set to HIGH, the LCD is processing the incoming data.

- **D0-D7 (Data bus)** are the pins that carries the 8 bit data we send to the display. For example, if we want to see the uppercase 'A' character on the display we will set these pins to 0100 0001(according to the ASCII table) to the LCD.

- **A-K (Anode & Cathode)** pins are used to control the backlight of the LCD.

1.6 Interfacing the PIR & LCD through the Arduino IDE

1.6.1 Interfacing the PIR & LCD

Controlling LCD is a quite complicated task. Fortunately, thanks to the LiquidCrystal library, this library simplifies the process of controlling LCD for you so we don't need to know the low-level instructions. We just need to connect Arduino to LCD and use the functions of the library.

1. Include the library:

```
#include<LiquidCrystal.h>
```

2. Creates an LCD object with parameters: (rs, enable, d4, d5, d6, d7)

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

3. Set up the LCD's number of columns and rows and clear the lcd screen.

```
  // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Clears the LCD screen
  lcd.clear();
```

4. 4. The cursor is moved to the desired position, and the output "No motion", "Motion detected!" and "Motion ended" are displayed on the LCD.

```
  lcd.setCursor(0, 0); //Settinf cursor position to first row
    first coloumn
  val = digitalRead(inputPin);  // read input value

  if (val == HIGH)  // check if the input is HIGH
  {
      lcd.print(" Motion Detected!  ");
      pirState = 1;
      delay(500);
  }
  else if(val == LOW && pirstste==1)
  {
      lcd.print(" Motion ended!  ");
      delay(500);
  }
  else if(val == LOW && pirstste==0)
  {
      lcd.print(" No Motion!  ");
      delay(500);
  }
}
```

1.6.2 Arduino Code

Arduino Code 1.2 Read and display the PIR values. Available at `Origin/user-code/pir/arduino/pir-lcd/pir-lcd.ino`.

```
#include<LiquidCrystal.h>
/* LCD Display */
// Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int inputPin = 6;                 // input pin for pir sensor
int pirState = 0;               // we start, assuming no motion detected
int val = 0;                      // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin, INPUT);
   // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Clears the LCD screen
  lcd.clear();
  Serial.begin(115200);
}

void loop(){
  lcd.setCursor(0, 0); //Settinf cursor position to first row first
    coloumn
  val = digitalRead(inputPin);  // read input value

  if (val == HIGH)  // check if the input is HIGH
  {
      lcd.print(" Motion Detected!  ");
      pirState = 1;
      delay(500);
  }
  else if(val == LOW && pirstste==1)
  {
      lcd.print(" Motion ended!  ");
      delay(500);
  }
  else if(val == LOW && pirstste==0)
  {
      lcd.print(" No Motion!  ");
      delay(500);
  }
}
```

1.7 Interfacing the PIR & LCD through Python

1.7.1 Interfacing the PIR & LCD

In this section, we discuss how to carry out the experiments of the Sec. 1.6 from Python. We will list the same experiment. The LCD should be attached to the Arduino Uno board before doing these experiments and the Arduino Uno needs to be connected to the computer with a HC-SR501 PIR Sensor USB cable, as shown in Fig 1.1 The Firmware code given in appendix should be uploaded to the Arduino board.

1. Declare the pin used for the HC-SR501 using a variable and creating a counter variable pirstate and 3 variable nomot, motdet and motend for being used in the function from Step 2.

```
1          self.pir = 6
2          pirstate = 0
3          motend = 0
4          motdet = 1
5          nomot = 2
```

2. The python-based command to print the "Motion detected!", "Motion ended!" and "No Motion!" are given below respectively.

```
1 (setpath, Examples) = os.path.split(cwd)
```

3. Set up the LCD's number of columns and rows and clear the lcd screen.

```
1                  self.obj_arduino.cmd_lcd_out(1,motdet)
```

```
1                   self.obj_arduino.cmd_lcd_out(1,motend)
```

```
1                   self.obj_arduino.cmd_lcd_out(1,nomot)
```

1.7.2 Python Code

Python Code 1.2 Read and display the LDR values. Available at `Origin/user-code/pir/python/pir-lcd.py`.

```
1 import os
2 import sys
3 cwd = os.getcwd()
4 (setpath, Examples) = os.path.split(cwd)
5 sys.path.append(setpath)
6
7 from Arduino.Arduino import Arduino
```

```
8  from time import sleep
9
10 class LCD:
11     def __init__(self,baudrate):
12         self.baudrate = baudrate
13         self.setup()
14         self.run()
15         self.exit()
16
17     def setup(self):
18         self.obj_arduino = Arduino()
19         self.port = self.obj_arduino.locateport()
20         self.obj_arduino.open_serial(1, self.port, self.baudrate)
21
22     def run(self):
23         #declaring pins
24         self.pir = 6
25         pirstate = 0
26         motend = 0
27         motdet = 1
28         nomot = 2
29         for i in range(1000):
30             val = self.obj_arduino.cmd_digital_in(1, self.pir)
31             if val=="1":
32                pirstate=1
33                self.obj_arduino.cmd_lcd_out(1,motdet)
34                #function to display "Motion detected!" on LCD
35                sleep(0.5)
36             elif val=="0" and pirstate==1:
37                 self.obj_arduino.cmd_lcd_out(1,motend)
38                 #function to display "Motion Ended!" on LCD
39                 sleep(0.5)
40             else:
41                 self.obj_arduino.cmd_lcd_out(1,nomot)
42                 #function to display "No Motion!" on LCD
43                 sleep(0.5)
44
45
46
47     def exit(self):
48         self.obj_arduino.close_serial()
49
50 def main():
51     obj_lcd = LCD(115200)
52
53 if __name__=='__main__':
54     main()
```

1.8 Setup & Output

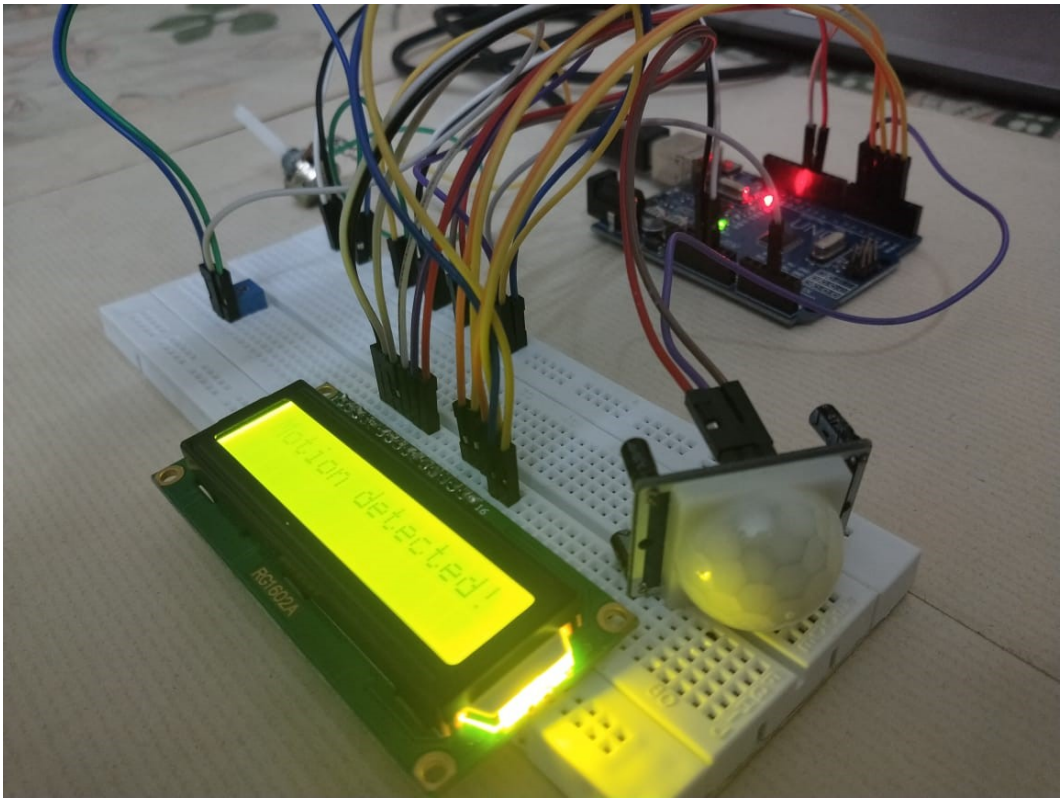


Figure 1.6: Setup 1

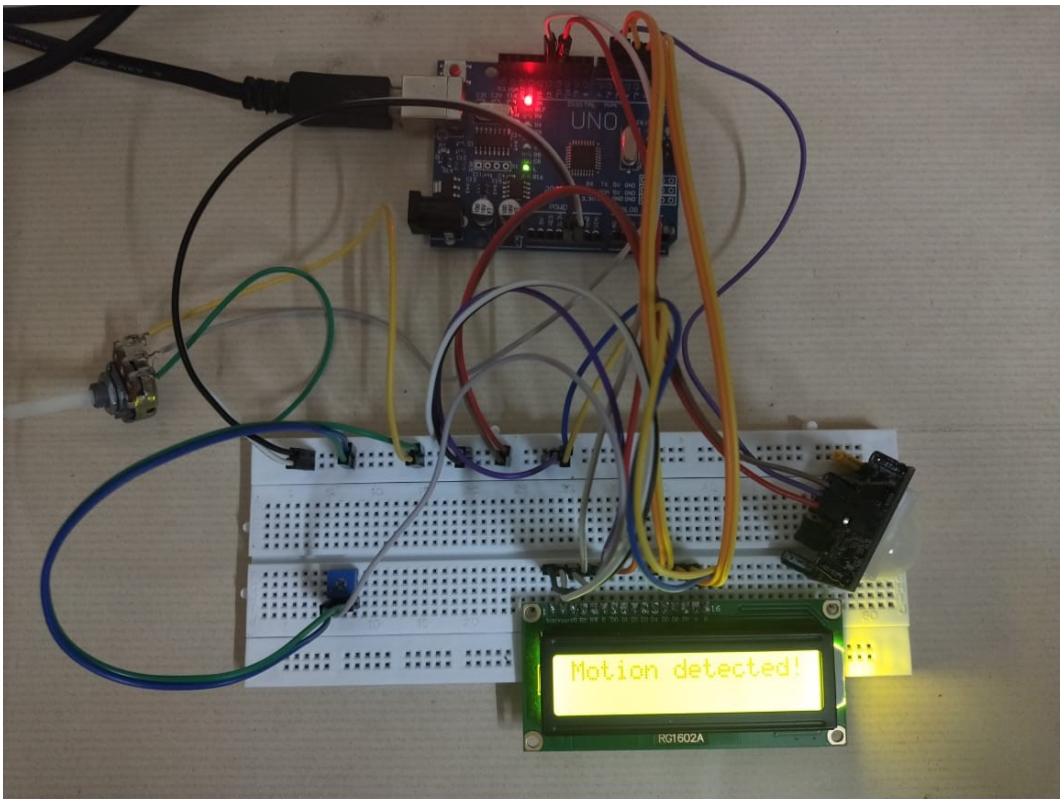


Figure 1.7: Setup 2

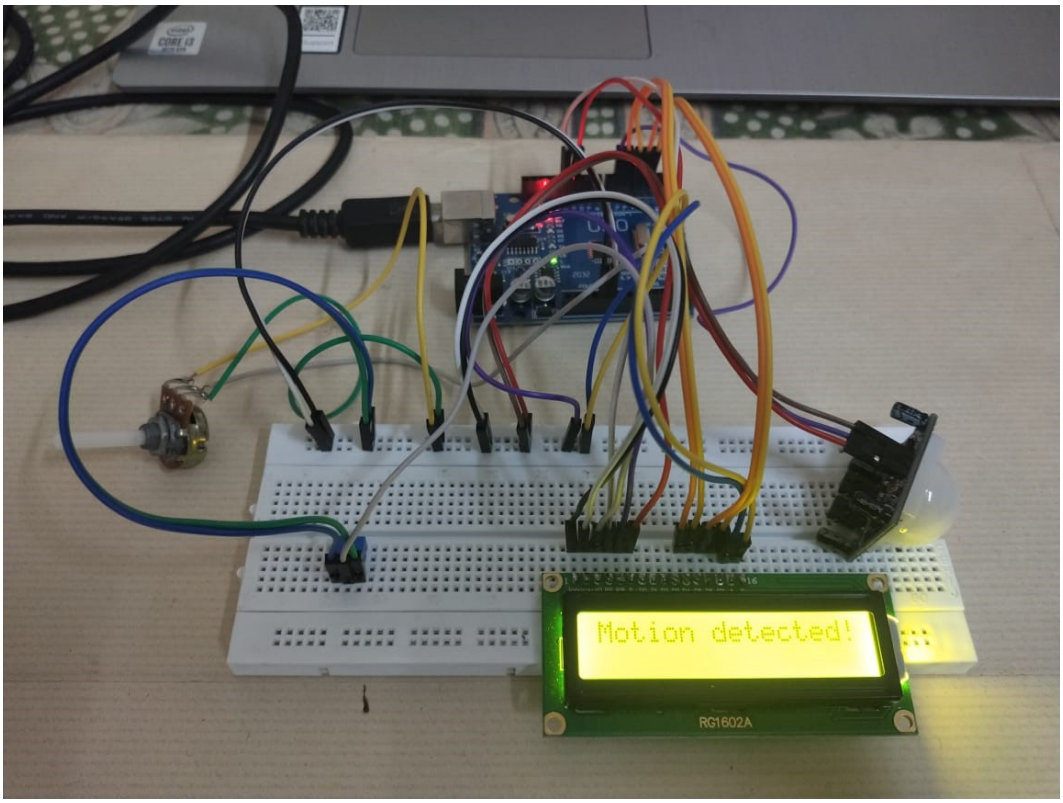


Figure 1.8: Setup 3

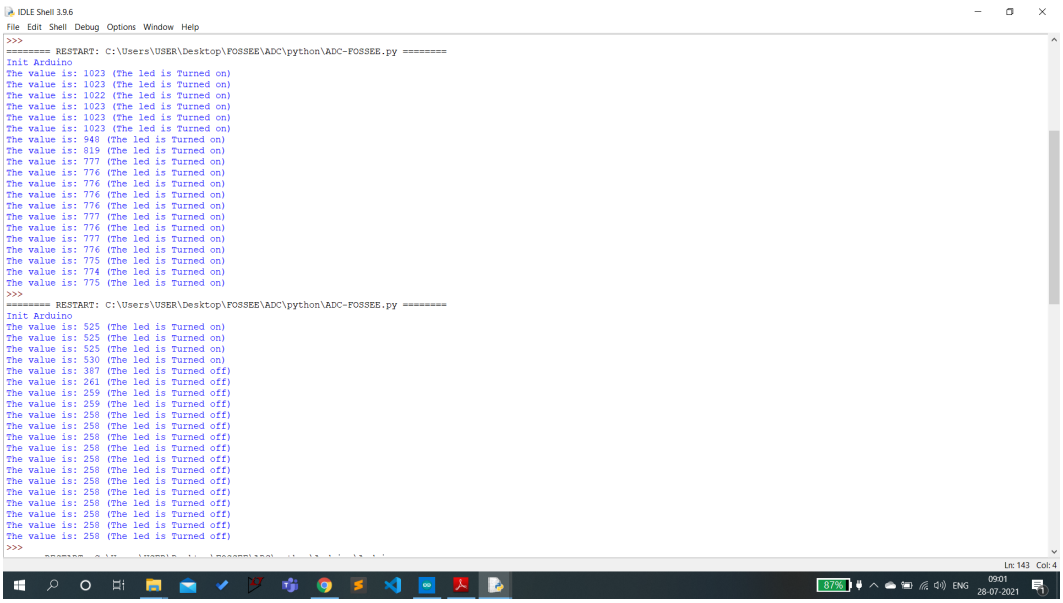


Figure 1.9: Output 1

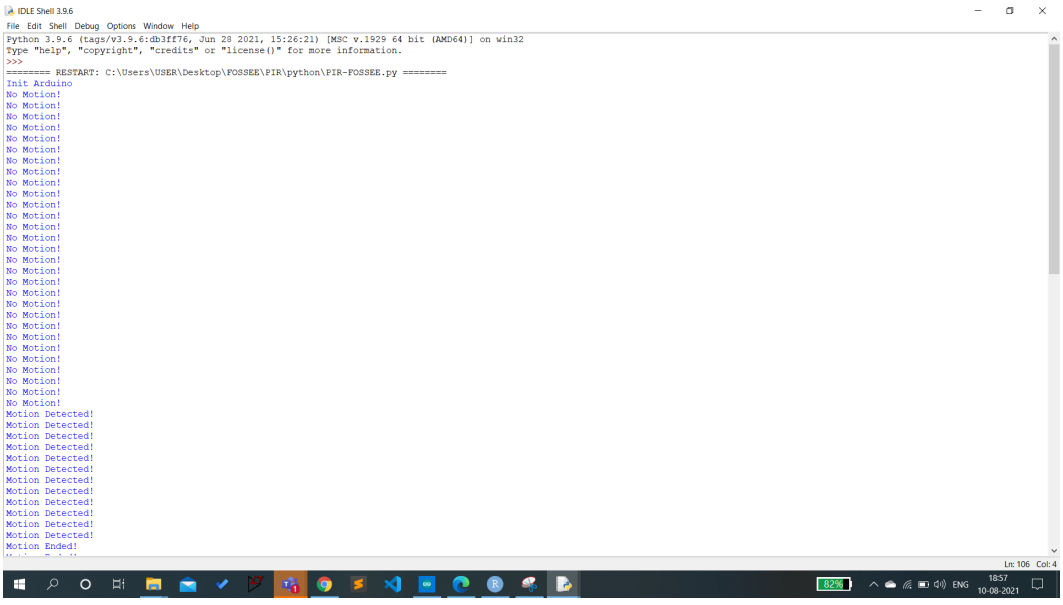


Figure 1.10: Output 2

References

[1] T. Martin. Use of scilab for space mission analysis. https://www.scilab.org/community/scilabtec/2009/Use-of-Scilab-for-space-mission-analysis. Seen on 28 June 2015.

[2] B. Jofret. Scilab arduino toolbox. http://atoms.scilab.org/. Seen on 28 June 2015.

[3] oshwa.org. http://www.oshwa.org/definition. Seen on 28 June 2015.

[4] Mateo Zlatar. Open source hardware logo. http://www.oshwa.org/open-source-hardware-logo. Seen on 28 June 2015.

[5] Arduino uno. https://www.arduino.cc/en/uploads/Main/ArduinoUnoFront240.jpg. Seen on 28 June 2015.

[6] Arduino mega. https://www.arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Fronte.jpg. Seen on 28 June 2015.

[7] Lilypod arduino. https://www.arduino.cc/en/uploads/Main/LilyPad_5.jpg. Seen on 28 June 2015.

[8] Arduino phone. http://www.instructables.com/id/ArduinoPhone/. Seen on 28 June 2015.

[9] Candy sorting machine. http://beta.ivc.no/wiki/index.php/Skittles_M%26M%27s_Sorting_Machine. Seen on 28 June 2015.

[10] 3d printer. http://www.instructables.com/id/Arduino-Controlled-CNC-3D-Printer/. Seen on 28 June 2015.

[11] Shield. http://codeshield.diyode.com/about/schematics/. Seen on 28 June 2015.

[12] scilab.org. http://www.scilab.org/scilab/about. Seen on 28 June 2015.

[13] scilab.org. http://www.scilab.org/scilab/interoperability. Seen on 28 June 2015.

[14] scilab.org. http://www.scilab.org/scilab/features/xcos. Seen on 28 June 2015.

[15] python.org. https://www.python.org/doc/essays/blurb/. Seen on 24 February 2021.

[16] pyserial - pypi. https://pypi.org/project/pyserial/. Seen on 21 April 2021.

[17] julialang.org/. https://julialang.org/. Seen on 12 April 2021.

[18] Juliaio/serialports.jl: Serialport io streams in julia backed by pyserial. https://github.com/JuliaIO/SerialPorts.jl. Seen on 15 April 2021.

[19] Openmodelica. https://www.openmodelica.org/. Seen on 2 April 2021.

[20] Thermistor - wikipedia. https://en.wikipedia.org/wiki/Thermistor. Seen on 2 May 2021.

[21] Secrets of arduino pwm. https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM. Seen on 5 May 2021.

[22] Servo. https://www.arduino.cc/reference/en/libraries/servo/. Seen on 6 May 2021.

[23] Modbus. https://modbus.org/. Seen on 6 May 2021.

[24] Paavni Shukla, Sonal Singh, Tanmayee Joshi, Sudhakar Kumar, Samrudha Kelkar, Manas R. Das, and Kannan M. Moudgalya. Design and development of a modbus automation system for industrial applications. In *2017 6th International Conference on Computer Applications In Electrical Engineering-Recent Advances (CERA)*, pages 515–520, 2017.

[25] Simply modbus. https://simplymodbus.ca/. Seen on 6 May 2021.

[26] Online crc. https://www.lammertbies.nl/comm/info/crc-calculation. Seen on 2 May 2021.

[27] Floating point converter. https://www.h-schmidt.net/FloatConverter/IEEE754.html. Seen on 6 May 2021.