# PIR Motion Sensor Based Home Security System Using Arduino

***Abstract*- The project is designed to enhance home security using a PIR (Passive Infrared) motion sensor connected to an Arduino. The system detects movement within a defined area and triggers an alarm, notifying the homeowner of potential intrusions.**

## I. Objective

A. To design a simple, low-cost home security system.
B. To simulate the system using Proteus 8 before actual hardware implementation.
C. To provide visual and audible alerts when motion is detected.

## II. Components Used

| Component | Description | Quantity |
|---|---|---|
| Arduino UNO | Microcontroller board | 1 |
| PIR Motion Sensor | Detects Motion | 1 |
| Buzzer | Produces audible alarm | 1 |
| LED | Visual alert | 1 |
| Resistors | Current limiting | As needed |
| Connecting Wires | To connect components | As needed |
| Proteus 8 | Simulation software | 1 |

## III. Working Principle

Motion Detection: The PIR sensor detects infrared radiation changes caused by moving objects (humans, animals).
Signal Processing: The sensor sends a high signal to the Arduino when motion is detected.
Alarm Activation: The Arduino triggers the buzzer and LED for visual and audible alerts.
Simulation: Proteus 8 is used to simulate the circuit, validate the connections, and test the functionality before physical implementation.

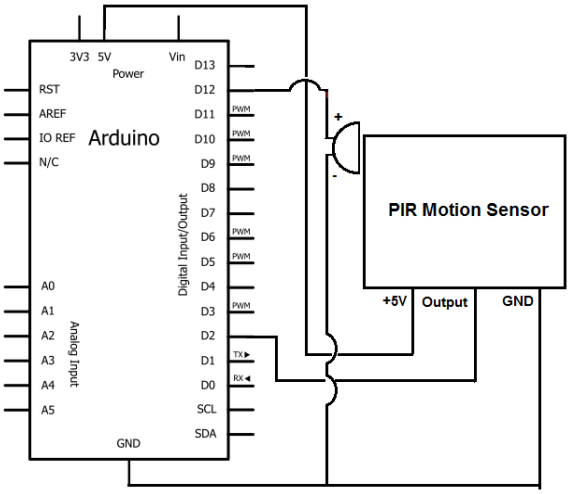
*How it works-*

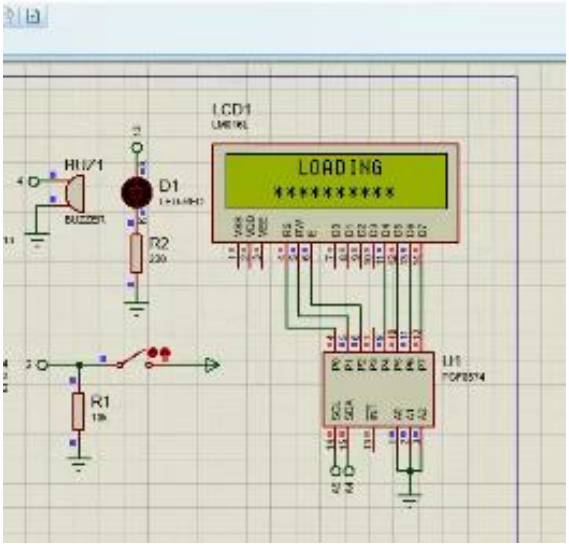
- The LCD will display "Alarm Security System" in the beginning, followed by "LOADING **********".
- After loading the screen, the LCD will display "Alarm Activated".
- If the sensor detects movement, the LED light will blink, followed by a buzzer. Then, the LCD will display "Motion Detected Alarm Triggered".
- The BUZZER and LED are still ON even though the PIR sensor does not detect any movement after the alarm is triggered.
- Press the push button and ensure the PIR sensor did not detect any movement to reset the alarm. Then, the LCD will display "Alarm Reset" followed by "Alarm Activated".

*The circuit:*

- Connect PIR motion sensor GND, OUT, and VCC pins to the GROUND, 3, and 5V pins of Arduino.
- Connect LED from pin 13 Arduino to the GROUND through 220 Ω resistor.
- Connect pushbutton from pin 2 Arduino to +5V. Then, add a 10kΩ resistor from pin 2 to the GROUND.
- Connect buzzer + pin to pin 3 Arduino and - pin to the GROUND.
- Connect LCD 1602 I2C SDA and SCL pins to Arduino UNO SDA and SCL pins. Then, connect GND and VCC pins to the ground and +5V.

## IV. Block Diagram:




*Explanation of Arduino Code:*

*A. LCD Initialization:*

```
LiquidCrystal_I2C lcd(0x20,16,2);
```

Initializes a 16×2 I2C LCD with address 0x20. The LCD is used to display system status such as loading, motion detected, alarm activated, etc.

*B. Pin Configuration:*

```
pinMode(pirPin, INPUT);
pinMode(ledPin, OUTPUT);
pinMode(buzPin, OUTPUT);
pinMode(buttonPin, INPUT);
```

Set PIR sensor as Input, LED & Buzzer as Outputs (these blink when alarm is triggered) and Pushbutton is Input (resets the alarm).

*C. PIR Sensor Calibration*

```
for(int i = 0; i < calibrationTime; i++)
{
 lcd.setCursor(i+3,1);
 lcd.print("*");
 delay(100);
}
```

The PIR sensor requires a short warm-up time. During this time, the LCD shows a loading animation.

*D. Motion Detection Logic*

```
if (digitalRead (pirPin)==HIGH)
{
  lcd.print("Motion Detected");
  lcd.print("Alarm Triggered");
```

When the PIR sensor detects motion, the system enters alarm mode.

*E. Alarm Blinking (LED and Buzzer)*

```
if (currentMillis - previousMillis >= interval)
{
  previousMillis = currentMillis;
  ledState = (ledState == LOW) ? HIGH : LOW;
  digitalWrite(ledPin, ledState);
  digitalWrite(buzPin, ledState);
}
```

Uses millis() instead of delay() which Allows non-blocking blinking of the LED and buzzer.

*F. Reset Button Debouncing*

```
if (reading != lastButtonState)
  lastDebounceTime = millis();
```

Debouncing ensures that noise from the pushbutton does not falsely register multiple presses.

*G. Alarm Reset Mechanism*

```
if (buttonState == HIGH)
{
```

```
    lockLow=false;
    digitalWrite(ledPin, LOW);
    digitalWrite(buzPin, LOW);
}
```

When the button is pressed:
Alarm stops, LED and buzzer are turned off and system waits for next motion event.

## V. Features

● Detects motion within a range of 5–7 meters (depending on PIR sensor).
● Provides audible and visual alerts.
● Can be expanded to send notifications or trigger other devices.
● Fully simulated in Proteus 8 before hardware deployment.

## VI. Challenges and Solutions

A. Simulating the PIR sensor in Proteus:
Used the PIR component library and adjusted sensitivity settings.

B. Avoiding false triggers: Added a small delay (delay(200)) in the loop to prevent rapid triggering.

C. Debouncing pushbutton: Implemented software debouncing with a 50 ms delay to avoid false resets.

## VII. Outcome / Learning

A. Learned how to interface sensors with Arduino.

B. Gained experience in simulating electronic circuits using Proteus 8.

C. Developed skills in embedded programming with Arduino IDE.

D. Understood the principles of home security systems and motion detection.

## VIII. Applications

A. Home and office security.

B. Intrusion detection in restricted areas.

C. Energy-efficient lighting systems – Automatically turn lights on/off when someone enters/leaves a room.

D. Wildlife monitoring – Detect animals in conservation or research projects.

## Conclusion

The PIR Motion Sensor Based Home Security System using Arduino successfully detects motion and triggers alerts. Simulation in Proteus 8 ensures the circuit works correctly before real-world implementation. This project is an effective, low-cost solution for basic home security.

## References


1. Arduino Official Documentation: https://www.arduino.cc/
2. https://www.circuitmagic.com/arduino/pir-motion-sensor-with-arduino/
3. https://www.theengineeringprojects.com/2016/01/pir-sensor-library-proteus.html
4. https://docs.arduino.cc/built-in-examples/digital/BlinkWithoutDelay/