Project Report on

“Smart Intrusion Detection System”

By:

Aryan Deshmukh

Page Index

Abstract **6**

1. Introduction **7**

1.1 Need for the new System
1.2 Detailed Problem Definition
1.3 Presently Available Systems
1.4 Modules on the System
1.5 Future Prospects

2. Requirement Analysis **9**

2.1 Methods used for Requirement Analysis
2.2 Data Requirement
2.3 Functional Requirement
2.4 System Specification

3. Planning And Scheduling **11**

3.1 Project Planning
3.2 Project Scheduling
3.3 Risk Assessment

4. System Modelling **13**

4.1 Software Requirements Specification
4.2 Data Flow Diagram

5. Design **14**

5.1 Use Case Specification
5.2 UML Diagram

6. Coding **17**

6.1 Hardware Specification

6.2 Additional Hardware Components Used

6.3 Platform

6.4 Programming Languages Used

6.5 Software Tools Used

6.6 Coding Style Followed

6.7 Code

7. Testing **30**

7.1 Test Plan

7.2 Runtime Snaps

Conclusion **34**

Table Index

Table No.	Table	Page No.
2.4.1	**System Specification**	**10**

Figure Index

Figure No.	Figure Name	Page No.
4.2.1	**Data Flow Diagram**	**13**
5.2.1	**Use Case Diagram**	**16**
6.1.1	**Raspberry Pi 3b**	**18**
6.1.2	**PIR Sensor Module**	**20**
6.1.3	**Pi Camera Module**	**22**
6.2.1	**Micro SD Card**	**23**
6.2.2	**Power Supply**	**23**
6.2.3	**Jumper Cables**	**24**
7.1.2	**Python Script**	**32**
7.2.2	**Email Alert Sent**	**32**
7.2.3	**Email Alert Received**	**33**

Abstract

The "Smart Intrusion Detection System" project presents a comprehensive solution for real-time security monitoring and intrusion detection. Leveraging the capabilities of the Raspberry Pi, a compact and cost-effective single-board computer, this system combines a Passive Infrared (PIR) sensor for motion detection and a Pi Camera for capturing visual evidence. The system continuously monitors the environment, and upon detecting motion, triggers the camera to capture images or record videos. Additionally, it sends notifications to the owner, enhancing security and enabling remote surveillance. This project showcases the practical application of IoT (Internet of Things) technology in enhancing home and workplace security, offering an affordable, customizable, and effective solution for intrusion detection. The report outlines the system's components, implementation, and results, underscoring its potential for broader use in various security applications.

1. INTRODUCTION

Security is a paramount concern in today's world, and as technology continues to advance, so do the tools available to enhance it. The "Intrusion Detection System using Raspberry Pi, PIR Sensor, and Pi Camera" project represents an innovative approach to security monitoring and intrusion detection. This report delves into the design, implementation, and results of a cost-effective, IoT-based solution that leverages the power of the Raspberry Pi, Passive Infrared (PIR) sensor, and Pi Camera to create a real-time, automated security system.

By combining motion detection and visual evidence capture, this system provides an effective means of safeguarding spaces and assets while offering remote surveillance capabilities. This introduction sets the stage for a detailed exploration of the project's components, methodologies, and outcomes, highlighting its relevance in the realm of modern security technology.

1.1 Need for the new system

The "Smart Intrusion Detection System" using Raspberry Pi, PIR Sensor, and Pi Camera addresses the need for affordable, customizable, and remotely accessible security solutions. It enhances security, offers cost-effective surveillance, and accommodates remote monitoring in an era of increasing security concerns and the demand for personalized security systems. Additionally, it serves as an educational platform for IoT technology and sensor integration.

1.2 Detailed Problem Definition

The problem at hand pertains to the need for an effective and accessible security solution that can detect and record unauthorized intrusions in various environments, including homes, offices, and public spaces. Traditional security systems can be expensive and lack remote monitoring capabilities.

This project aims to address these issues by designing a cost-effective, IoT-based intrusion detection system using a Raspberry Pi, PIR sensor, and Pi Camera. The system must be capable of real-time motion detection, visual evidence capture, and remote surveillance while allowing for customization and educational value. The challenge is to create a reliable, modular, and user-friendly solution that aligns with modern security needs and technological advancements.

1.3 Presently Available Systems

Existing security and intrusion detection systems vary in complexity, features, and applications. Some of the presently available systems include:

- Traditional Alarm Systems: These systems typically include door/window sensors, motion detectors, and alarm panels. They trigger audible alarms when an intrusion is detected but lack remote monitoring capabilities.
- CCTV (Closed-Circuit Television) Systems: CCTV systems use cameras to record video footage. They are widely used for surveillance but may not always include real-time intrusion detection.
- Smart Home Security Systems: Smart security systems integrate IoT technology, allowing users to monitor their premises remotely. They often include cameras, motion sensors, and mobile apps for control and notifications.
- Commercial Intrusion Detection Systems: These are advanced security solutions used in commercial and industrial settings. They may include access control, biometrics, and central monitoring stations.

1.4 Modules on the System

The "Smart Intrusion Detection System" incorporates essential modules to achieve its security objectives. The Raspberry Pi serves as the central control unit, while the PIR sensor detects motion and the Pi Camera captures visual evidence. Python programming handles data processing, GPIO pins facilitate sensor connections, and email notifications provide real-time alerts. A motion detection algorithm triggers image capture, and an image/video storage module retains evidence. Remote access and monitoring enhance the system's utility, collectively forming a comprehensive intrusion detection solution.

2. REQUIREMENT ANALYSIS

2.1 Methods Used for Requirement Analysis

Mentioned below are some of the methods that can be used for Requirement Analysis:

- Stakeholder Interviews: Gathered user input and security expectations.
- Use Case Scenarios: Explored real-life intrusion scenarios and system requirements.
- Market Research: Examined existing security systems and technology trends.
- Risk Assessment: Identified potential security risks and vulnerabilities.
- Regulatory Compliance: Ensured adherence to legal and privacy regulations.
- Functional and Non-Functional Requirements: Documented technical and operational specifications.
- Prototyping and Testing: Validated system feasibility and refined specific requirements.
- Feedback and Iteration: Incorporated user feedback to align system features with expectations.

2.2 Data Requirement

The IoT Smart Intrusion Detection System project necessitates the collection of specific data to achieve its objectives. This includes real-time data from the PIR sensor for motion detection and triggering image capture, the storage of captured images or videos from the Pi Camera, email notification details, user data for system control, system logs for activity tracking, and relevant metadata. These data components collectively enable the system to detect intrusions, capture visual evidence, and notify users, forming the core of the project's functionality and reporting capabilities.

2.3 Fractional Requirement

- The system shall be capable of detecting motion using a PIR sensor, triggering an intrusion event when motion is detected.
- The system shall capture images or video footage using the Pi Camera when an intrusion event occurs.
- The system shall send email notifications to a specified recipient with attached intruder photos upon intrusion detection.
- The system shall maintain logs of intrusion events, including timestamps and sensor triggers, for auditing and reference.

2.4 System Requirement

Sr. No.	COMPONENTS	SHORT DESCRIPTION
1	Raspberry Pi	Raspberry Pi 3B v1.2
2	PIR Sensor	Passive Infrared Sensor
3	Pi Camera	Pi Camera Module rev 1.3
4	SD Card	Micro SD Card with Raspbian OS
5	Jumper Cables	Jumper Cables for Connection
6	Power Supply	Pi Power Supply (15 Watts)

Table No. – 2.4.1

3. PLANNING AND LIMITATION

3.1 Planning

i. Objective Setting: Clearly define project goals, including motion detection, image capture, and email notifications for intrusion events.
ii. Resource Allocation: Allocate Raspberry Pi, PIR sensor, Pi Camera, and email server resources. Designate roles and responsibilities.
iii. Timeline: Create a project schedule with milestones for sensor setup, software development, testing, and report generation.
iv. Budget: Estimate costs for hardware, software, and any additional components.
v. Risk Assessment: Identify potential risks, such as sensor malfunction or email delivery issues, and plan for mitigation.
vi. Testing and Validation: Plan testing phases to ensure the system meets objectives.
vii. Documentation: Outline a reporting structure for documenting project progress and results.

3.2 Risk Assessment

Mentioned below are some of the key risk assessments to be kept in mind for the project:

- Sensor Reliability: Risk of PIR sensor malfunctions leading to false alarms or missed intrusions. Mitigation: Use high-quality sensors, conduct regular maintenance.
- Network Issues: Risk of email notifications not being sent or received due to network problems. Mitigation: Ensure stable network connections, implement email server redundancy.
- Data Security: Risk of unauthorized access to captured images or email content. Mitigation: Implement encryption and secure access controls.
- System Compatibility: Risk of hardware or software incompatibility issues. Mitigation: Thoroughly test and ensure compatibility between components.

- Power Supply Interruption: Risk of power failures leading to system downtime. Mitigation: Use backup power sources or uninterruptible power supplies (UPS).
- Environmental Factors: Risk of weather or environmental conditions affecting sensor performance. Mitigation: Employ weather-resistant enclosures and protective measures.
- User Privacy: Risk of privacy concerns regarding captured images. Mitigation: Comply with data privacy regulations and user consent.

4. SYSTEM MODELING

4.1 Software Requirement Specification

i. <u>Raspbian OS:</u>

Raspbian is the official operating system for Raspberry Pi. It's based on Debian and is optimized for the Raspberry Pi's ARM architecture. Raspbian provides essential Linux tools and a user-friendly interface, making it ideal for IoT projects. It offers easy access to system resources and hardware interfaces.

ii. <u>Python Libraries:</u>

- OpenCV (Open-Source Computer Vision Library): OpenCV is a widely used open-source computer vision and image processing library. It enables image and video analysis, making it valuable for tasks like object detection and image capture in IoT projects.
- smtplib (Simple Mail Transfer Protocol Library): The smtplib library allows Python programs to send email messages. In IoT projects, it's used for email notifications, making it a crucial tool for alerting users when specific events occur.
- RPi.GPIO (Raspberry Pi GPIO): RPi.GPIO is a Python library for Raspberry Pi's General-Purpose Input/Output (GPIO) pins. It provides easy access to GPIO pins, enabling interaction with external sensors and devices, like the PIR sensor in the intrusion detection system.

4.2 Data Flow Diagram

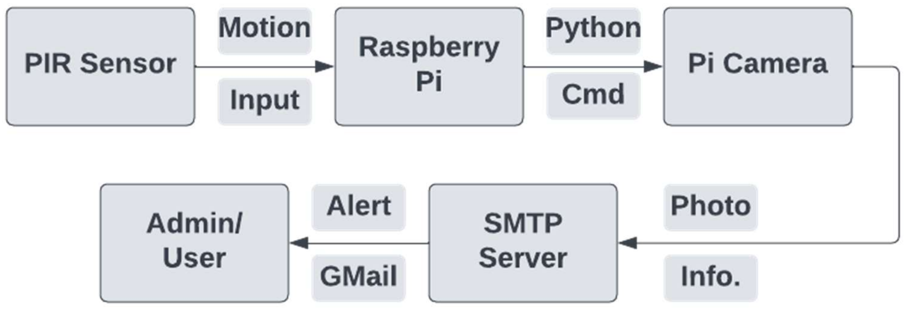


Figure 4.2.1 – Data Flow Diagram

5. DESIGN

5.1 Use Case Specification

Use Case 1: Monitor Motion Detection

Description: The system continuously monitors the PIR sensor for motion detection events.

Actors: PIR Sensor

Main Flow: The PIR sensor detects motion.

The system processes the motion event.

The system triggers the image capture process.

Use Case 2: Capture Intrusion Images

Description: The system captures images or video when an intrusion event is detected.

Actors: Pi Camera

Main Flow: An intrusion event is detected by the PIR sensor.

The system activates the Pi Camera.

The Pi Camera captures images or video.

Use Case 3: Send Email Notification

Description: The system sends email notifications upon intrusion detection.

Actors: Email Server (SMTP)

Main Flow: Intrusion is detected, and images or video are captured.

The system initiates the email notification process.

The system sends an email notification with attached images.

Use Case 4: Maintain Intrusion Logs

Description: The system maintains logs of intrusion events, timestamps, and sensor triggers.

Actors: Logging Process

Main Flow:

Intrusion events are detected.

The system logs intrusion event data, timestamps, and sensor triggers.

Use Case 5: User Interaction

Description: The user interacts with the system for configuration and access to alerts.

Actors: User

Main Flow:

The user accesses the system for monitoring and control.

The user may configure system preferences or view intrusion alerts.

5.2 UML Diagram

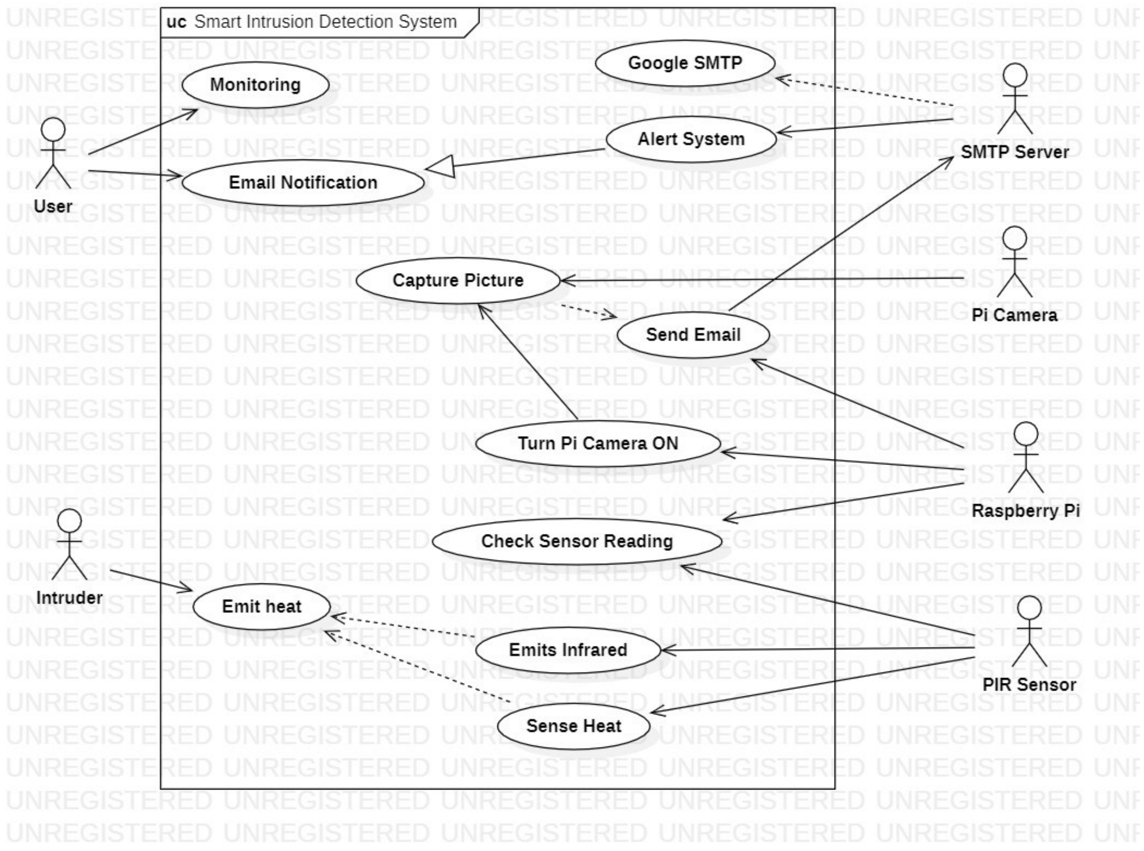


Figure 5.2.1 - Use Case Diagram

6. CODING AND SPECIFICATIONS

We have used various hardware for the implementation for this project.

The main components used for this project are Raspberry Pi 3b, PIR Sensor and Pi Camera.

Some of the other components used in this project are Micro SD Card, Jumper Cables and Power Supply.

6.1 Hardware Specification

Raspberry Pi 3b v1.2:

The Raspberry Pi 3 Model B (v1.2) is a highly versatile single-board computer, known for its compact size and robust hardware capabilities. Featuring a quad-core ARM Cortex-A53 processor running at 1.2 GHz and 1 GB of RAM, it delivers ample processing power to cater to a variety of applications. With built-in Wi-Fi and Bluetooth, it seamlessly connects to networks and peripheral devices, making it suitable for IoT and automation projects, educational initiatives, multimedia centres, retro gaming setups, and much more. Its GPIO pins enable interfacing with external sensors and components, adding to its appeal for developers and tinkerers. This cost-effective and developer-friendly platform enjoys extensive community support, making it a popular choice for diverse projects and applications.

The Raspberry Pi 3B v1.2 is the go-to platform for educational institutions, hobbyists, and professionals seeking a compact yet capable computer for a wide range of uses. It’s hardware, including GPIO pins and wireless connectivity, extends its utility to countless applications, from building smart home systems and educational tools to hosting web servers and creating multimedia centres. Its low cost, extensive community resources, and support for various operating systems make it an ideal choice for those looking to experiment, innovate, and create within the realms of computing, electronics, and IoT.

Detailed Hardware Specifications are:

- Broadcom BCM2837 64-bit Quad-Core ARM Cortex-A53 CPU - 1.2 GHz clock speed.
- 1 GB LPDDR2 SDRAM
- Built-in Wi-Fi 802.11n
- Built-in Bluetooth 4.2/BLE (Bluetooth Low Energy)
- 4 x USB 2.0 ports for connecting peripherals like keyboards, mice, and USB drives.
- 1 x HDMI port for connecting to displays.
- 1 x 10/100 Ethernet port for wired network connectivity.
- 1 x 3.5mm audio/composite video jack for audio and video output.
- 1 x CSI (Camera Serial Interface) port for connecting the Raspberry Pi Camera Module.
- 1 x DSI (Display Serial Interface) port for connecting a touchscreen display.
- 40 GPIO (General Purpose Input/Output) pins for connecting to external devices and sensors.
- MicroSD card slot for the operating system and data storage.
- Powered via a micro-USB port. Recommended power supply voltage: 5V, 2.5A.
- MicroSD card slot for the operating system and data storage.
- 4-pole stereo audio and composite video output.
- Dimensions: 85.6mm x 56.5mm
- Compatible with various operating systems, including Raspbian, Raspberry Pi OS, and many Linux distributions.
- Compatible with a wide range of official Raspberry Pi accessories, including cases, power supplies, and HATs (Hardware Attached on Top).

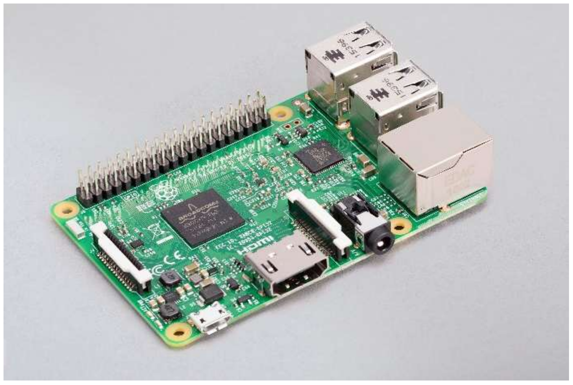


Figure 6.1.1 – Raspberry Pi

PIR Sensor:

Passive Infrared (PIR) sensors are essential components for motion detection in various applications. These sensors work by detecting changes in the infrared radiation emitted by objects in their field of view. PIR sensors are commonly used in security systems, lighting control, and IoT devices. They are designed to detect motion, making them valuable in intruder detection systems. When an object moves within the sensor's range, it triggers an output signal. In the context of an IoT Smart Intrusion Detection System, PIR sensors serve as the primary means of detecting unauthorized motion within a monitored area. Upon detecting motion, the PIR sensor sends a signal to the system, initiating further actions such as capturing images, sending alerts, and activating security measures.

PIR sensors consist of a few key components. Inside the sensor, you'll find a pyroelectric sensor that can detect changes in infrared radiation. The sensor's housing typically includes a protective lens to focus its field of view. PIR sensors are equipped with two slots or windows, allowing them to detect motion in two adjacent areas. When an object moves within the sensor's field, the temperature difference between the slots changes, and the sensor triggers a voltage change, generating an output signal. These sensors are compact, low-cost, and consume minimal power, making them suitable for battery-powered and remote applications. They are often integrated with other components like resistors and capacitors to control sensitivity and response time. PIR sensors have become invaluable in modern security systems, home automation, and energy-saving applications due to their efficiency in detecting motion and presence.

Technical Specifications of the Sensor:

- Operating Principle: Passive Infrared (PIR) sensors detect changes in infrared radiation emitted by objects in their field of view, based on the temperature difference between objects and their surroundings.
- Sensitivity: PIR sensors are designed to detect motion within a specific range. The sensitivity of the sensor can often be adjusted to fine-tune its response.
- Field of View (FOV): PIR sensors have a defined FOV, typically around 110 to 180 degrees, depending on the model. This angle determines the sensor's coverage area.

- Range: The effective range of a PIR sensor varies, but it's commonly around 20 to 30 feet (6 to 9 meters). This range depends on factors like sensor quality and the size of the detected object.
- Output Signal: PIR sensors generate an output signal when motion is detected. This signal is typically in the form of a digital pulse.
- Delay/Timeout: PIR sensors often include a configurable delay or timeout period. This setting determines how long the sensor remains active after detecting motion, preventing false triggers.
- Operating Voltage: PIR sensors are usually designed to operate at low voltage, typically around 3-5 volts, making them suitable for battery-powered devices.
- Components: Inside a PIR sensor, you'll find a pyroelectric sensor, a protective lens, and other components for signal processing and control.
- Response Time: PIR sensors have a specific response time, which is the time it takes for the sensor to detect motion and generate an output signal. This time is typically very short.
- Power Consumption: PIR sensors are known for their low power consumption, making them suitable for battery-operated and energy-efficient applications.
- Dimensions: PIR sensors come in various sizes and form factors, allowing for flexibility in design and integration into different devices and systems.
- Environmental Considerations: PIR sensors are sensitive to temperature changes. Extreme temperature variations can affect their performance.

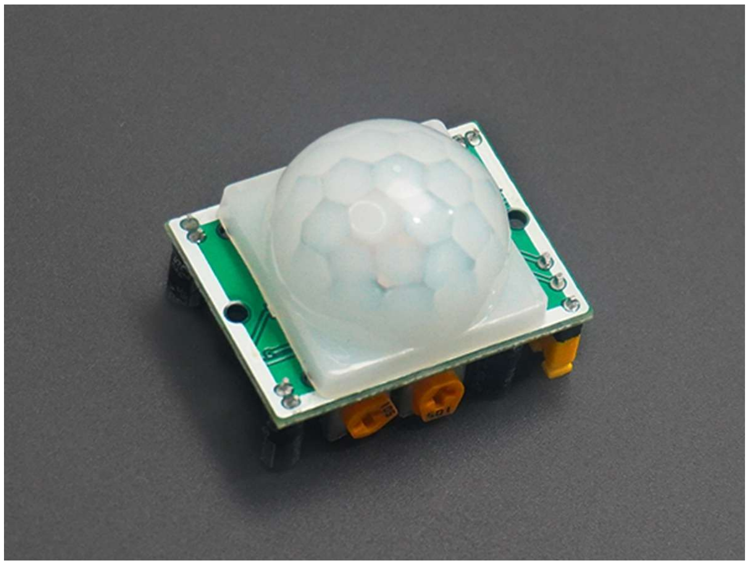


Figure 6.1.2 – PIR Sensor

Pi Camera Module:

The Raspberry Pi Camera Module is a versatile and compact camera designed specifically for Raspberry Pi single-board computers. It features high-quality imaging capabilities with fixed-focus lenses and various configurations. This camera module is commonly used in applications such as home security systems, wildlife monitoring, photography, video streaming, and robotics. It is ideal for capturing high-resolution images and videos and can be controlled programmatically, making it an essential tool for projects requiring visual data.

The Raspberry Pi Camera Module boasts a compact form factor with a high-quality image sensor, typically offering resolutions from 5 to 12 megapixels. Its fixed-focus lens ensures sharp image and video capture without manual adjustments. This module connects securely to the Raspberry Pi via a dedicated interface connector. It may come with mounting holes for additional accessories and allows for programmable control, enabling automation and customized functionality through languages like Python.

Technical Specifications of the Module:

- Resolution: The camera module is available in different resolutions, with common options being 5 megapixels (2592 x 1944) and 8 megapixels (3280 x 2464).
- Lens: The module features a fixed-focus lens, which means it doesn't require manual adjustments for focusing.
- Sensor: It uses an OmniVision OV5647 image sensor.
- Image Sensor Type: CMOS (Complementary Metal-Oxide-Semiconductor) sensor technology.
- Frame Rate: The camera can capture video at various frame rates, such as 30 frames per second (fps) for 1080p video.
- Video Formats: It supports H.264 video compression, allowing efficient storage and streaming of video data.
- Interface: The module connects to the Raspberry Pi via a dedicated ribbon cable and the CSI (Camera Serial Interface) connector.
- Dimensions: The camera module is compact, measuring approximately 25mm x 20mm x 9mm.
- Operating Voltage: It operates on the 3.3V power supply from the Raspberry Pi.

- Mounting: The module often includes mounting holes for attaching it to various structures or enclosures.
- Software Support: It is compatible with Raspberry Pi OS and can be controlled programmatically using Python, enabling users to capture images and record video through code.
- Weight: The camera module is lightweight, making it suitable for portable and embedded applications.

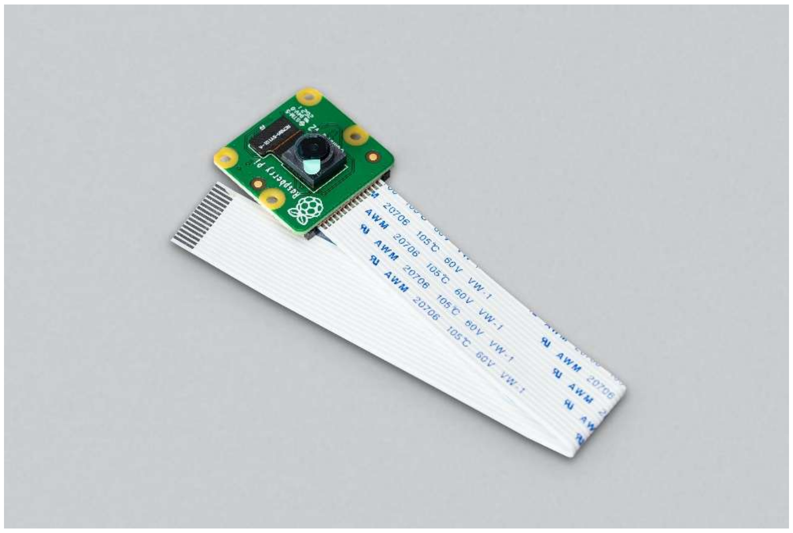


Figure 6.1.3 – Pi Camera Module

6.2 Additional Hardware Components Used

Micro SD Card:

The Micro SD (Secure Digital) card employed in this project serves as the primary storage medium for the Raspberry Pi. These cards are compact, high-capacity, and feature fast data transfer rates. The Micro SD card holds the operating system, application software, and project data. Its portability, storage capacity, and compatibility make it an essential component for the Raspberry Pi, ensuring the seamless execution of IoT projects.

Figure 6.2.1 – Micro SD Card

Power Supply:

The power supply for this project is a crucial element to ensure the Raspberry Pi and its associated components receive stable and reliable electrical power. It typically provides 5 volts and ample current to support the Raspberry Pi's operation. A stable power supply is essential to prevent system crashes and data corruption. The power source can vary from USB chargers to dedicated power adapters, depending on project requirements. It's a critical consideration for the seamless and uninterrupted functioning of the IoT-based Intrusion Detection System.

Figure 6.2.2 – Power Supply

Jumper Cables:

Jumper cables are essential for establishing electrical connections within the project. They are flexible, insulated wires with connectors at each end that facilitate easy and temporary connections between components. In this project, jumper cables are used to link the Raspberry Pi with various sensors, the camera module, and other peripherals, enabling data exchange and control. Their versatility and ease of use make them invaluable in prototyping and customizing IoT systems like the Smart Intrusion Detection System, simplifying component integration and experimentation.

Figure 6.2.3 – Jumper Cables

6.3 Platform Used

The project leverages the Raspberry Pi 3B v1.2 single-board computer as its primary platform. The Raspberry Pi is a highly versatile and compact computing device renowned for its robust processing capabilities and GPIO (General-Purpose Input/Output) pins, making it an excellent choice for Internet of Things (IoT) applications.

As the project's core control unit, the Raspberry Pi plays a pivotal role in hosting and executing the system's software. It manages the integration of multiple components, including the Pi Camera Module and PIR (Passive Infrared) sensor, ensuring seamless communication and coordination between these devices. With its ability to run a variety of programming languages, such as Python, the Raspberry Pi enables flexible and programmable control, allowing for custom automation and real-time decision-making.

In summary, the Raspberry Pi serves as the backbone of the project, providing the computational power and connectivity required to create a Smart Intrusion Detection System, enhancing security and enabling intelligent response to potential intruders.

6.4 Programming Languages Used

This project primarily utilizes Python as the main programming language for developing the software and controlling the various components, including the

Raspberry Pi, Pi Camera Module, and PIR sensor. Python is known for its ease of use, extensive libraries, and compatibility with the Raspberry Pi's GPIO pins, making it an ideal choice for IoT applications. Additionally, Python's versatility enables the implementation of email notifications and the handling of sensor data seamlessly.

6.5 Software Tools Used

Raspberry Pi OS (formerly Raspbian): The operating system installed on the Raspberry Pi, providing the foundational software environment for running applications and controlling hardware components.

Python: The primary programming language for developing the project's software. Python is used for sensor control, data processing, email notifications, and system automation.

OpenCV: An open-source computer vision library that may be utilized for image and video processing tasks, enhancing the capabilities of the Pi Camera Module.

smtplib: A Python library for sending email notifications, essential for alerting the system owner in case of an intrusion event.

GPIO (General-Purpose Input/Output) Libraries: Python libraries such as RPi.GPIO for controlling and interacting with the GPIO pins on the Raspberry Pi, facilitating sensor integration and control.

Text Editor or IDE: Software tools like Thonny, Visual Studio Code, or Nano may be used for writing and editing Python code on the Raspberry Pi.

Terminal or SSH (Secure Shell): For remote access and command-line control of the Raspberry Pi.

6.6 Coding Style Followed

The coding style followed in this project adheres to the Python community's widely accepted style guide known as PEP 8 (Python Enhancement Proposal 8). PEP 8 provides a set of conventions for writing clean, readable, and consistent Python code. Some key aspects of PEP 8 coding style include:

- Indentation: Using 4 spaces for indentation (not tabs) to ensure code readability.

- Line Length: Limiting lines to a maximum of 79 characters for code and 72 characters for docstrings and comments.
- Whitespace: Using whitespace around operators and after commas to enhance code clarity.
- Imports: Import statements are organized and grouped at the top of the script. Standard library imports come first, followed by third-party library imports, and then local imports.
- Function and Variable Names: Employing descriptive and lowercase_with_underscores names for functions, variables, and modules to convey their purpose.
- Comments: Including clear and concise comments to explain complex or critical parts of the code.
- Consistency: Maintaining a consistent coding style throughout the project to enhance code readability and maintainability.

6.7 Code

```
import RPi.GPIO as GPIO

import time

import cv2

import smtplib

from email.mime.image import MIMEImage

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText


# GPIO pin connected to the PIR sensor's OUT pin

PIR_PIN = 17


# Email settings

SMTP_SERVER = 'smtp.gmail.com'  # Gmail's SMTP server

SMTP_PORT = 587  # Port for TLS

SMTP_USERNAME = 'colonial.raspi@gmail.com'

SMTP_APP_PASSWORD = 'ycadmehsmkjsyydq'   # Generate an App Password from your
Google account

RECIPIENT_EMAIL = '202000584@vupune.ac.in'


def send_email(image_path):

    msg = MIMEMultipart()

    msg['From'] = SMTP_USERNAME

    msg['To'] = RECIPIENT_EMAIL

    msg['Subject'] = 'Intrusion Detected'


    text = MIMEText('Motion detected. Images attached.')

    msg.attach(text)
    with open(image_path, 'rb') as fp:
        img = MIMEImage(fp.read())
        msg.attach(img)

    server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    server.starttls()
    server.login(SMTP_USERNAME, SMTP_APP_PASSWORD)
    server.sendmail(SMTP_USERNAME, RECIPIENT_EMAIL, msg.as_string())
    server.quit()

def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(PIR_PIN, GPIO.IN)

    try:
        print("Intrusion Detection System Running...")
        while True:
            if GPIO.input(PIR_PIN):
                print("Motion detected!")
                timestamp = time.strftime("%Y%m%d%H%M%S")
                image_path = f"/home/pi_{timestamp}.jpg"

                # Initialize the camera
                camera = cv2.VideoCapture(0)
                time.sleep(2)  # Warm-up time

                # Capture an image
                ret, frame = camera.read()
                if ret:
```

```
                cv2.imwrite(image_path, frame)
                camera.release()
                send_email(image_path)
                time.sleep(10)  # Wait 10 seconds before rearming the sensor
            else:
                print("Failed to capture image.")

            time.sleep(1)
    except KeyboardInterrupt:
        print("Exiting...")
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

7. TESTING

7.1 Testing Plan

The following points are kept in mind for the testing phase of the project:

Unit Testing:

- Raspberry Pi and GPIO Testing: Verify that the Raspberry Pi and GPIO pins are functioning correctly.
- Camera Module Testing: Ensure the Pi Camera Module captures images and videos as expected.
- PIR Sensor Testing: Validate the PIR sensor's ability to detect motion accurately.

Integration Testing:

- Software and Hardware Integration: Test the integration of software components (Python code, OpenCV, smtplib) with hardware components (Raspberry Pi, camera, PIR sensor).
- Data Flow: Verify that data flows smoothly between sensors, the Raspberry Pi, and the camera module.

Functional Testing:

- Motion Detection: Confirm that the PIR sensor accurately detects motion and triggers actions.
- Camera Operation: Ensure the camera captures images and videos in response to motion detection.
- Email Notifications: Verify that the system successfully sends email notifications with intruder images.

User Interface Testing:

- If a web-based interface is implemented, test its functionality and user-friendliness for remote system monitoring.

Performance Testing:

- Response Time: Measure the time between motion detection and email notifications.
- Image/Video Quality: Assess the quality of captured images and videos.

Security Testing:

- Evaluate the system's resistance to unauthorized access or tampering.
- Ensure that email notifications are secure and reliable.

Usability Testing:

- If a user interface is created, assess its ease of use and effectiveness in managing the system.

Compatibility Testing:

- Device Compatibility: Check compatibility with different devices (e.g., smartphones, tablets) used for monitoring.
- Browser Compatibility: Verify compatibility with various web browsers if a web interface is included.

Stress Testing:

- Subject the system to stress conditions to identify any potential failures or weaknesses.

Documentation Review:

- Verify that all documentation, including user manuals and system guides, is accurate and complete.

User Acceptance Testing (UAT):

- Involve end-users or stakeholders to perform final testing to ensure that the system meets their requirements and expectations.

Regression Testing:

- After making any changes or updates, retest the system to ensure that existing functionalities are not affected.

Performance Optimization:

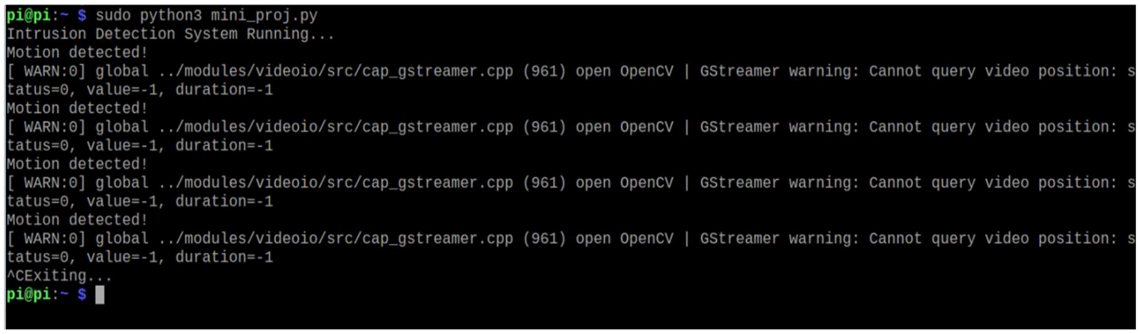
- Evaluate system performance and optimize code or configurations as needed.

Long-Term Testing:

- Leave the system operational for an extended period to monitor its reliability and stability.
- Compliance Testing: If required, ensure that the system complies with relevant industry standards or regulations.

7.2 Testing Snapshots

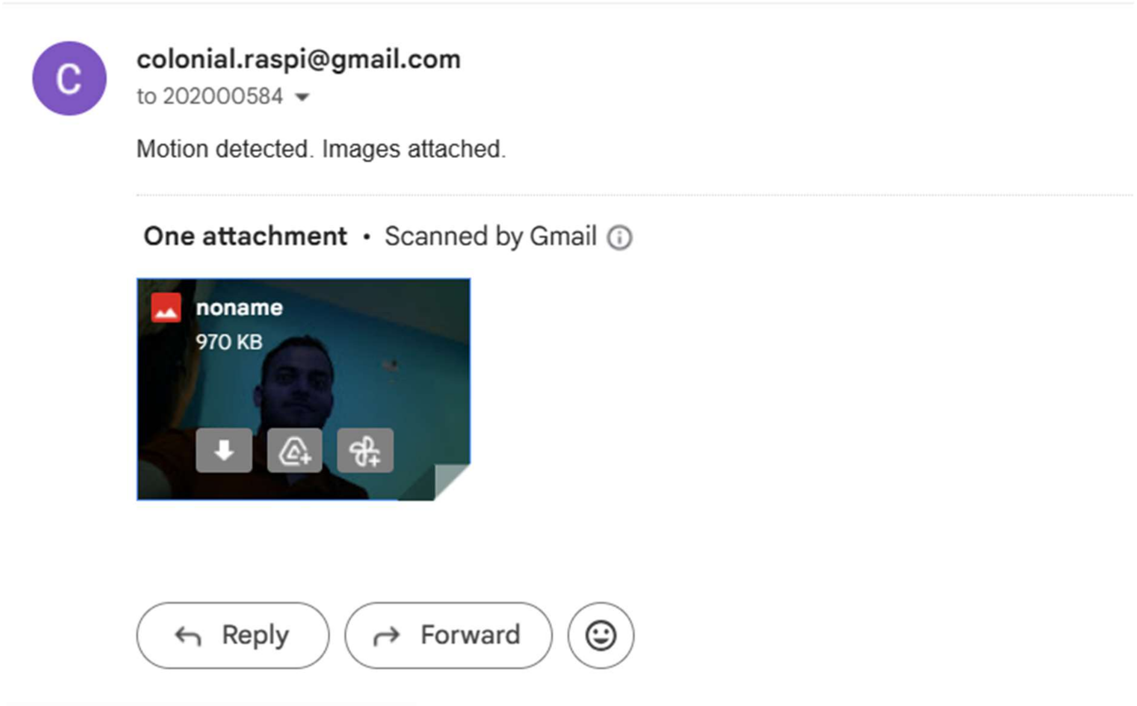
Python Script Running on Raspberry Pi.



```
pi@pi:~ $ sudo python3 mini_proj.py
Intrusion Detection System Running...
Motion detected!
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (961) open OpenCV | GStreamer warning: Cannot query video position: s
tatus=0, value=-1, duration=-1
Motion detected!
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (961) open OpenCV | GStreamer warning: Cannot query video position: s
tatus=0, value=-1, duration=-1
Motion detected!
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (961) open OpenCV | GStreamer warning: Cannot query video position: s
tatus=0, value=-1, duration=-1
Motion detected!
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (961) open OpenCV | GStreamer warning: Cannot query video position: s
tatus=0, value=-1, duration=-1
^CExiting...
pi@pi:~ $
```

Figure 7.2.1 – Python Script

Email sent to Admin from Raspberry's Gamil with the images attached.



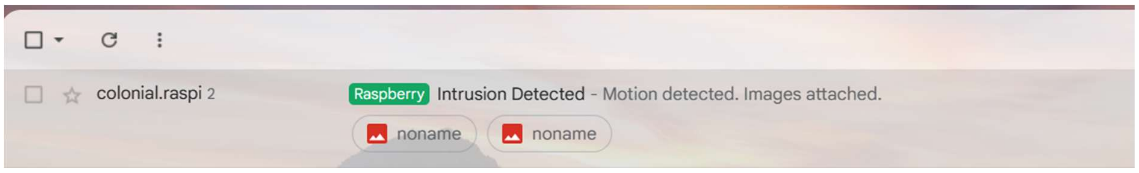
colonial.raspi@gmail.com
to 202000584

Motion detected. Images attached.

One attachment • Scanned by Gmail

Figure 7.2.2 – Email Alert Send

Motion Detection alert received by the admin.


colonial.raspi 2 Raspberry Intrusion Detected - Motion detected. Images attached.

noname noname

Figure 7.2.3 – Email Alert Recieved

Conclusion

In conclusion, the Smart Intrusion Detection System powered by the Raspberry Pi, PIR sensor, and Pi Camera Module represents a significant step forward in the realm of home security and Internet of Things (IoT) applications. This project successfully combines hardware and software elements to create an intelligent system that can detect unauthorized intrusions and respond promptly.

The project's primary objective was to design a robust security solution that incorporates motion sensing and visual capture, ultimately alerting the system owner through email notifications. By leveraging the Raspberry Pi's computational capabilities and the PIR sensor's ability to detect motion, the system offers real-time monitoring and enhances the security of a given environment.

Throughout this project, we utilized a Python-based codebase and adhered to the PEP 8 coding style to ensure a well-structured and maintainable software component. The project demonstrates the integration of multiple technologies, including image processing with OpenCV and email notifications through smtplib, to create a comprehensive and intelligent system.

This Smart Intrusion Detection System has far-reaching potential for applications in residential and small-scale commercial security, ensuring peace of mind through automated surveillance and timely alerts. By addressing the need for enhanced security in a connected world, this project showcases the practicality and effectiveness of IoT solutions in safeguarding our surroundings.