# IoT Theft Detection System – Raspberry Pi Implementation Plan

## 1. Overview

The IoT Theft Detection System leverages Raspberry Pi for real-time monitoring, face recognition, motion detection, and sensor integration (PIR, door sensor). The system alerts users through Email, MQTT, and Telegram while storing captured intrusion evidence locally.

## 2. Hardware Requirements

• Raspberry Pi 4 Model B (2GB/4GB/8GB RAM depending on performance needs)

• Raspberry Pi Camera Module v2 or USB Webcam

• PIR Motion Sensor (HC-SR501)

• Magnetic Door Sensor

• MicroSD card (32GB or higher) with Raspberry Pi OS installed

• Power supply (5V, 3A recommended)

• Internet connection (Ethernet or Wi-Fi)

## 3. Software Requirements

• Raspberry Pi OS (latest stable release)

• Python 3.10 or later

• Virtual Environment (venv)

• Required Python libraries (from requirements.txt)

• OpenCV, dlib/face\_recognition, imutils, numpy, paho-mqtt, requests, smtplib

## 4. Project Folder Structure

alerts/ – Alert modules (email, MQTT, Telegram)

camera/ – Camera interface modules

captures/ – Storage for captured images/videos

config/ – Configuration files

detection/ – Face recognition, motion detection, person detection scripts

known\_faces/ – Store known faces for recognition

models/ – Pre-trained models (MobileNetSSD)

sensors/ – Sensor scripts (PIR, door sensor)

utils/ – Helper functions

venv310/ – Virtual environment

main.py – Main entry point

requirements.txt – Dependencies list

## 5. Raspberry Pi Setup

1. Flash Raspberry Pi OS onto microSD card and boot Pi.

2. Enable Camera and I2C using raspi-config.

3. Update system: sudo apt update && sudo apt upgrade -y.

4. Install Python 3.10 and venv if not available.

5. Create virtual environment: python3 -m venv venv310.

6. Activate environment: source venv310/bin/activate.

7. Install dependencies: pip install -r requirements.txt.

8. Connect PIR and door sensors to GPIO pins as per wiring configuration.

9. Attach camera module or USB webcam.

## 6. Execution Flow

• Run main.py to initialize the system.

• Sensors monitor for intrusion (motion/door).

• Detection modules recognize faces and detect people.

• If intrusion is detected, image is stored in captures/ folder.

• Alerts are triggered through Email, MQTT, and Telegram.

• Logs are generated for system monitoring.

## 7. Alerts & Notifications

• Email: email\_alert.py sends alerts via SMTP.

• MQTT: mqtt\_publish.py sends intrusion messages to IoT dashboards.

• Telegram: telegram.py sends notifications to user’s Telegram bot.

## 8. Testing & Validation

• Test PIR sensor detection and ensure GPIO inputs are read correctly.

• Validate door sensor trigger events.

• Test face recognition with known\_faces dataset.

• Simulate unauthorized intrusion and check alert system.

• Verify captured images are stored in captures/ folder.

## 9. Future Enhancements

• Add cloud storage integration (Google Drive/Dropbox).

• Implement live video streaming using Flask/WebRTC.

• Improve detection accuracy using YOLOv8 models.

• Add two-way audio communication with intruder.

• Integration with smart home systems (Home Assistant, Node-RED).  
  
  
  
  
**Arduno integration:  
  
1️⃣ Hardware Setup**

* Arduino board (Uno/Nano/MEGA)
* Sound sensor or any other sensor you want
* Connect **sensor output** → Arduino **digital or analog pin**
* Connect **Arduino to PC** via USB

For example:

* Sound sensor **digital output** → Arduino **D2 pin**
* VCC → 5V, GND → GND

**2️⃣ Arduino Code**

Upload a simple sketch to Arduino that **sends data over serial** when the sensor is triggered:

// Arduino Sound Sensor

const int soundPin = 2; // Digital pin for sound sensor

void setup() {

Serial.begin(9600);

pinMode(soundPin, INPUT);

}

void loop() {

int val = digitalRead(soundPin);

if (val == HIGH) {

Serial.println("SOUND");

}

delay(50); // small delay to avoid flooding

}

* Every time the sound sensor detects a loud noise, Arduino sends "SOUND" over serial.

**3️⃣ Python Integration**

You can read Arduino serial data in your existing Python code using **pyserial**:

**Install pyserial:**

pip install pyserial

**Python Example (integrate into your main loop):**

import serial

import time

# Connect to Arduino (replace COM3 with your Arduino port)

arduino = serial.Serial('COM3', 9600, timeout=1)

time.sleep(2) # wait for Arduino to initialize

def detect\_sound\_from\_arduino():

if arduino.in\_waiting > 0:

line = arduino.readline().decode('utf-8').strip()

if line == "SOUND":

return True

return False

**4️⃣ Replace Microphone Detection**

In your current code, replace:

sound\_detected = detect\_sound()

with:

sound\_detected = detect\_sound\_from\_arduino()

* Now, your Python program **triggers alerts** based on the Arduino sensor.
* No microphone or sounddevice needed anymore if Arduino is used.

**5️⃣ Tips**

1. Make sure the **COM port** matches your Arduino (Device Manager → Ports COMx).
2. timeout=1 ensures Python doesn’t block waiting for serial data.
3. You can later add **multiple sensors** (light, PIR, vibration) by sending different messages from Arduino:

Serial.println("SOUND");

Serial.println("PIR");

* Then Python can act accordingly.

✅ **Summary Flow with Arduino:**

1. Arduino monitors the sound sensor.
2. When a loud sound is detected, Arduino sends a serial message.
3. Python reads the serial port every frame.
4. If sound detected + hand in restricted area → alert + log + snapshot.