Titlu proiect:

Dispozitiv de Supraveghere în Timp Real cu Detectare de Mișcare și Localizare

Motivația alegerii temei:

Proiectul propus urmărește crearea unui dispozitiv autonom de supraveghere, capabil să detecteze mișcare în timp real, să aprindă un LED, să obțină coordonatele GPS și să trimită un email de alertă către utilizator. Alegerea acestei teme a fost motivată de necesitatea unor soluții simple și eficiente pentru monitorizarea spațiilor fără supraveghere permanentă: locuințe, rulote, ateliere sau depozite. Am dorit să implementăm un sistem complet integrat care să funcționeze local, dar să notifice la distanță prin email.

Rezumat:

Proiectul propune realizarea unui dispozitiv embedded autonom care detectează mișcarea și trimite automat o alertă prin email ce conține locația GPS estimată. La baza funcționării sale stau un senzor de mișcare PIR conectat la un Raspberry Pi, un modul GPS pentru localizare și un sistem de notificare prin rețea, toate integrate printr-un script Python.

Dispozitivul monitorizează continuu mișcarea folosind senzorul PIR. Atunci când este detectată activitate, LED-ul conectat se aprinde pentru a semnaliza vizual evenimentul. În paralel, sunt citite coordonatele GPS prin comunicație serială, iar locația este transformată într-un link Google Maps. Acest link este apoi transmis automat către o adresă de email predefinită, folosind un server SMTP securizat.

Importanța în domeniul Embedded System-SI:

Proiectul evidențiază o aplicație practică a sistemelor embedded, demonstrând integrarea între senzori, comunicație serială, control prin GPIO și trimiterea de alerte prin rețea. Utilitatea sa este evidentă în securizarea spațiilor, monitorizarea în teren, aplicații portabile sau de salvare. Este o soluție scalabilă și extensibilă cu alte tipuri de senzori. ChatGPT a fost folosit pentru optimizarea fluxurilor logice și pentru suport în tratarea erorilor și testare modulară.

Analiza - design - implementare:

Analiza:

Obiectivul este realizarea unui dispozitiv de supraveghere activ, care detectează mișcarea și notifică utilizatorul prin email cu locația estimată. Dispozitivul trebuie să funcționeze în mod autonom și fiabil, cu logică de debounce și fără intervenție umană directă.

Design:

Dispozitivul este format din următoarele componente interconectate:

- Senzor PIR conectat la GPIO 17 – pentru detectarea mișcării;
- LED conectat la GPIO 27 – oferă feedback vizual;
- Modul GPS conectat serial la /dev/ttyAMA0 – oferă localizare în timp real prin mesaje NMEA;
- Raspberry Pi – gestionează întreaga logică de funcționare;
- Conexiune SMTP (Gmail) – pentru trimiterea automată a notificărilor prin email.

Implementare:

Codul este scris în Python și utilizează librăriile gpiozero, pynmea2, smtplib și datetime. La detecția mișcării, senzorul PIR declanșează aprinderea LED-ului și inițierea secvenței de colectare a datelor GPS. Coordonatele sunt transformate într-un link Google Maps, care este inclus într-un email automat trimis prin SMTP către o adresă prestabilită. Pentru evitarea trimiterii multiple de emailuri într-un interval scurt, se folosește un mecanism de debounce software cu datetime.now() și timedelta. Aplicația rulează continuu cu ajutorul pause() din signal, ascultând permanent evenimentele generate de senzor.

Mediu de dezvoltare: Dezvoltarea software s-a realizat pe Raspberry Pi rulând sistemul de operare Raspberry Pi OS. Codul Python a fost editat folosind [Thonny, cu SSH remote, dar am folosit și interfața grafică cu care vine acest model de raspberry pi].

Detalii secvență demonstrativă:

Testarea integrată a sistemului s-a realizat conform secvenței demonstrative prezentate în laborator, asigurându-se că toate modulele funcționează corect împreună.

La laborator a fost realizată următoarea demonstrație:

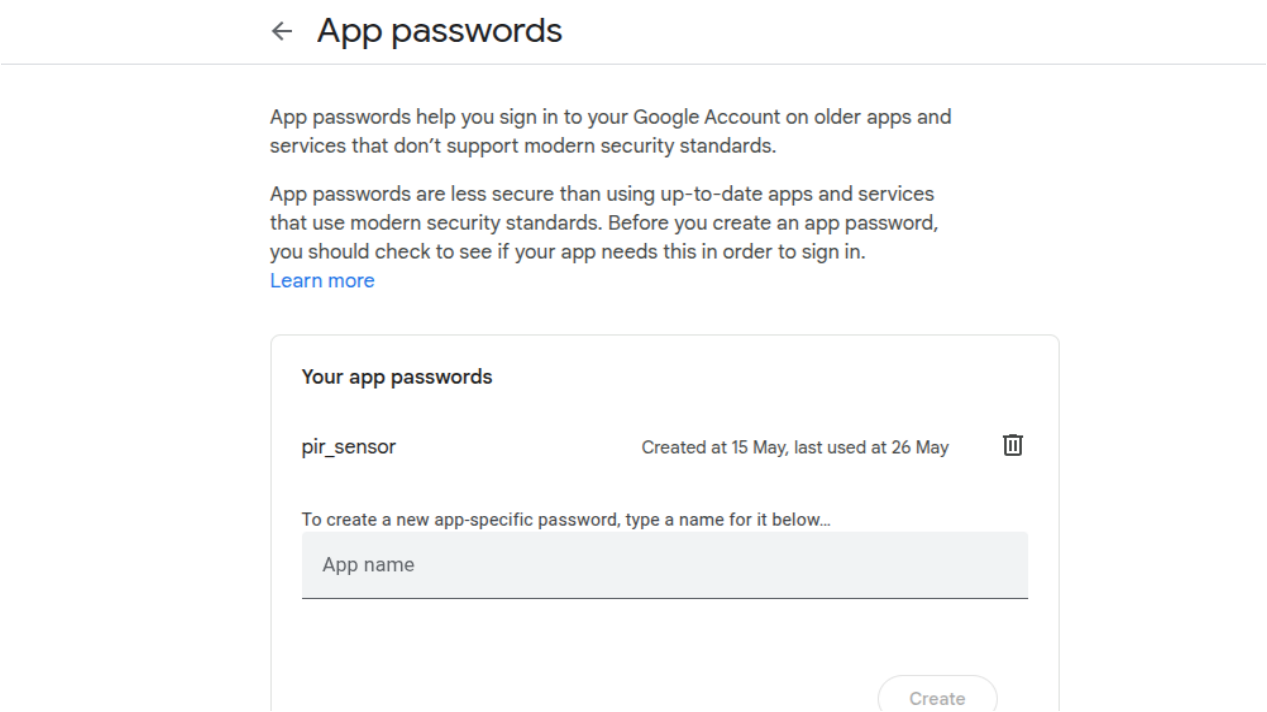
- *La mișcare detectată de senzorul PIR, LED-ul se aprinde.*
- *Modulul GPS returnează coordonatele.*
- *Emailul este transmis automat cu locația sub formă de link Google Maps.*
- *După un timp prestabilit (15s), sistemul se reactivează.*

***Modulul PIR și LED:** S-a verificat independent funcționarea corectă a detecției de mișcare și aprinderea/stingerea LED-ului.*

***Modulul GPS:** S-a testat separat capacitatea de a citi și parsa corect datele NMEA și de a genera link-ul Google Maps.*

***Modulul Email:** S-a testat trimiterea email-urilor cu un link generic pentru a valida configurarea SMTP.*

Pentru autentificarea la serverul SMTP al Gmail (`smtp.gmail.com`), scriptul Python utilizează un mecanism securizat de parole specifice aplicațiilor. În loc să se folosească parola principală a contului Google, s-a generat o "App Password" unică din setările contului Google (așa cum se poate observa în imaginea următoare:



Această parolă generată, care în script este stocată în variabila `PASSWORD = 'mcwrvuhlplbtceav'`, este apoi folosită împreună cu `USERNAME = 'danalacheemanuel@gmail.com'` în funcția `server.login(USERNAME, PASSWORD)` pentru a autentifica scriptul la serverul SMTP. Această metodă crește securitatea, deoarece parola principală a contului nu este expusă direct în cod sau stocată în aplicație și permite aplicațiilor care nu suportă standardele moderne de securitate (OAuth 2.0) să acceseze contul într-un mod controlat. Utilizarea unei "App Password" este de asemenea esențială dacă opțiunea "Accesul aplicațiilor mai puțin sigure" este dezactivată în contul Google, asigurând astfel funcționarea continuă a sistemului de notificare prin email."

Este important de menționat că această parolă specifică aplicației va fi revocată (ștearsă) din setările contului Google după finalizarea prezentării și evaluării proiectului, ca măsură suplimentară de securitate.

Probleme întâmpinate și modul de rezolvare:

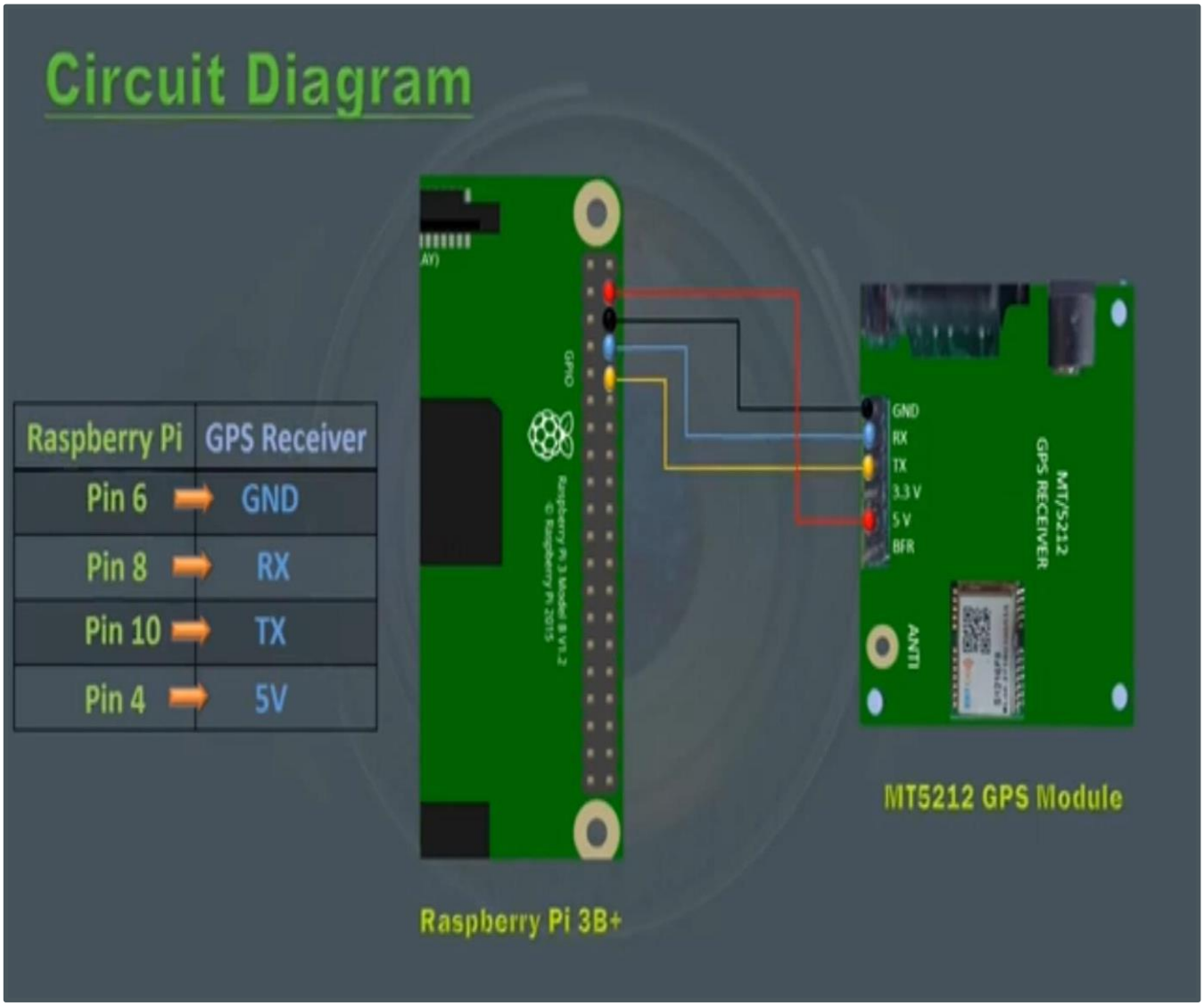
- Alimentarea insuficientă a modulului GPS → soluționată prin sursă externă stabilă de 3.3V.
- Lipsa semnalului GPS în interior → testarea s-a efectuat aproape de fereastră.
- Trimiteri multiple de emailuri → s-a introdus control cu datetime pentru debounce.
- Eșec la parsarea datelor GPS → s-a folosit try/except și librăria pynmea2 pentru robusteză.

Ce se învață dacă se replică proiectul:

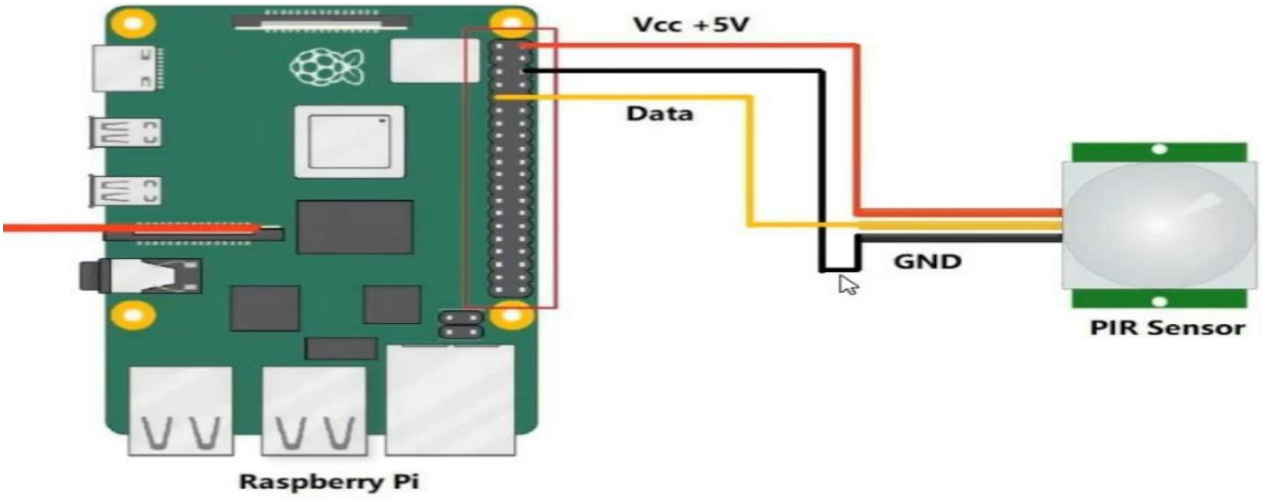
- Programarea GPIO pe Raspberry Pi cu gpiozero.
- Utilizarea modulelor GPS cu citire NMEA și serială.
- Tratarea și filtrarea datelor brute de locație.
- Implementarea de sisteme de notificare automată prin email.
- Programare asincronă și bazată pe evenimente.
- Managementul energiei, temporizărilor și prevenirea alertelor false.

Prezentare scheme si circuite hardware:

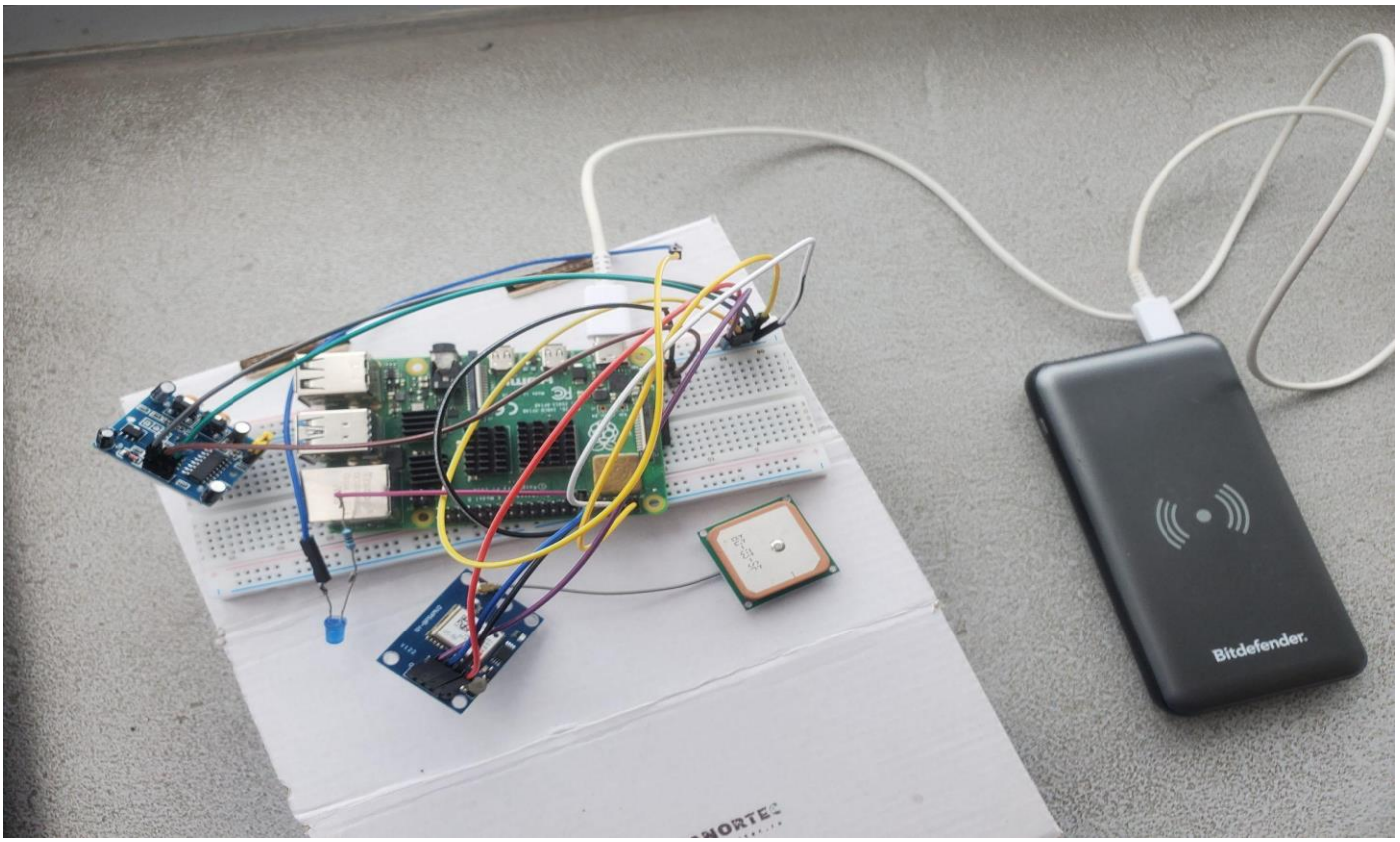
- ***Senzor PIR conectat la:***
  - ***VCC → Pin 2 (5V)***
  - ***GND → Pin 6 (GND)***
  - ***OUT (Data) → Pin 11 (GPIO 17)***
- ***LED conectat la GPIO 27 (pin fizic 13);***
- ***Modul GPS MT5212 conectat la:***
  - ***GND → Pin 6***
  - ***RX → Pin 8 (GPIO 14 - TXD)***
  - ***TX → Pin 10 (GPIO 15 - RXD)***
  - ***5V → Pin 4***



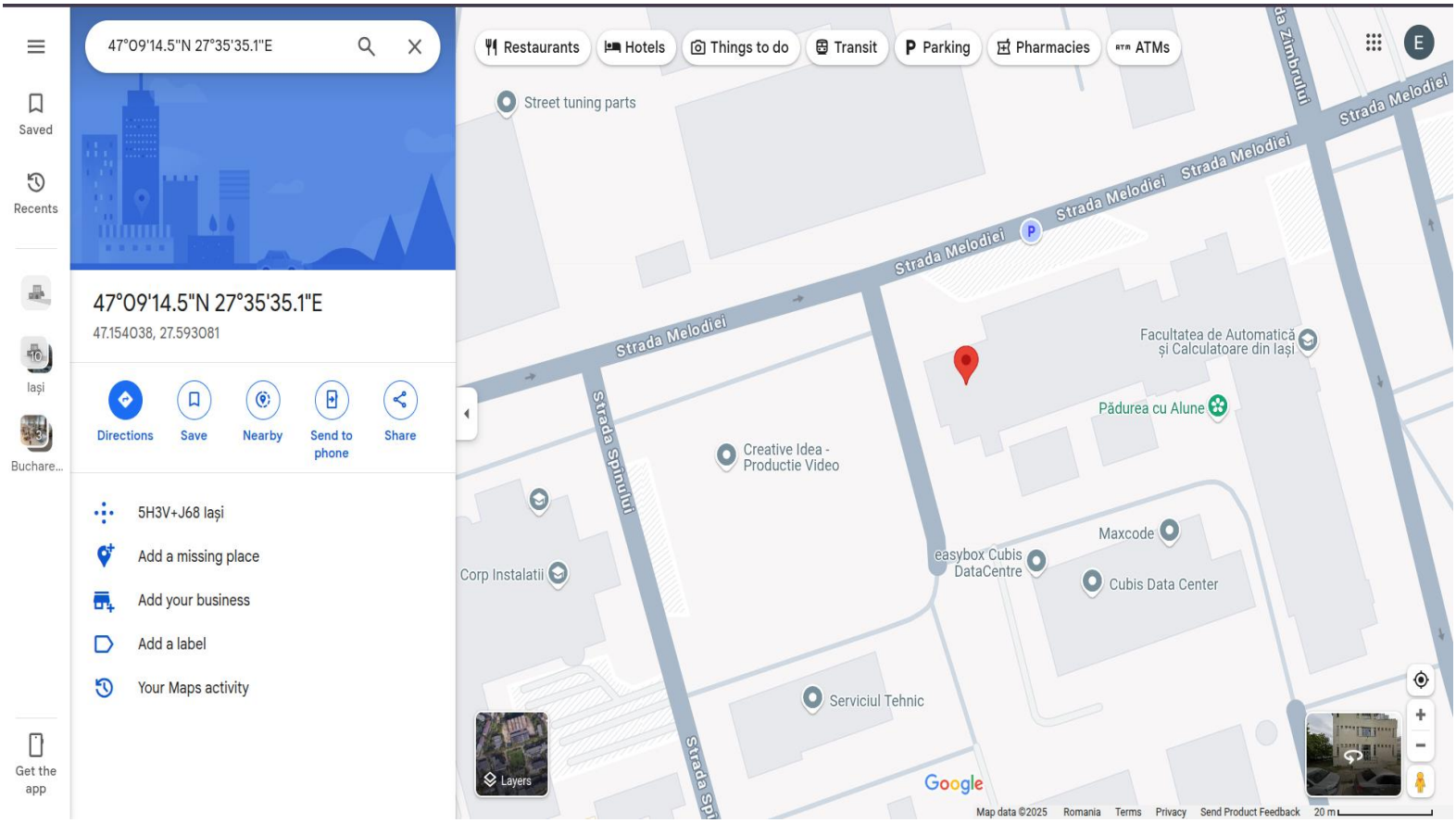
Hardware Schematic



PIR Pins	Raspberry Pi PIN
Vcc	2
GND	6
Data	11



Demonstrație:



Cod:

```
import time

import serial

import pynmea2

import smtplib

from email.mime.text import MIMEText

from gpiozero import MotionSensor, LED

from gpiozero.pins.rpigpio import RPiGPIOFactory

from gpiozero import Device

from datetime import datetime, timedelta

from signal import pause

import sys

# === CONFIGURARE GPIO ===

Device.pin_factory = RPiGPIOFactory()

pir = MotionSensor(17)    # PIR pe GPIO 17 (pin fizic 11)

led = LED(27)             # LED pe GPIO 27 (pin fizic 13)

# === CONFIGURARE EMAIL ===

SMTP_SERVER = 'smtp.gmail.com'

SMTP_PORT = 587

USERNAME = 'danalacheemanuel@gmail.com'

PASSWORD = 'mcwrvuhlplbtceav'

RECIEVER_EMAIL = 'emanuel.danalache@student.tuiasi.ro'

# === TIMP MINIM INTRE EMAILURI ===

last_motion_time = datetime.min
```

```
DEBOUNCE_SECONDS = 15

# === FUNCTIE PENTRU TRIMITERE EMAIL ===

def send_email(link):

    subject = 'Alerta: miscare detectata'

    body = f'Senzorul PIR a detectat miscare.\nLocatie estimata: {link}'

    msg = MIMEText(body)

    msg['Subject'] = subject

    msg['From'] = USERNAME

    msg['To'] = RECIEVER_EMAIL

    try:

        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:

            server.starttls()

            server.login(USERNAME, PASSWORD)

            server.send_message(msg)

            print("[EMAIL] Email trimis cu succes.")

    except Exception as e:

        print(f"[EMAIL] Eroare la trimitere: {e}")

# === FUNCTIE PENTRU OBTINERE LOCATIE GPS ===

def get_gps_link():

    try:

        with serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=1) as ser:

            for _ in range(15):  # Incearca 15 citiri (max ~15 sec)

                line = ser.readline().decode('ascii', errors='replace').strip()

                if line.startswith(('$GPGGA', '$GNGGA', '$GPRMC', '$GNRMC')):

                    try:
```

```
                msg = pynmea2.parse(line)
                if msg.latitude and msg.longitude:
                    latitude = msg.latitude
                    longitude = msg.longitude
                    link = f"https://maps.google.com/?q={latitude:.6f},{longitude:.6f}"
                    return link
            except pynmea2.ParseError:
                continue
    except Exception as e:
        print(f"[GPS] Eroare la citire: {e}")
    return None
# === FUNCTII PENTRU SENZOR ===
def motion_detected():
    global last_motion_time
    if datetime.now() - last_motion_time < timedelta(seconds=DEBOUNCE_SECONDS):
        return  # Ignora miscare daca e prea devreme

    last_motion_time = datetime.now()
    print("[SENZOR] Miscare detectata!")
    led.on()
    link = get_gps_link()
    if link:
        print(f"[GPS] Link Google Maps: {link}")
        send_email(link)
    else:
```

```
        print("[GPS] Nu s-a putut obține locația.")
def motion_stopped():
    print("[SENZOR] Miscare oprita.")
    led.off()
pir.when_motion = motion_detected
pir.when_no_motion = motion_stopped
# === FUNCTIE DE INCHIDERE CURATA ===
def clean_exit():
    led.off()
    print("LED oprit.")
    print("Conexiune inchisa.")
    sys.exit(0)
# === MAIN ===
print("Initializare sistem...")
print("Sistem activ. Asteptam detectie miscare...")
try:
    pause()  # asteapta evenimente de la senzor
except KeyboardInterrupt:
    print("\nProgram oprit manual.")
    clean_exit()
```

Bibliografie:

- ***https://gpiozero.readthedocs.io/***
- ***https://pynmea2.readthedocs.io/***
- ***https://realpython.com/python-send-email/***
- ***https://pinout.xyz/***
- ***https://chat.openai.com***
- ***https://www.youtube.com/watch?v=lDp4EfFCWvA***
- ***https://www.youtube.com/watch?v=OWP3D-51vlc***