

PROIECT SIOT

Monitorizare Litieră Pisică

1. Introducere

Acest proiect utilizează un **modul de greutate și un senzor de gaz** pentru a monitoriza activitatea pisicii în litieră, oferind informații despre tipul și frecvența utilizării acesteia.

Datele generate sunt transmise către **ThingSpeak**, o **platformă cloud** care permite stocarea, analiza și vizualizarea acestora prin grafice, precum și trimiterea de notificări automate în funcție de condiții predefinite.

Informațiile colectate includ cantitatea de reziduuri lichide și solide eliminate, durata petrecută în litieră și frecvența utilizării. **Analizând aceste date**, sistemul poate identifica tipul de activitate desfășurat de pisică, trimite **notificări** pentru curățarea litierei și ajută la **detectarea** eventualelor **probleme de sănătate**.

2. Descrierea soluției tehnice adoptate

Sistemul este construit pe baza unui **Raspberry Pi 3 Model B+**, la care sunt conectați un **modul de greutate** și un **senzor de gaz**. Aceste componente colectează date privind utilizarea litierei, care sunt prelucrate și transmise către **ThingSpeak**, o platformă IoT utilizată pentru stocarea și vizualizarea informațiilor.

Detectarea utilizării litierei

- **Intrare în litieră:** Sistemul detectează o creștere bruscă a greutății (~4 kg), corespunzătoare greutății pisicii.
- **Ieșire din litieră:** Sistemul detectează o scădere semnificativă a greutății, revenind aproape la valoarea inițială.

Determinarea tipului de activitate

- **Eliminare de reziduuri lichide (urină):**
 - Senzorul de gaz detectează prezența amoniacului.
 - Se înregistrează o greutate reziduală după ieșirea pisicii.
 - Cantitatea de urină se determină prin diferența dintre greutatea maximă detectată și greutatea pisicii (~4 kg).
- **Eliminare de reziduuri solide (fecale):**
 - Senzorul de gaz nu detectează prezența amoniacului.
 - Se înregistrează o greutate reziduală semnificativă după ieșirea pisicii.
 - Cantitatea de fecale se determină prin măsurarea greutății rămase în litieră.
- **Fără activitate semnificativă:**

- Senzorul de gaz nu detectează gaze.
- Nu există variații semnificative de greutate, ceea ce sugerează că pisica doar a intrat și a ieșit fără a lăsa reziduuri.

Utilizarea platformei ThingSpeak pentru stocarea și analiza datelor

Pentru stocarea și analiza datelor colectate, sistemul utilizează **ThingSpeak**, o platformă dedicată proiectelor IoT, care oferă un mediu accesibil pentru colectarea, procesarea și vizualizarea datelor în timp real.

Caracteristicile platformei ThingSpeak:

- Suportă trimiterea datelor prin **HTTP/MQTT**.
- Permite setarea de **alerte automate** în funcție de condiții predefinite.
- Oferă un **API flexibil** pentru extragerea și procesarea datelor.
- Versiunea gratuită permite utilizarea a **4 canale** și gestionarea a până la **3 milioane de mesaje pe an**.

Utilizarea datelor colectate:

- **Monitorizarea frecvenței utilizării litierei** pentru a detecta eventuale schimbări în comportamentul pisicii.
- **Evaluarea posibilelor probleme de sănătate**, analizând tiparele de utilizare a litierei.
- **Vizualizarea datelor sub formă de grafice** pentru interpretarea rapidă și ușoară a informațiilor.

Alarmer și notificări:

Pentru menținerea igienei litierei și a stării de sănătate a pisicii, sistemul trimite notificări automate prin e-mail atunci când:

- **Greutatea reziduală** indică acumularea semnificativă de reziduuri solide, sugerând necesitatea curățării litierei.

3. Modul de selecție și caracteristicile componentelor

1. **Raspberry Pi 3 Model B+** - Mini computer care coordonează senzorii și gestionează colectarea datelor.
2. **Modul de greutate:**
 - **Load Cell 5 kg** - Detectează modificările de greutate.
 - **Convertor HX711** - Amplifică semnalul de la senzor și permite conectarea la Raspberry Pi.
3. **Senzor de gaz MH MQ-135 Flying Fish** - Detectează gazele specifice, inclusiv amoniacul.

4. Punerea în funcțiune

Biblioteci necesare

Pentru funcționarea corectă a sistemului, sunt necesare următoarele biblioteci:

- **RPi.GPIO** pentru controlul pinilor Raspberry Pi.
- **hx711** pentru citirea datelor de la senzorul de greutate.
- **requests** pentru transmiterea datelor la ThingSpeak.
- **smtpplib** pentru trimiterea notificărilor prin e-mail.

Conectarea componentelor

Prin comanda pinout obținem harta pinilor de pe Raspberry PI:

```
daci@daci:~$ pinout
Description       : Raspberry Pi 3B+ rev 1.3
Revision          : a020d3
SoC               : BCM2837
RAM               : 1GB
Storage           : MicroSD
USB ports         : 4 (of which 0 USB3)
Ethernet ports    : 1 (300Mbps max. speed)
Wi-fi             : True
Bluetooth         : True
Camera ports (CSI): 1
Display ports (DSI): 1

-----
| 0000000000000000 J8 PoE |
| 1000000000000000 12 | USB |
| Wi |
| Fi | Pi Model 3B+ V1.3 | | | | |
| ID | SoC | 10 | RUN | USB |
| SI |  |  |  |  |  |
| IO |  |  |  |  |  |
| PMW | HDMI | 01 | | | Net |
|  |  |  |  |  |  |
-----

J8:
3V3 (1) (2) 5V
GPIO2 (3) (4) 5V
GPIO3 (5) (6) GND
GPIO4 (7) (8) GPIO14
GND (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
GND (25) (26) GPIO7
GPIO0 (27) (28) GPIO1
GPIO5 (29) (30) GND
GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
GND (39) (40) GPIO21

RUN:
POWER ENABLE (1)
RUN (2)

POE:
TR01 TAP (1) (2) TR00 TAP
TR03 TAP (3) (4) TR02 TAP

For further information, please refer to https://pinout.xyz/
daci@daci:~$
```

Conectam senzorii la Raspberry PI:

- **HX711:**
 - VCC (HX711) → pinul 1 (alimentam cu 3V)
 - GND (HX711) → pinul 6
 - DT (HX711) → la pinul 29
 - SCK (HX711) → GPIO6 (Pin 31)
- Load cell de 5 kg este conectat la HX711 astfel:

- Firul roșu → E+
 - Firul negru → E-
 - Firul albastru → A+
 - Firul alb → A-
- Senzor MQ-135:
 - VCC → 5V (Pin 2)
 - GND → GND (Pin 9)
 - DO → GPIO17 (Pin 11)

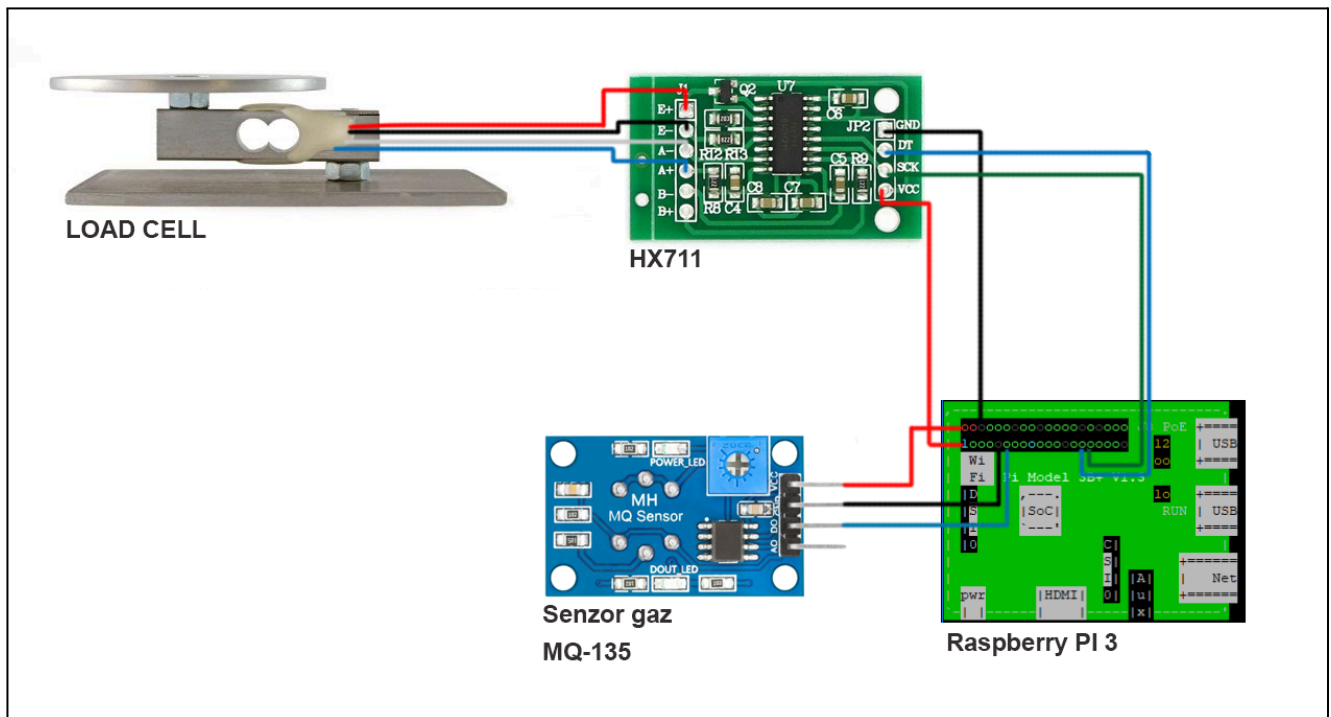


Fig1. Schema Monitor Litieră

Mod de instalare

Instalăm sistemul de operare Raspberry Pi OS 64-bit astfel:

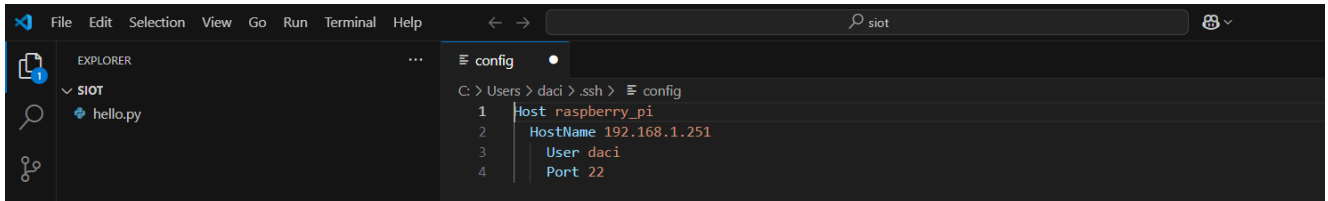
1. Descărcăm Appliance.
2. Rulăm Appliance.
3. Setăm următoarele opțiuni: utilizator, parolă, SSID Wi-Fi, activare SSH, setări de localizare.

Instalăm PuTTY și ne conectăm prin SSH la Raspberry Pi, folosind numele de utilizator și parola (daci, maya).

Verificăm versiunea de Python instalată pe Raspberry Pi:

```
daci@daci: ~
daci@daci:~ $ python --version
Python 3.11.2
daci@daci:~ $ python3 --version
Python 3.11.2
daci@daci:~ $
```

Instalăm SSH în Visual Studio Code și creăm conexiunea SSH: Add New SSH Host raspberry_pi.



Pentru a ne conecta prin SSH în VS Code, avem nevoie de fișierul C:\Users\daci.ssh\config, unde sunt specificate următoarele detalii: adresa IP a Raspberry Pi, portul și utilizatorul.

Exemplu de configurație:

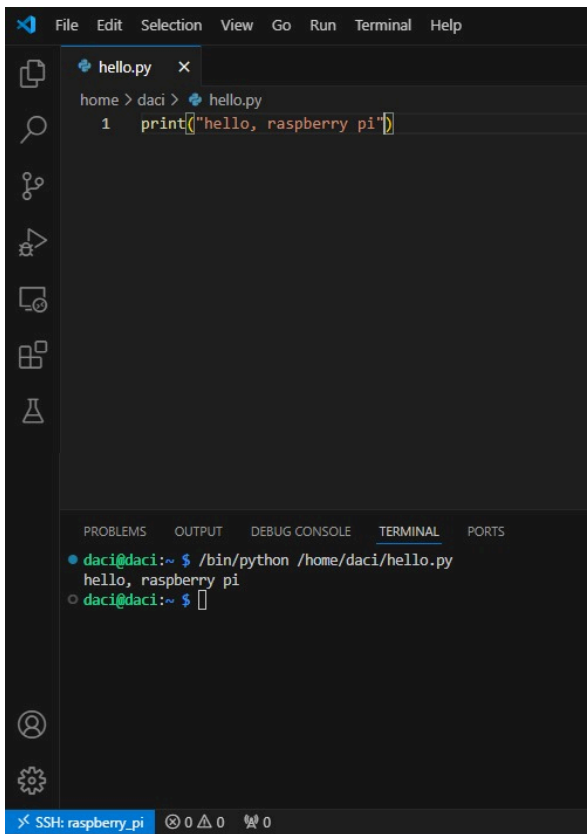
Host raspberry_pi

HostName 192.168.1.251 # Sau 192.168.128.146

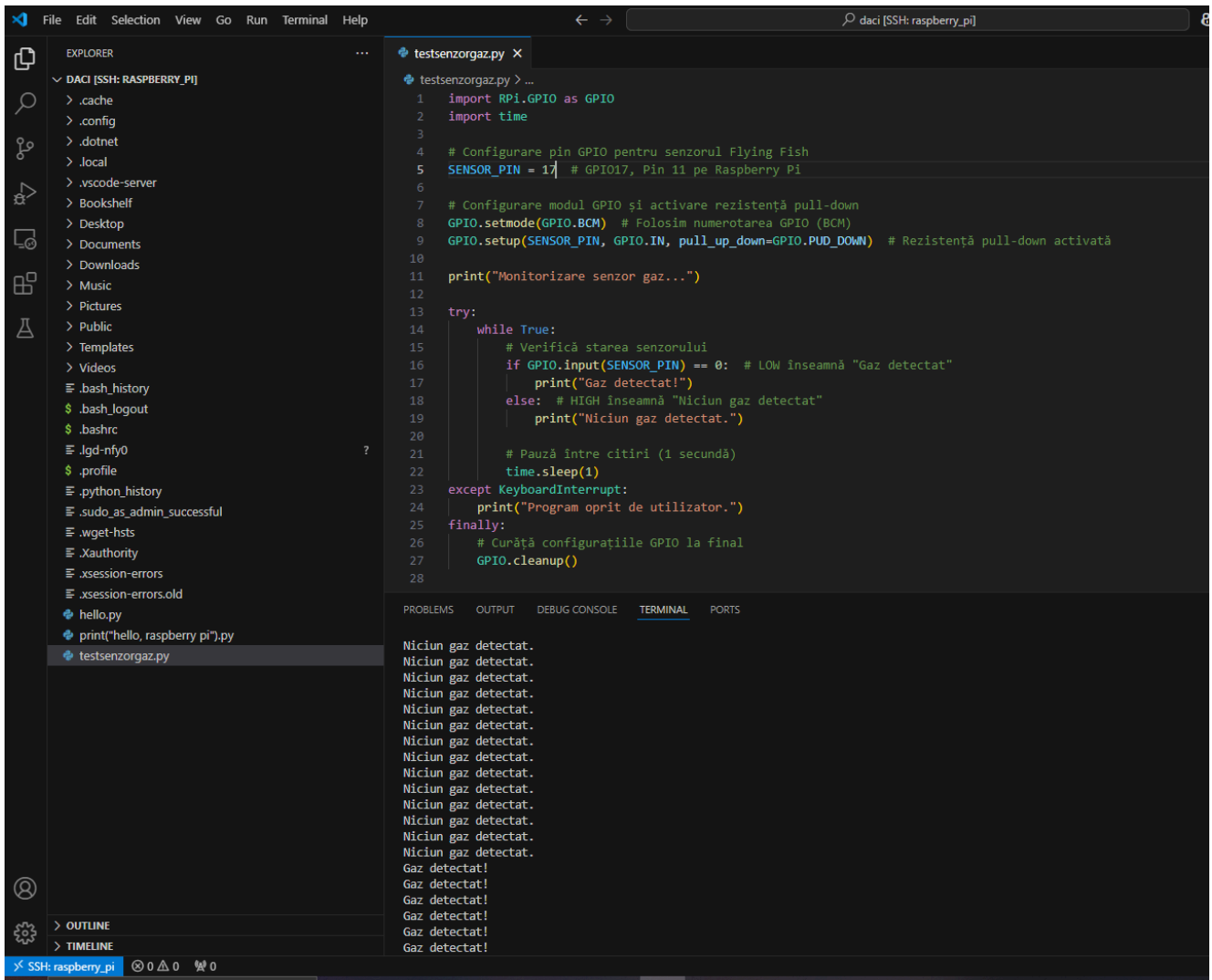
User daci

Port 22

După configurare, testăm codul din hello.py pentru a verifica conexiunea.



Conectăm senzorul de gaz MH MQ Flying Fish și testăm funcționalitatea acestuia. Când eliberăm gazul din brichetă, apare mesajul „Gaz detectat!”, confirmând funcționarea corectă a senzorului.



```
File Edit Selection View Go Run Terminal Help
daci [SSH: raspberry_pi]

EXPLORER
DACI [SSH: RASPBERRY_PI]
> .cache
> .config
> .dotnet
> .local
> .vscode-server
> Bookshelf
> Desktop
> Documents
> Downloads
> Music
> Pictures
> Public
> Templates
> Videos
.bash_history
.bash_logout
.bashrc
.lgd-nfy0
.profile
.python_history
.sudo_as_admin_successful
.wget-hsts
.Xauthority
.xsession-errors
.xsession-errors.old
hello.py
print("hello, raspberry pi").py
testsenzorgaz.py

testsenzorgaz.py X
testsenzorgaz.py > ...
1 import RPi.GPIO as GPIO
2 import time
3
4 # Configurare pin GPIO pentru senzorul Flying Fish
5 SENSOR_PIN = 17 # GPIO17, Pin 11 pe Raspberry Pi
6
7 # Configurare modul GPIO și activare rezistență pull-down
8 GPIO.setmode(GPIO.BCM) # Folosim numerotarea GPIO (BCM)
9 GPIO.setup(SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Rezistență pull-down activată
10
11 print("Monitorizare senzor gaz...")
12
13 try:
14     while True:
15         # Verifică starea senzorului
16         if GPIO.input(SENSOR_PIN) == 0: # LOW înseamnă "Gaz detectat"
17             print("Gaz detectat!")
18         else: # HIGH înseamnă "Niciun gaz detectat"
19             print("Niciun gaz detectat.")
20         # Pauză între citiri (1 secundă)
21         time.sleep(1)
22 except KeyboardInterrupt:
23     print("Program oprit de utilizator.")
24 finally:
25     # Curăță configurațiile GPIO la final
26     GPIO.cleanup()
27
28

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Niciun gaz detectat.
Gaz detectat!
Gaz detectat!
Gaz detectat!
Gaz detectat!
Gaz detectat!
Gaz detectat!
```

Conectăm senzorul de greutate și testăm funcționalitatea acestuia.
Creăm un mediu virtual și instalăm biblioteca HX711 folosind următoarele comenzi:

```
python3 -m venv siot_env
source siot_env/bin/activate

pip3 install 'git+https://github.com/gandalf15/HX711.git#egg=HX711&subdirectory=HX711_Python3'
```

Rulăm codul folosind: `/home/daci/siot_env/bin/python hx711_test.py` și observăm că senzorul funcționează corect:

The image shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'DACI [SSH: RASPBERRY_PI]' with various files and folders. The code editor shows a Python script named 'hx711_test.py' with the following code:

```
1 import time
2 import RPi.GPIO as GPIO
3 from hx711 import HX711
4
5 # Deactivează avertismentele GPIO
6 GPIO.setwarnings(False)
7
8 # Setează modul de numerotare al pinilor
9 GPIO.setmode(GPIO.BCM)
10
11 # Definirea pinilor pentru HX711
12 DT_PIN = 5 # GPIO5 (Pin 29)
13 SCK_PIN = 6 # GPIO6 (Pin 31)
14
15 # Inițializează HX711
16 hx = HX711(DT_PIN, SCK_PIN)
17
18 # Resetare HX711
19 hx.reset()
20
21 # PASUL 1: Măsoară valoarea brută fără greutate (tare)
22 print("[: Scoate toate obiectele de pe Load Cell... Așteaptă 5 secunde...")
23 time.sleep(5)
24 zero_offset = hx.get_raw_data_mean()
25 print(f"◆ Valoare offset (fără greutate): {zero_offset}")
26
27 # PASUL 2: Pune o greutate cunoscută (ex. 1 kg)
28 print("✖ Pune pe Load Cell un obiect cu greutate cunoscută (ex. 1kg). Așteaptă 5 secunde...")
```

The terminal output shows the execution of the script:

```
(siot_env) daci@daci:~ $ /home/daci/siot_env/bin/python hx711_test.py
[: Scoate toate obiectele de pe Load Cell... Așteaptă 5 secunde...
◆ Valoare offset (fără greutate): 74271
✖ Pune pe Load Cell un obiect cu greutate cunoscută (ex. 1kg). Așteaptă 5 secunde...
◆ Valoare brută cu greutate: 486217
✓ Factor de calibrare calculat: 411.946
✓ Sistem calibrat! Acum putem măsura greutatea în timp real.
[: Greutate măsurată: 1000.02 g
[: Greutate măsurată: 41.29 g
[: Greutate măsurată: 0.16 g
[: Greutate măsurată: 0.47 g
[: Greutate măsurată: 983.74 g
[: Greutate măsurată: 1000.23 g
[: Greutate măsurată: 1000.14 g
[: Greutate măsurată: 1000.17 g
[: Greutate măsurată: 1000.16 g
[: Greutate măsurată: 1000.24 g
[: Greutate măsurată: 1000.25 g
[: Greutate măsurată: 1021.53 g
```

5. Scenarii de funcționare

1. **Pisica intră și iese fără activitate** → Se înregistrează doar durata petrecută.
2. **Pisica elimină reziduuri lichide (urină):** → Se detectează amoniacul și se calculează cantitatea de urină.
3. **Pisica elimină reziduuri solide** → Nu se detectează amoniac, dar greutatea reziduală indică activitate.

6. Codul sursă

Pentru a putea salva datele citite în cloud, accesăm ThingSpeak și creăm un cont gratuit. Creăm un canal(tabel de date) pentru proiect.

Channel Settings

Percentage Complete

50%

Channel ID

2827075

Name

Monitorizare Litieră Pisică

Description

Monitorizarea activității litierei unei pisici folosind senzori.

Field 1

Greutate Maximă

☒

Field 2

Cantitate lichid

☒

Field 3

Cantitate solid

☒

Field 4

Tip activitate

☒

Field 5

Durata in litiera

☒

Field 6

Timestamp

☒

Structura tabelului de date:

Coloană	Tip	Descriere
greutate_maxima	REAL	Greutatea totală detectată (pisică + reziduuri).
cantitate_lichid	REAL	Cantitatea de urină lăsată de pisică (în grame).
cantitate_solid	REAL	Cantitatea de materie solidă lăsată de pisică (în grame).
tip_activitate	TEXT	Tipul activității: lichid, solid sau nimic.
durata	INTEGER	Timpul petrecut de pisică în litieră (în secunde).
timestamp	DATETIME	Ora și data la care a avut loc evenimentul (ex. 2025-02-03 14:30:00).

Cheia Write API Key pe care o vom folosi în scriptul Python pentru a trimite date este:

44N8QCMMCWGH220J

Codul pentru monitorizarea litierei, thingspeak_monitor.py:

```

import time
import requests
import smtplib # 📧 Import pentru trimitere email
import RPi.GPIO as GPIO
from hx711 import HX711
import datetime
    
```



```

from email.mime.text import MIMEText # 📧 Formatare mesaj email
from email.mime.multipart import MIMEMultipart # 📧 Email cu subiect

# ♦ Configurare Email (Gmail SMTP)
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_SENDER = "daci.draghia@gmail.com" # 📧 Înlocuiește cu emailul tău Gmail
EMAIL_PASSWORD = "ujzl djbj bhlp ftvt" # 📧 Parola aplicației Gmail
EMAIL_RECEIVER = "daci.draghia@gmail.com" # 📧 Email-ul unde vrei să primești notificarea

# ♦ Funcție pentru trimiterea notificărilor prin email
def send_email(subject, body):
    msg = MIMEMultipart()
    msg["From"] = EMAIL_SENDER
    msg["To"] = EMAIL_RECEIVER
    msg["Subject"] = subject
    msg.attach(MIMEText(body, "plain"))

    try:
        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls() # 📧 Activează conexiunea securizată
        server.login(EMAIL_SENDER, EMAIL_PASSWORD)
        server.sendmail(EMAIL_SENDER, EMAIL_RECEIVER, msg.as_string())
        server.quit()
        print("📧 Email trimis cu succes!")
    except Exception as e:
        print(f"⚠️ Eroare la trimiterea emailului: {e}")

# ♦ Dezactivează avertismentele GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# ♦ Configurare HX711 (senzor greutate)
DT_PIN = 5
SCK_PIN = 6
hx = HX711(DT_PIN, SCK_PIN)
hx.reset()

# ♦ Configurare ThingSpeak
THINGSPEAK_WRITE_KEY = "44N8QCMMCWGH220J"
THINGSPEAK_URL = "https://api.thingspeak.com/update"

# ♦ Configurare senzor gaz (MQ-135 - Flying Fish)
SENSOR_PIN = 17
GPIO.setup(SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
print("🔍 Monitorizare senzor gaz...")

```

```

# ♦ Variabile globale pentru calibrare
zero_offset = 0
calibration_factor = 1

# ♦ PASUL 1: Măsoară valoarea brută fără greutate (tare)
print("⚖️ Scoate toate obiectele de pe Load Cell... Așteaptă 5 secunde...")
time.sleep(5)
zero_offset = hx.get_raw_data_mean()
print(f"♦ Valoare offset (fără greutate): {zero_offset}")

# ♦ PASUL 2: Pune o greutate cunoscută (ex. 1 kg)
print("🔴 Pune pe Load Cell un obiect cu greutate cunoscută (ex. 1kg). Așteaptă 5 secunde...")
time.sleep(5)
measured_value = hx.get_raw_data_mean()
print(f"📊 Valoare brută cu greutate: {measured_value}")

# ♦ Calculează factorul de conversie (scale factor)
known_weight = 1000 # Greutatea cunoscută (în grame)
calibration_factor = (measured_value - zero_offset) / known_weight
print(f"✅ Factor de calibrare calculat: {calibration_factor}")

print("✅ Sistem calibrat! Acum putem măsura greutatea în timp real.")

# ♦ Funcție pentru citirea datelor de la senzori
def read_sensors():
    global zero_offset, calibration_factor # 🔴 Adăugăm variabilele globale

    # Citire greutate (HX711)
    raw_value = hx.get_raw_data_mean()
    weight = (raw_value - zero_offset) / calibration_factor # Convertim în grame

    # Citire senzor gaz (MQ-135)
    gas_detected = GPIO.input(SENSOR_PIN)
    gas_value = 0 if gas_detected == 1 else 100 # LOW = Gaz detectat, HIGH = Fără gaz

    return weight, gas_value

# ♦ Funcție pentru trimiterea datelor la ThingSpeak
def send_to_thingspeak(weight, gas_value, activity_type, duration, cant_pipi, cant_caca, timestamp):
    data = {
        "api_key": THINGSPEAK_WRITE_KEY,
        "field1": weight,
        "field2": cant_pipi,
        "field3": cant_caca,
        "field4": activity_type,
        "field5": duration,
        "field6": timestamp
    }

```

```

}
response = requests.post(THINGSPEAK_URL, data=data)
return response.status_code

try:
    while True:
        print("🔔 Monitorizăm activitatea...")
        start_time = time.time()

        weight, gas_value = read_sensors()
        print(f"Greutate detectată: {weight:.2f} g, Gaz: {gas_value}%")

        if weight > 1000:
            time.sleep(10)
            residue_weight = weight - 1000
            activity_type = "nimic"
            cant_pipi = 0
            cant_caca = 0
            timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            if gas_value > 50 and residue_weight > 15:
                activity_type = "lichid"
                cant_pipi = residue_weight

            elif gas_value < 50 and residue_weight > 15:
                activity_type = "solid"
                cant_caca = residue_weight

            # 📧 Trimite email pentru curățare litieră
            email_subject = "💣 Atenție! Pisica a făcut caca!"
            email_body = f"Pisica a făcut caca!\nGreutate fecale: {cant_caca}g\nTimestamp: {timestamp}\nCurăță
litiera!"

            send_email(email_subject, email_body)

            status = send_to_thingspeak(weight, gas_value, activity_type, int(time.time() - start_time), cant_pipi,
cant_caca, timestamp)
            if status == 200:
                print(f"📡 Date trimise cu succes la ThingSpeak! Activitate: {activity_type}, Pipi: {cant_pipi}g, Caca:
{cant_caca}g, Timestamp: {timestamp}")
            else:
                print("⚠️ Eroare la trimiterea datelor.")

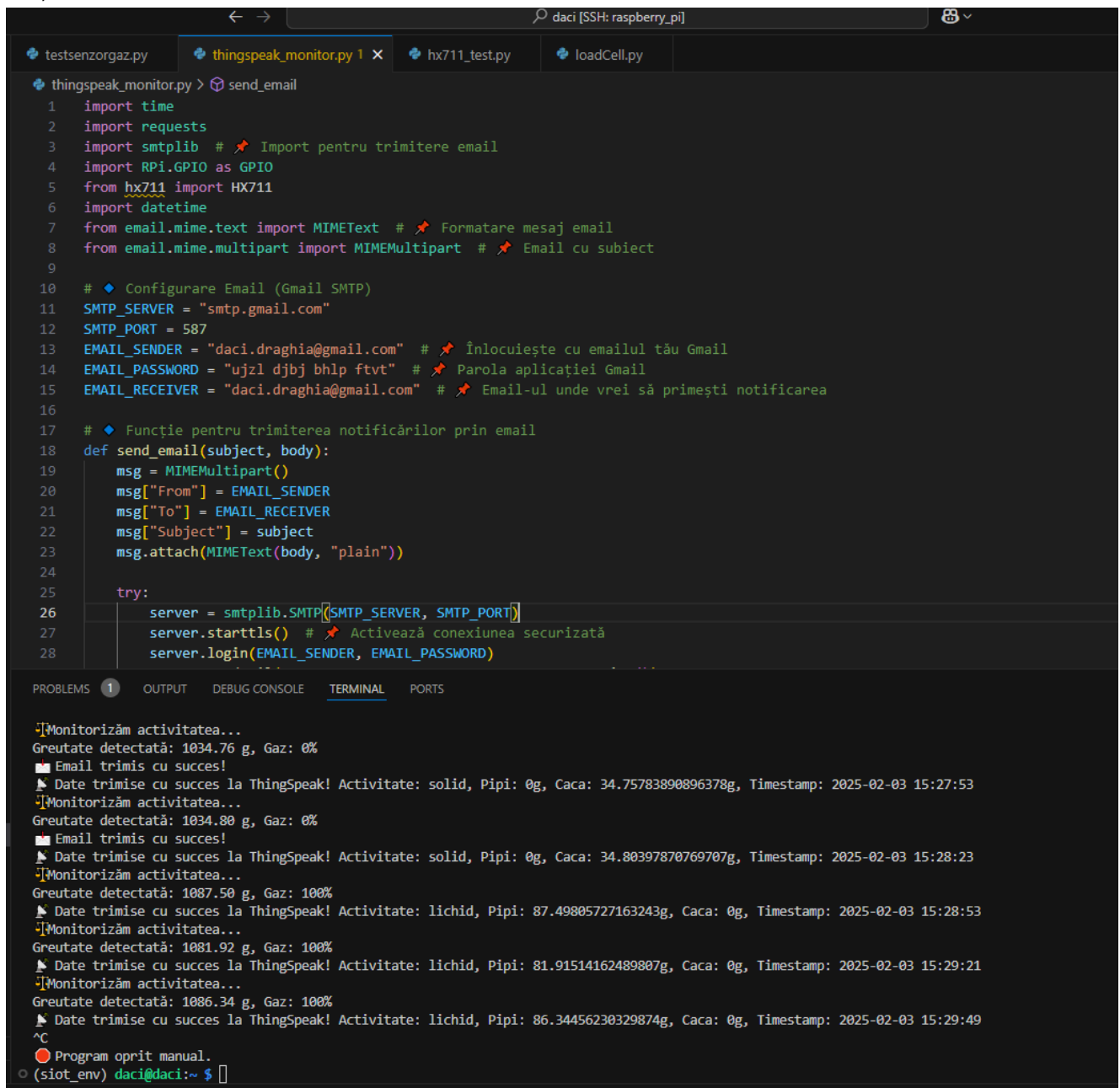
            time.sleep(15)

except KeyboardInterrupt:
    print("\n🛑 Program oprit manual.")
finally:

```

```
GPIO.cleanup()
```

Obținem:



```
thingspeak_monitor.py > send_email
1 import time
2 import requests
3 import smtplib # Import pentru trimitere email
4 import RPi.GPIO as GPIO
5 from hx711 import HX711
6 import datetime
7 from email.mime.text import MIMEText # Formatare mesaj email
8 from email.mime.multipart import MIMEMultipart # Email cu subiect
9
10 # Configurare Email (Gmail SMTP)
11 SMTP_SERVER = "smtp.gmail.com"
12 SMTP_PORT = 587
13 EMAIL_SENDER = "daci.draghia@gmail.com" # Înlocuiește cu emailul tău Gmail
14 EMAIL_PASSWORD = "ujzl djbj bhlp ftvt" # Parola aplicației Gmail
15 EMAIL_RECEIVER = "daci.draghia@gmail.com" # Email-ul unde vrei să primești notificarea
16
17 # Funcție pentru trimiterea notificărilor prin email
18 def send_email(subject, body):
19     msg = MIMEMultipart()
20     msg["From"] = EMAIL_SENDER
21     msg["To"] = EMAIL_RECEIVER
22     msg["Subject"] = subject
23     msg.attach(MIMEText(body, "plain"))
24
25     try:
26         server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
27         server.starttls() # Activează conexiunea securizată
28         server.login(EMAIL_SENDER, EMAIL_PASSWORD)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Monitorizăm activitatea...
Greutate detectată: 1034.76 g, Gaz: 0%
Email trimis cu succes!
Date trimise cu succes la ThingSpeak! Activitate: solid, Pipi: 0g, Caca: 34.75783890896378g, Timestamp: 2025-02-03 15:27:53
Monitorizăm activitatea...
Greutate detectată: 1034.80 g, Gaz: 0%
Email trimis cu succes!
Date trimise cu succes la ThingSpeak! Activitate: solid, Pipi: 0g, Caca: 34.80397870769707g, Timestamp: 2025-02-03 15:28:23
Monitorizăm activitatea...
Greutate detectată: 1087.50 g, Gaz: 100%
Date trimise cu succes la ThingSpeak! Activitate: lichid, Pipi: 87.49805727163243g, Caca: 0g, Timestamp: 2025-02-03 15:28:53
Monitorizăm activitatea...
Greutate detectată: 1081.92 g, Gaz: 100%
Date trimise cu succes la ThingSpeak! Activitate: lichid, Pipi: 81.91514162489807g, Caca: 0g, Timestamp: 2025-02-03 15:29:21
Monitorizăm activitatea...
Greutate detectată: 1086.34 g, Gaz: 100%
Date trimise cu succes la ThingSpeak! Activitate: lichid, Pipi: 86.34456230329874g, Caca: 0g, Timestamp: 2025-02-03 15:29:49
^C
Program oprit manual.
(siot_env) daci@daci:~ $
```

Am generat date de intrare:

A	B	C	D	E	F	G	H	I	J	K
178	2025-02-03T13:03:28+00:00	177	1.001.039.286.486.110	0	0	0	nimic	12	03/02/2025	15:03
179	2025-02-03T13:03:56+00:00	178	10.008.838.791.610.900	0	0	0	nimic	12	03/02/2025	15:03
180	2025-02-03T13:04:24+00:00	179	1.000.845.027.329.830	0	0	0	nimic	12	03/02/2025	15:04
181	2025-02-03T13:04:52+00:00	180	10.008.620.250.060.000	0	0	0	nimic	12	03/02/2025	15:04
182	2025-02-03T13:05:20+00:00	181	10.009.882.934.575.900	0	0	0	nimic	12	03/02/2025	15:05
183	2025-02-03T13:05:48+00:00	182	1.000.947.013.386.880	0	0	0	nimic	12	03/02/2025	15:05
184	2025-02-03T13:06:16+00:00	183	10.010.489.994.439.300	0	0	0	nimic	12	03/02/2025	15:06
185	2025-02-03T13:06:44+00:00	184	10.009.712.957.814.100	0	0	0	nimic	12	03/02/2025	15:06
186	2025-02-03T13:07:12+00:00	185	10.009.810.087.392.300	0	0	0	nimic	12	03/02/2025	15:07
187	2025-02-03T13:07:40+00:00	186	1.000.845.027.329.830	0	0	0	nimic	12	03/02/2025	15:07
188	2025-02-03T13:08:08+00:00	187	10.008.814.509.216.300	0	0	0	nimic	12	03/02/2025	15:08
189	2025-02-03T13:08:37+00:00	188	10.009.567.263.446.900	0	0	0	nimic	12	03/02/2025	15:08
190	2025-02-03T13:09:05+00:00	189	10.010.004.346.548.600	0	0	0	nimic	12	03/02/2025	15:09
191	2025-02-03T13:09:33+00:00	190	1.001.097.564.233.000	0	0	0	nimic	12	03/02/2025	15:09
192	2025-02-03T13:10:01+00:00	191	10.010.077.193.732.200	0	0	0	nimic	12	03/02/2025	15:10
193	2025-02-03T13:18:28+00:00	192	10.641.367.486.498.000	0	6.413.674.864.980.380	solid		14	03/02/2025	15:18
194	2025-02-03T13:18:56+00:00	193	10.472.398.686.715.600	4.723.986.867.156.230	0	lichid		12	03/02/2025	15:18
195	2025-02-03T13:19:24+00:00	194	10.216.079.962.699.600	2.160.799.626.996.160	0	lichid		12	03/02/2025	15:19
196	2025-02-03T13:19:52+00:00	195	10.529.830.594.086.300	5.298.305.940.863.360	0	lichid		12	03/02/2025	15:19
197	2025-02-03T13:20:20+00:00	196	10.269.165.015.347.500	26.916.501.534.755.500	0	lichid		12	03/02/2025	15:20
198	2025-02-03T13:20:50+00:00	197	1.047.380.716.478.220	0	47.380.716.478.221.900	solid		14	03/02/2025	15:20
199	2025-02-03T13:21:18+00:00	198	10.001.772.739.635.500	0	0	nimic		12	03/02/2025	15:21
200	2025-02-03T13:21:46+00:00	199	10.003.521.195.166.400	0	0	nimic		12	03/02/2025	15:21
201	2025-02-03T13:22:14+00:00	200	10.646.151.455.103.500	6.461.514.551.035.470	0	lichid		12	03/02/2025	15:22
202	2025-02-03T13:22:42+00:00	201	10.749.480.320.161.600	7.494.803.201.616.350	0	lichid		12	03/02/2025	15:22
203	2025-02-03T13:23:11+00:00	202	106.118.380.153.087	61.183.801.530.869.900	0	lichid		13	03/02/2025	15:23
204	2025-02-03T13:23:41+00:00	203	10.533.934.607.763.100	0	53.393.460.776.314.200	solid		14	03/02/2025	15:23
205	2025-02-03T13:24:09+00:00	204	10.006.071.026.149.100	0	0	nimic		12	03/02/2025	15:24
206	2025-02-03T13:24:37+00:00	205	10.490.538.912.849.200	4.905.389.128.492.060	0	lichid		12	03/02/2025	15:24
207	2025-02-03T13:25:05+00:00	206	10.036.911.838.986.600	0	0	nimic		12	03/02/2025	15:25
208	2025-02-03T13:25:33+00:00	207	11.177.973.345.766.700	11.779.733.457.667.900	0	lichid		12	03/02/2025	15:25
209	2025-02-03T13:26:02+00:00	208	10.015.007.576.640.600	0	0	nimic		12	03/02/2025	15:26
210	2025-02-03T13:26:30+00:00	209	10.007.066.674.437.500	0	0	nimic		12	03/02/2025	15:26
211	2025-02-03T13:26:57+00:00	210	10.876.753.312.351.800	8.767.533.123.518.680	0	lichid		12	03/02/2025	15:26
212	2025-02-03T13:27:26+00:00	211	10.347.651.241.403.400	3.476.512.414.034.270	0	lichid		12	03/02/2025	15:27
213	2025-02-03T13:27:55+00:00	212	10.347.578.389.089.600	0	3.475.783.890.896.370	solid		14	03/02/2025	15:27
214	2025-02-03T13:28:25+00:00	213	1.034.803.978.707.690	0	3.480.397.870.769.700	solid		14	03/02/2025	15:28
215	2025-02-03T13:28:53+00:00	214	10.874.980.572.716.300	8.749.805.727.163.240	0	lichid		12	03/02/2025	15:28
216	2025-02-03T13:29:21+00:00	215	1.081.915.141.624.890	8.191.514.162.489.800	0	lichid		12	03/02/2025	15:29
217	2025-02-03T13:29:49+00:00	216	10.863.445.623.032.900	8.634.456.230.329.870	0	lichid		12	03/02/2025	15:29
218										
219										

Pentru a putea vizualiza datele sub formă de grafice vom scrie un cod MATLAB care ruleaza in platforma ThingSpeak. Codul:

1. **Citește ultimele 25 de înregistrări** din canalul ThingSpeak.
2. **Grupează datele în 5 zile** (fiecare zi conținând 5 înregistrări).
3. **Afișează un grafic** care arată cantitatea de lichide si solide pe zi.

% Configurare

channelID = 2827075; % ID-ul canalului

readAPIKey = '44N8QCMWCWGH220J'; % Cheia API de citire

% Citește ultimele 25 de puncte de date din ThingSpeak

data = thingSpeakRead(channelID, ...

'Fields', [2, 3], ... % Field 2 = CantitateLichid (pipi), Field 3 = CantitateSolid (caca)

'NumPoints', 25, ... % Ultimele 25 de puncte

'OutputFormat', 'table', ...

'ReadKey', readAPIKey);

% Extrage câmpurile relevante

pipi = data.CantitateLichid; % Field 2 - Cantitate pipi

caca = data.CantitateSolid; % Field 3 - Cantitate caca

% Definire grupare pe zile (5 zile, fiecare având câte 5 măsurători)

numDays = 5;

groupSizes = repelem(1:numDays, 5); % Creează 5 grupuri a câte 5 înregistrări

% Calculează suma cantității de pipi și caca pe fiecare zi

dailyPipi = accumarray(groupSizes, pipi, [], @sum);

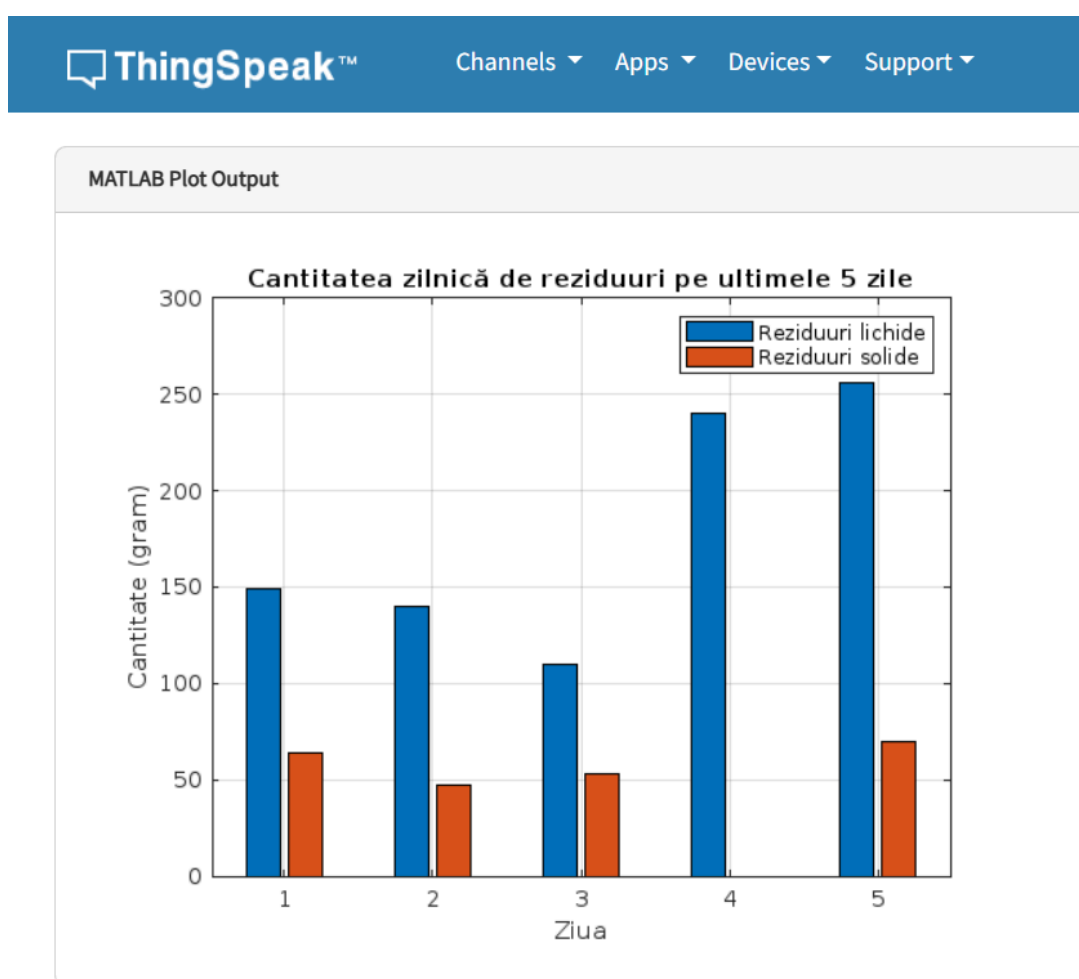
dailyCaca = accumarray(groupSizes, caca, [], @sum);

```
% Grafic cu datele grupate pe zile
figure;
bar(1:numDays, [dailyPipi, dailyCaca], 'grouped');
xlabel('Ziua');
ylabel('Cantitate (gram)');
legend({'Reziduuri lichide', 'Reziduuri solide'});
title('Cantitatea zilnică de reziduuri pe ultimele 5 zile');
grid on;
```

7. Rezultate experimentale

Datele sunt colectate și analizate folosind MATLAB pe ThingSpeak:

- Ultimele 25 de înregistrări sunt grupate în 5 zile (câte 5 înregistrări pe zi).
- Se generează grafice care arată cantitatea zilnică de reziduuri lichide și solide.



Se trimite notificare pe email pentru a semnală necesitatea curățării literei:



me

🚨 **Atenție! Pisica a făcut caca!** - Pisica a făcut caca! Greutate fecale: 33.7646696015986g Tim...

me

🚨 **Atenție! Pisica a făcut caca!** - Pisica a făcut caca! Greutate fecale: 26.533372508929688g T...



Atenție! Pisica a făcut caca!  Inbox 

daci.draghia@gmail.com

to me ▾

Pisica a făcut caca!

Greutate fecale: 26.533372508929688g

Timestamp: 2025-02-03 12:29:49

Curăță litiera!

8. Concluzii

Acest proiect demonstrează utilitatea senzorilor IoT în monitorizarea activității pisicilor în litieră, oferind o soluție automatizată pentru colectarea și analiza datelor. Sistemul permite identificarea tipului de activitate, frecvenței utilizării și poate contribui la detectarea timpurie a eventualelor probleme de sănătate ale pisicii.

Pentru îmbunătățirea și extinderea funcționalităților, proiectul poate fi dezvoltat în următoarele direcții:

- Integrarea cu aplicații mobile pentru o monitorizare facilă și notificări în timp real.
- Optimizarea platformei de vizualizare a datelor, astfel încât să ofere informații mai relevante și interpretări mai intuitive pentru utilizatori.

Prin implementarea acestor îmbunătățiri, sistemul poate deveni un instrument esențial pentru proprietarii de pisici, facilitând menținerea igienei litierii și monitorizarea stării de sănătate a animalului într-un mod simplu și eficient.