QUBE2SPACE

RAPORT TEHNIC PRELIMINAR

Raport întocmit de:	**Matei Sîrghe**

Echipă

Nume echipă:	NoDynamics
Lider:	Matei Sirghe
Membri echipă:	Boanta Mihai Mara Matei
	Vasilescu Claudia
	Sîrghe Matei
Profesor coordonator:	—

Abstract

Exteriorul PocketQube-ului nostru va fi de dimensiuni 5 5 5 cm, construit din plastic ușor dar durabil, prin intermediul imprimării 3D.

Misiunea principala a PocketQube-ului este măsurarea presiunii atmosferice prin intermediul unui barometru, cât și a accelerației folosind un accelerometru. Misiunea secundară este mai ambițioasă: se dorește constatarea condițiilor meteorologice din proximitatea rachetei, dar și monitorizarea poluării, în vederea stabilirii pericolului pe care îl prezintă și a localizării poluanților.

Pe lângă microprocesor și senzorii menționați anterior, pentru a putea îndeplini cu succes a doua misiune, se remarcă necesitatea unui GPS și a unor dispozitive precum: termometru, umidometru, senzor de CO2 și gaze inflamabile.

Introducere

Pentru mulți, poluarea este o umbră de incertitudine planând la limita înțelegerii lor, o îngrijorare îndepărtată care, în lipsa unui pericol iminent, este deseori ignorată. Nu și în cazul nostru. Ne-am dat seama de amenințarea pe care o prezinta poluarea la adresa planetei Terra și am decis să facem ceva în această privință.

Am ajuns deci cu ușurință la misiunea secundară potrivită nouă: studiul CO_2, și, în general, sursele acestuia, în speranța că rezultatele vor contribui la înțelegerea poluării și că vor ajuta la combaterea ei. Viziunea noastră constă, în mare, în conștientizarea primejdiei pe care o reprezintă. Suntem patru adolescenți care vor să facă ceva în privința aceasta, uniți de pasiunea pentru electronică.

Având experiența atât a proiectelor de la orele de fizică sau informatică, cât și a activităților extrașcolare și a concursurilor naționale, sperăm să aducem ceva inovativ.

Misiune principală

Se măsoară pe tot parcursul zborului:

- Presiunea atmosferică măsurată cu un barometru;
- Accelerația pe 3 axe aflată cu un ajutorul unui accelerometru;

Misiune secundară

Misiunea secundară a echipei noastre este:

- Monitorizarea poluarii impreuna cu gasirea locatiei aproximative a surseii, utilizând datele aflate pe parcursul traiectoriei;

- Starea meteorologică din zona rachetei.

Modalitățile propuse privind implementarea funcționalităților

Misiunea principala:

- Barometrul și accelerometrul vor fi conectati de microcontroller Raspberry PI. Datele vor fi procesate și apoi transmise la cardul SD.(Nu este nevoie de convertor analog-digital pentru senzori).

Misiunea secundară:

1. Locația zonei care emana poluare.
- Senzorul de CO2 și gaze inflamabile măsoară concentrația de gaze de seră sau nocive din atmosfera pe parcursul zborului.
- Utilizând GPS, vom știi unde ne aflăm exact și putem salva locația experimentului pentru calculele de mai târziu.
- Următoarele date le putem trimite unui algoritm care să găsească locația aproximativă. Algoritmul ar lua în calcul concentratia normala de CO2 din atmosferă la diferite altitudini. Datele obtinute de către senzor ar fi înmulțite corespunzător cu concentrația normala ca sa avem date mai concrete care nu sunt influențate în mod negativ de schimbările rapide de altitudine.
- La graficul care iese din prima parte a algoritmului utilizăm direcția vantului ca sa știm direcția de unde vine poluarea.Cu ajutorul datelor de pe internet despre cum se comporta vantul din zona în care lansăm, putem sa aproximam cât a parcurs aerul nociv din atmosferă până a ajuns pe traiectoria rachetei noastre.
- Știind direcția și distanța fata de punctul de concentrație maximă de dioxid de carbon de pe graficul de la început, împreună cu datele GPS putem afla locatia din care provine poluarea și sa o calculăm la final după ce recuperăm cubul, sau în timp real utilizând emițătorul de la bord.

2. Date meteorologice despre zona în care am lansat racheta.
- Folosim umidometru pentru a afla cat de umedă este atmosfera.
- Vom utiliza și un senzor de raze uv pentru mai multe rezultate meteorologice.
- Alfam și temperatura exacta pe parcursul zborului.(Utilizăm și presiunea obținută de către barometru)
- Utilizand un algoritm care folosește aceste date, putem prezice vremea și să aflăm dacă urmează sa se intample un eveniment meteorologic important.

Echipamente necesare

Componentele și instrumentele utilizate de către echipa noastră sunt următoarele :

- Microprocesor:
- Umidometru
- Termometru
- Barometru
- ADC(Analog to Digital Converter)
- Senzor de Gaze
- Accelerometru
- GPS
- Transmițător
- Breadboard
- Rezistori(10k ,1k ,470 Ohm)
- SD card
- Cabluri

Etapele necesare realizării proiectului

Părțile sunt cumpărate de pe link-urile puse mai sus. Vom utiliza calculatoarele noastre ca sa scriem algoritmul necesar pentru a realiza graficele cu datele finale.

Masuri de precautie:

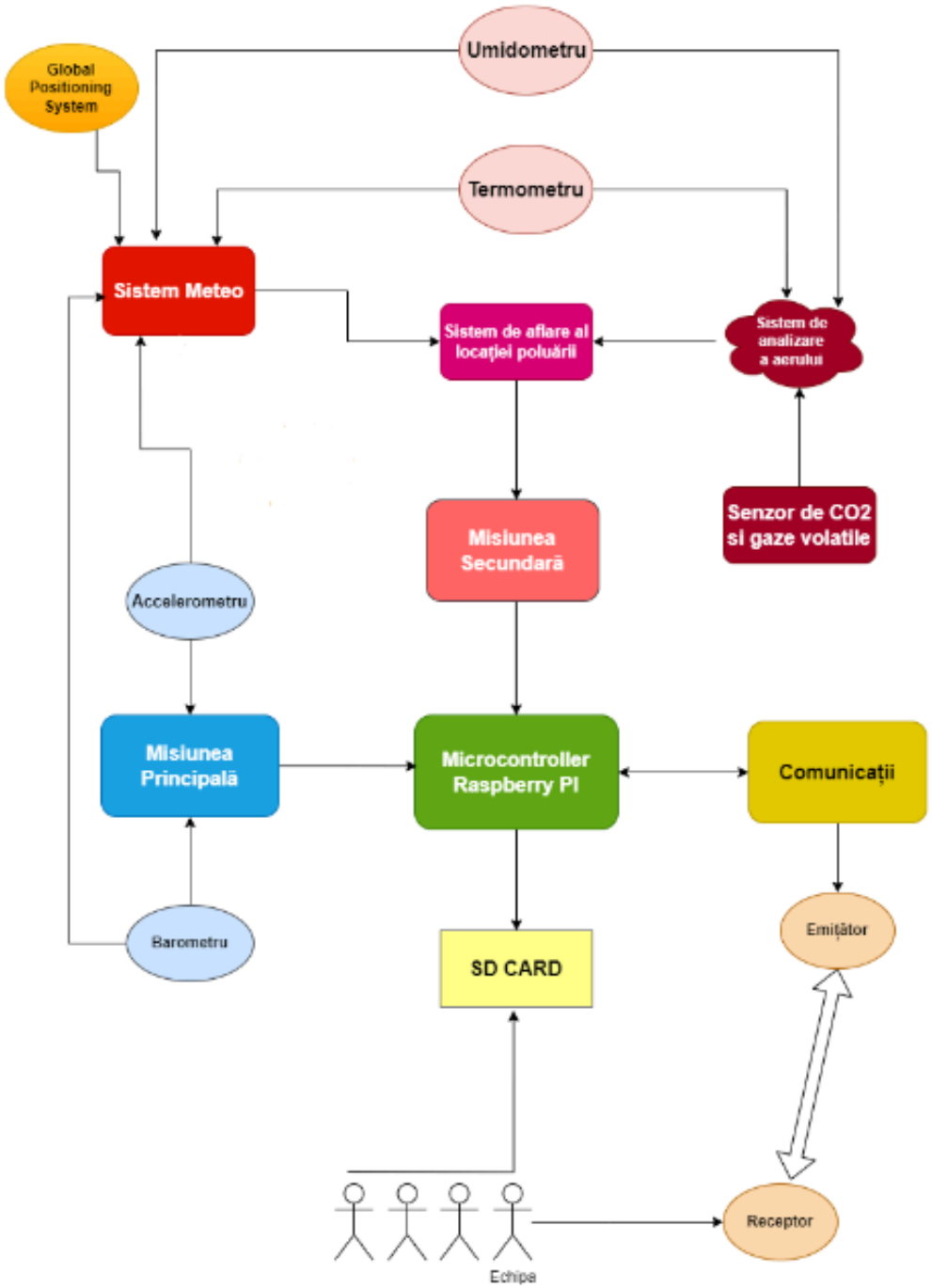
Depozitare: cubul și componentele lui nu trebuie fie lăsate într-un mediu umed.

Cubul trebuie sa fie protejat de praf sau apa.

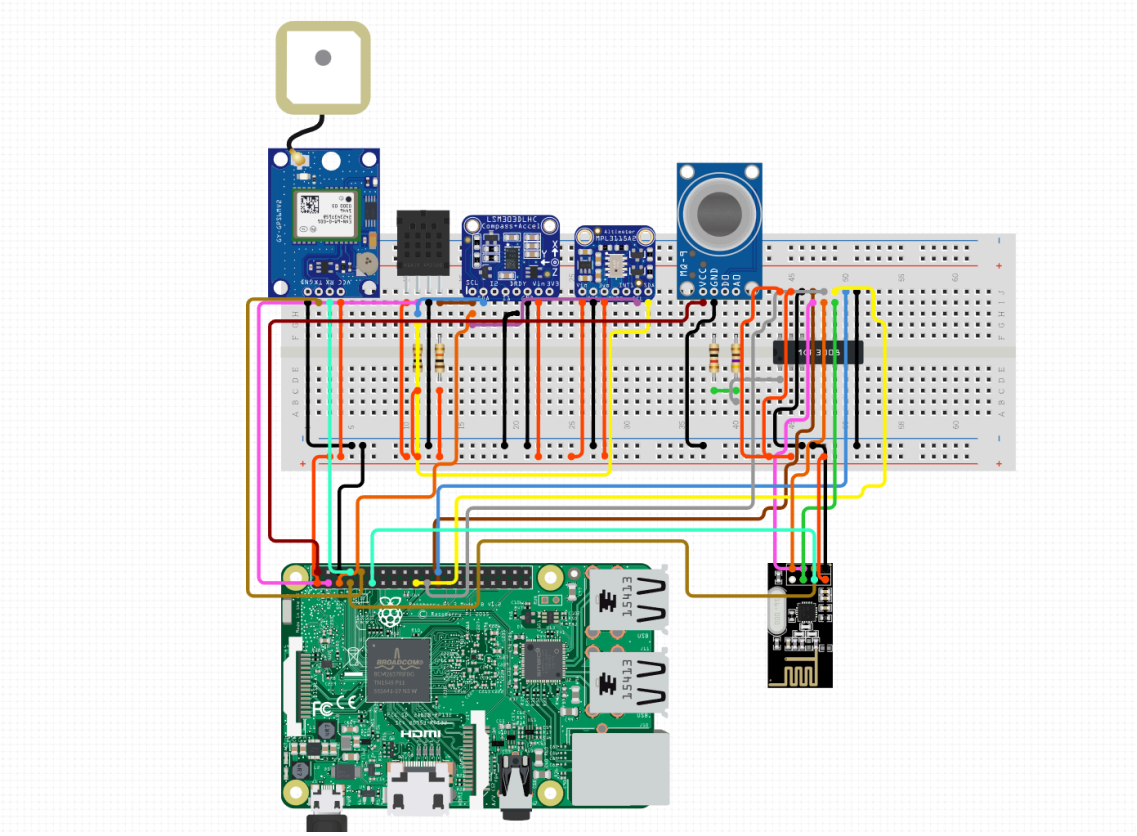
Nu trebuie puse obiecte grele pe el.

Nu trebuie scăpat de la o înălțime mare(ex:1 m).

Diagramă bloc


Global Positioning System
Umidometru
Termometru
Sistem Meteo
Sistem de aflare al locației poluării
Sistem de analizare a aerului
Misiunea Secundară
Senzor de CO2 si gaze volatile
Accelerometru
Misiunea Principală
Microcontroller Raspberry PI
Comunicații
Barometru
SD CARD
Emițător
Receptor
Echipa

Schema Electronica



Cod

```
import time

from smbus2 import SMBus,i2c_msg

import Adafruit_DHT

from serial import Serial

from spidev import SpiDev




#Senzor de umidiatate si temperatura
```

```
sensorTempUmd=Adafruit_DHT.DHT11

pinTU=27



#Barometru

bar=SMBus(1)

BAR_ADRESS=0x76



#GPS

ser = Serial(0)

numberBytesGPS=35



#acelerometru

accBus=SMBus(1)

ACC_ADDRES=0x53



#TransReciever

spiTR=SpiDev()

deviceTR,busTR=0,0

spiTR.open(busTR,deviceTR)



#Detector Gaze

spiDG=SpiDev()

deviceDG,busDG=1,0

numberBytesDG=10
```

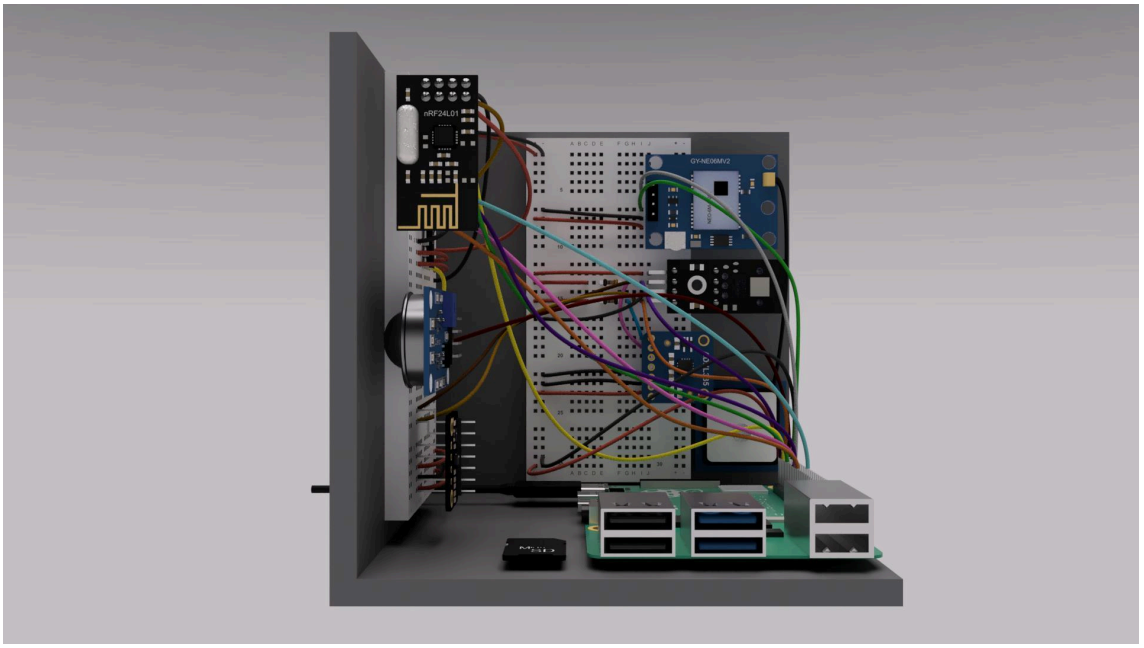
```
spiDG.open(busDG,deviceDG)



def DataToByteArray(arr):
    byte=[]
    for data in arr:
        byte=byte+bytes(str(data)+'\n',"ascii")
    return byte


def ReadUmdTemp():
    hum,temp=sensorTempUmd.read_retry(sensorTempUmd,pinTU)
    return hum,temp


def ReadPressTemp():
    press=0
    listBytes=bar.read_i2c_block_data(BAR_ADRESS,0x04,3)
    listBytes.reverse()
    for byte in listBytes:
        press=press<<8
        press+=byte
    temp=0
    listBytes=bar.read_i2c_block_data(BAR_ADRESS,0x07,3)
    listBytes.reverse()
    for byte in listBytes:
        temp=temp<<8
```

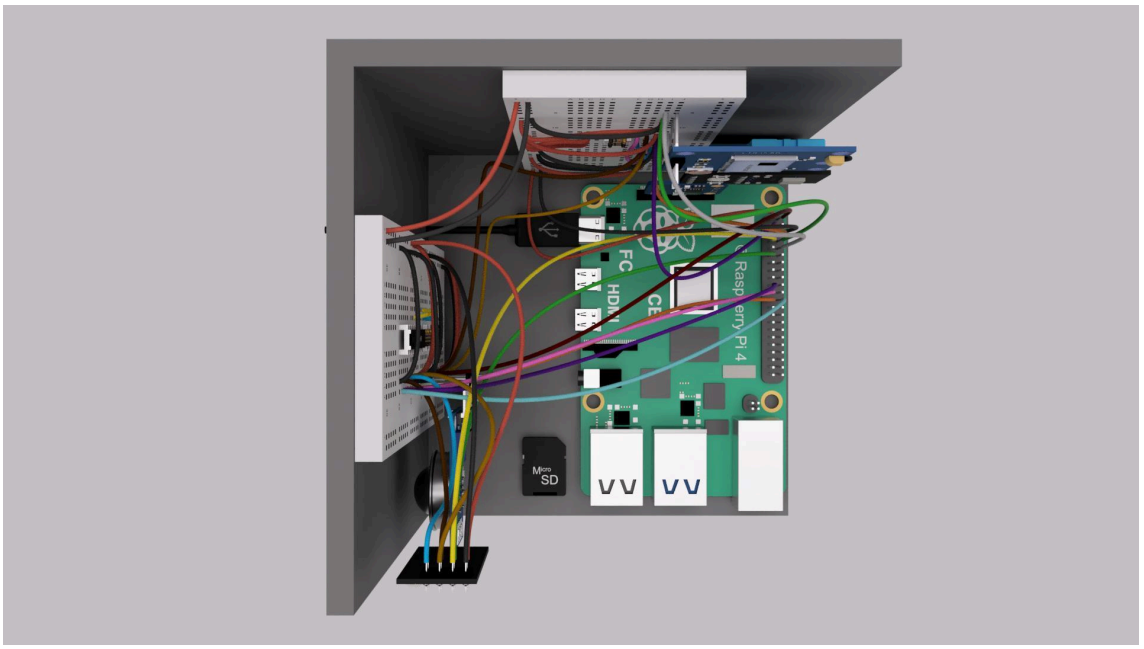
```
        temp+=byte
    return press,temp


def ReadGPSData():
    Bytes=ser.read_all()
    return Bytes.decode("ascii")



def ReadAcc():
    accX=0
    accY=0
    accZ=0
    accOffset=bar.read_i2c_block_data(ACC_ADDRES,0x1E,3)


    listBytes=bar.read_i2c_block_data(ACC_ADDRES,0x32,2)
    listBytes.reverse()
    for byte in listBytes:
        accX=accX<<8
        accX+=byte


    listBytes=bar.read_i2c_block_data(ACC_ADDRES,0x34,2)
    listBytes.reverse()
    for byte in listBytes:
        accY=accY<<8
```

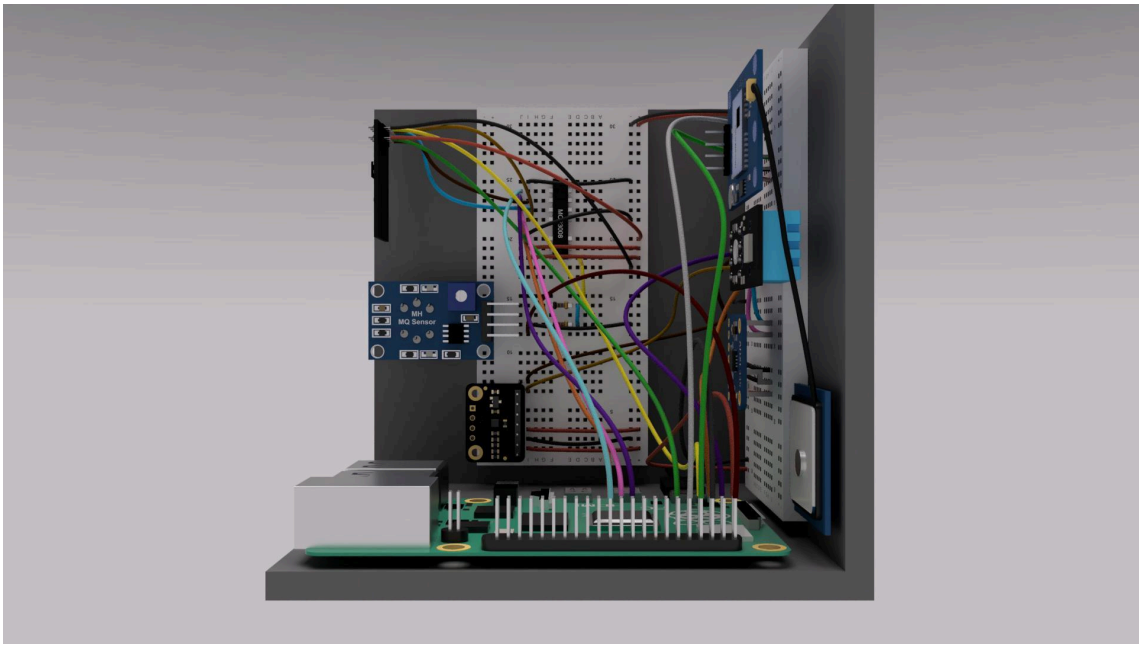
```
        accY+=byte


    listBytes=bar.read_i2c_block_data(ACC_ADDRES,0x36,2)
    listBytes.reverse()
    for byte in listBytes:
        accZ=accZ<<8
        accZ+=byte


    return accX,accY,accZ


#data under byte format
def SendTransReciverInfo(data):
    spiTR.xfer(data)


def ReadGasData():
    gas=0
    listBytes=spiDG.readbytes(numberBytesDG)
    listBytes[1]>>6
    for byte in listBytes:
        gas=gas<<8
        gas+=byte
    return gas
def WriteData(data):
    f=open("data.txt","a")
```

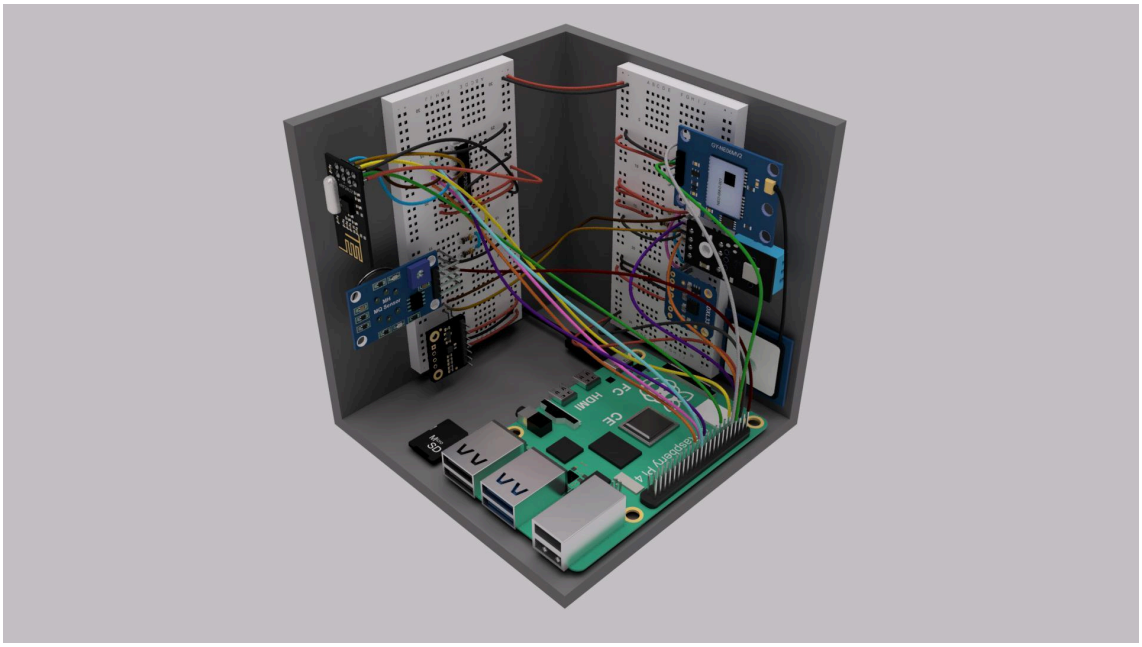
```
    f.write(data)
    f.close()


while True:
    umd,temp0=ReadUmdTemp()
    press,temp1=ReadPressTemp()
    gps=ReadGPSData()
    accx,accy,accz=ReadAcc()
    gas=ReadGasData()


    TotalData=[umd,temp0+temp1/2,press,gps,accx,accy,accz,gas]
    SendTransReciverInfo(DataToByteArray(TotalData))
    WriteData(TotalData)
    time.Sleep(0.5)
```

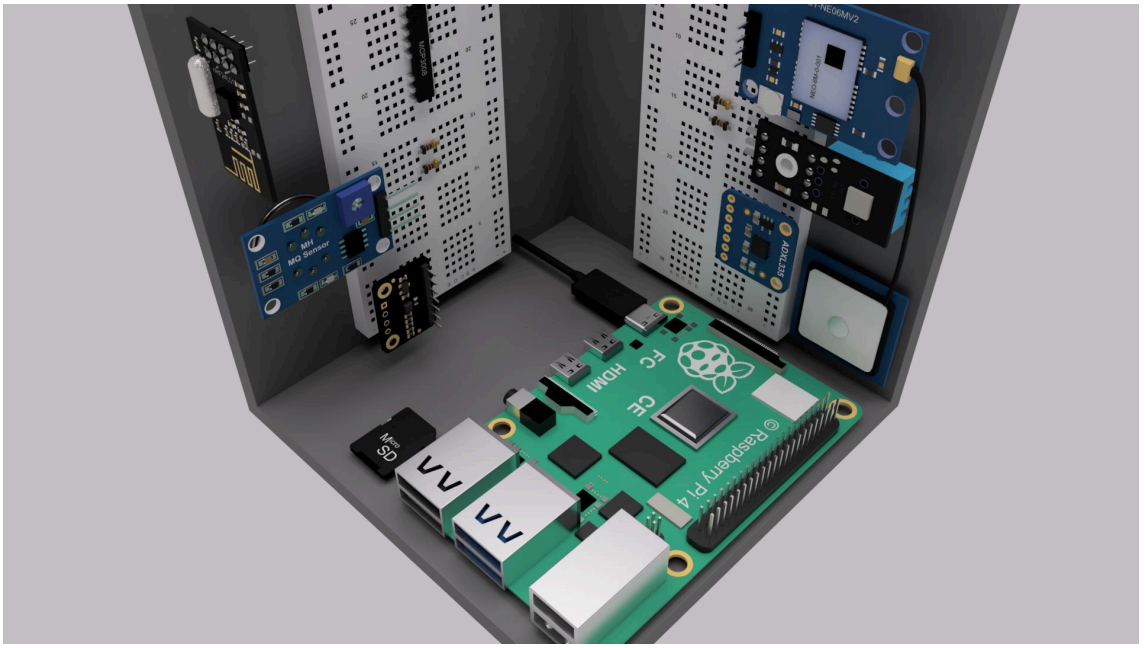
Render


nRF24L01
GY-NEO6MV2


Raspberry Pi 4
Micro SD


MH
MQ Sensor





Concluzii

În concluzie, având în vedere aspectele punctate, misiunea secundară va putea sa detecteze cu aproximare zone care generează poluare utilizând datele obtinute de senzorii din cub și prelucrându-le printr-un algoritm care este rulat pe Raspberry PI.