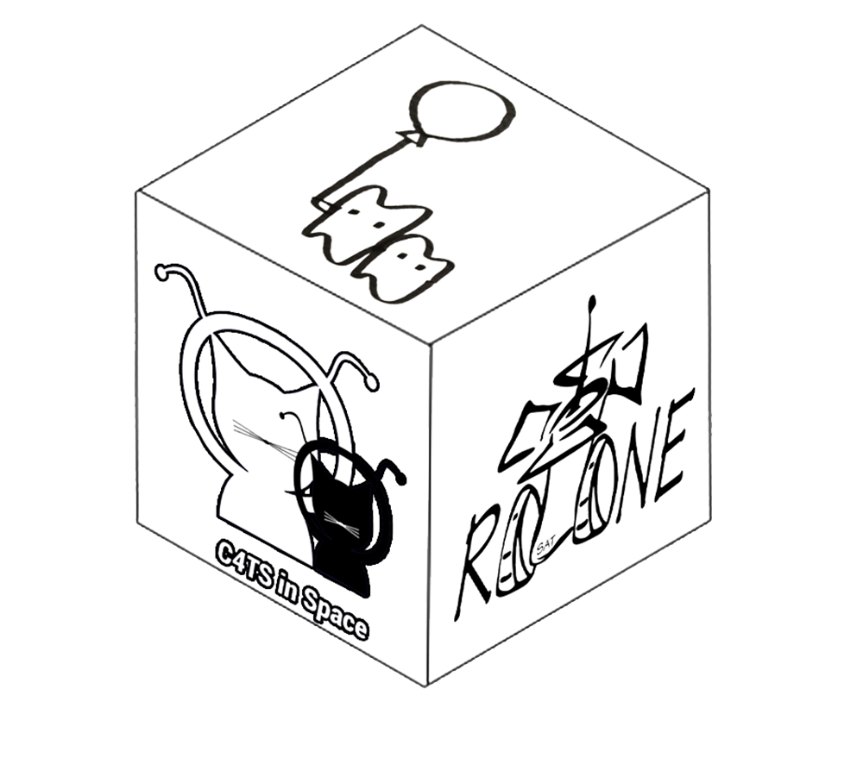
Încercare de documentație pentru radiosondă

Primul Balon meteorologic lansat de C4TS in Space pe 16.09.2018

Proiect al asociației RoSatOne

Radu Crăciun – șăf pă coade

Schema ce ar trebui sa pun pe aici:

Introducere ?

Detaliere hardware

Explicatii asupra protocoalelor de comunicare intre placi, senzori, etc. + anexe pentru protocoale poate, nuj

Comunicare GSM si APRS

Detaliere APRS si explicare driver de pe ardunio

Detaliere comenzi GSM – procedura trimitere SMS

Detaliere protocoale proprii – modul de stocare a datelor

Scheme logice cod

Scheme ale bucatilor de soft in sine (ce .py il cheama pe altul si asa mai departe)

Cuprins

[Hardware 3](#_Toc530078751)

[Listă componente 3](#_Toc530078752)

[Schemă generală a dispozitivului 4](#_Toc530078753)

[Alimentare 4](#_Toc530078754)

[Pinout 4](#_Toc530078755)

[Scheme detaliate plăci 4](#_Toc530078756)

[Schemă MPU9250 + Arduino Pro Mini + Modul SD 4](#_Toc530078757)

[PCB-uri 5](#_Toc530078758)

[Comunicarea internă 6](#_Toc530078759)

[Protocoale de comunicare 8](#_Toc530078760)

[I²C: 8](#_Toc530078761)

[Serial: 8](#_Toc530078762)

[1-Wire: 8](#_Toc530078763)

[SPI: 8](#_Toc530078764)

[analogRead(): 8](#_Toc530078765)

[PWM: 8](#_Toc530078766)

[Cod – Programarea radiosondei 9](#_Toc530078767)

[Codul de pe Raspberry Pi 10](#_Toc530078768)

[Moduri de împachetare a datelor 12](#_Toc530078769)

# 

# Hardware

## Listă componente

Calculatoare:

* **Raspberry Pi Zero W – computer principal** (Wi-Fi pentru a ușura programarea)
* Arduino Pro Mini (modul SD + accelerometru)
* Arduino Pro Mini (Driver APRS și monitor tensiune baterie)

Senzori:

* BMP180 – presiune (și temperatură)
* SHT21 – umiditate si temperatură
* TSL2561 – luminozitate vizibilă și infraroșie
* DS18B20 – temperatură interioară
* MPU9250 – accelerometru + giroscop + magnetometru

Auxiliare:

* Ublox Neo 6m – modul GPS
* Radiometrix HX1 144,8 MHz
* Neoway M590E – modul GSM
* Modul SD
* Bate Li-Po principală
* Baterie Li-Po modul GSM și MPU9250 + Arduino
* Întrerupător
* Antenă APRS

## Schemă generală a dispozitivului

## Alimentare

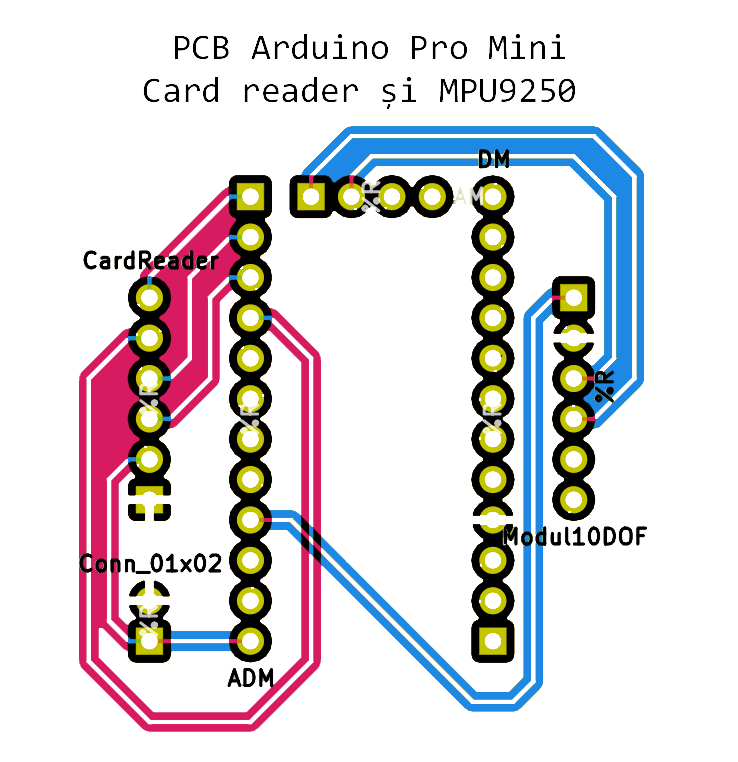
## Pinout

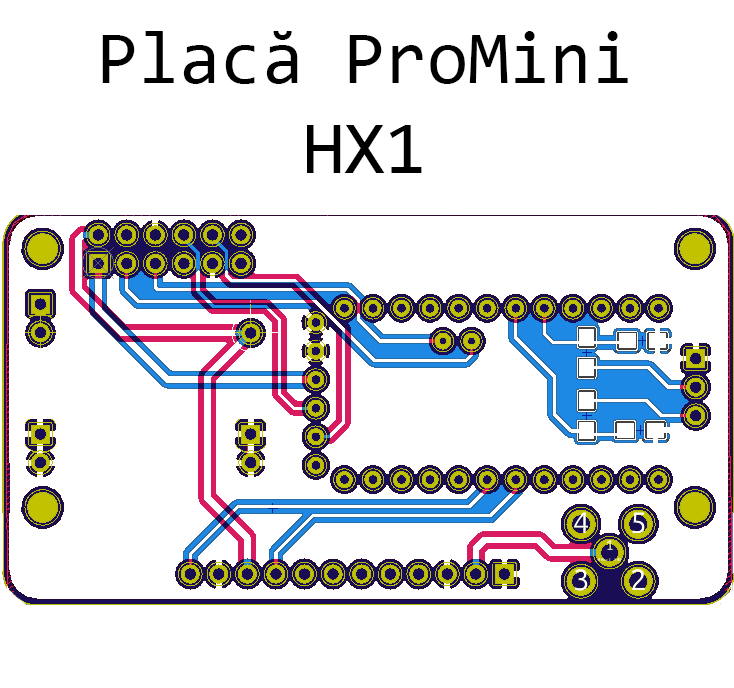
## Scheme detaliate plăci

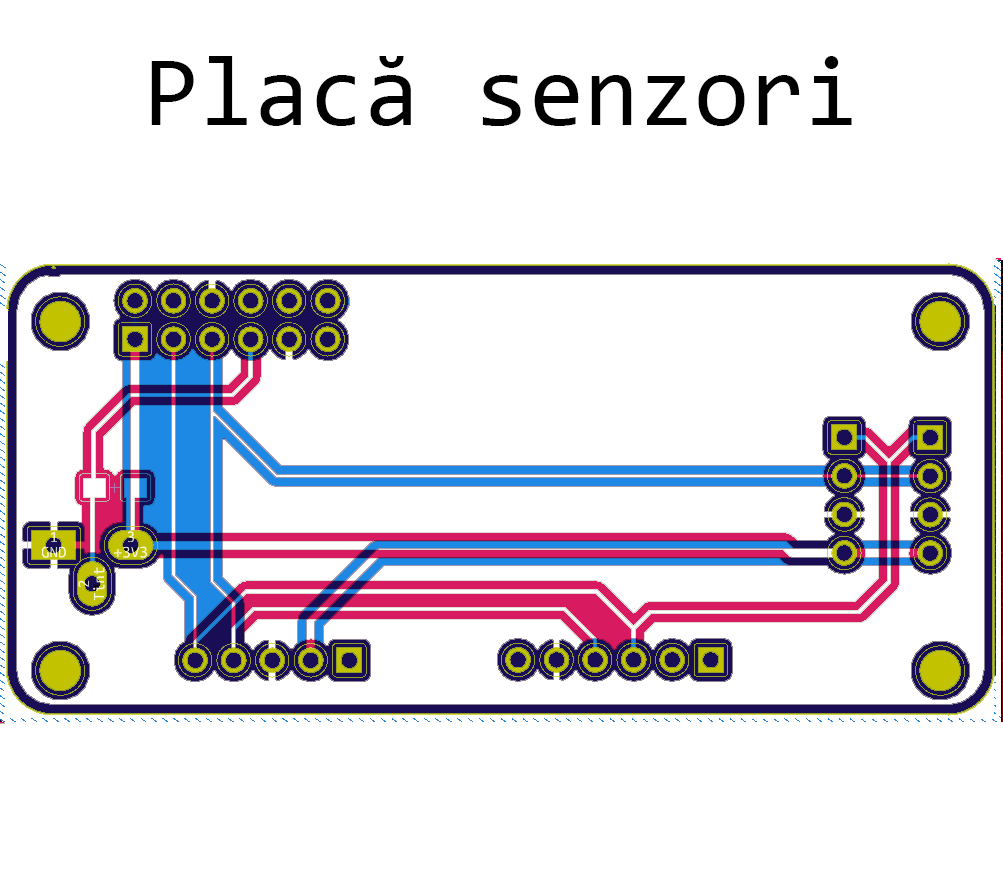
## Schemă MPU9250 + Arduino Pro Mini + Modul SD

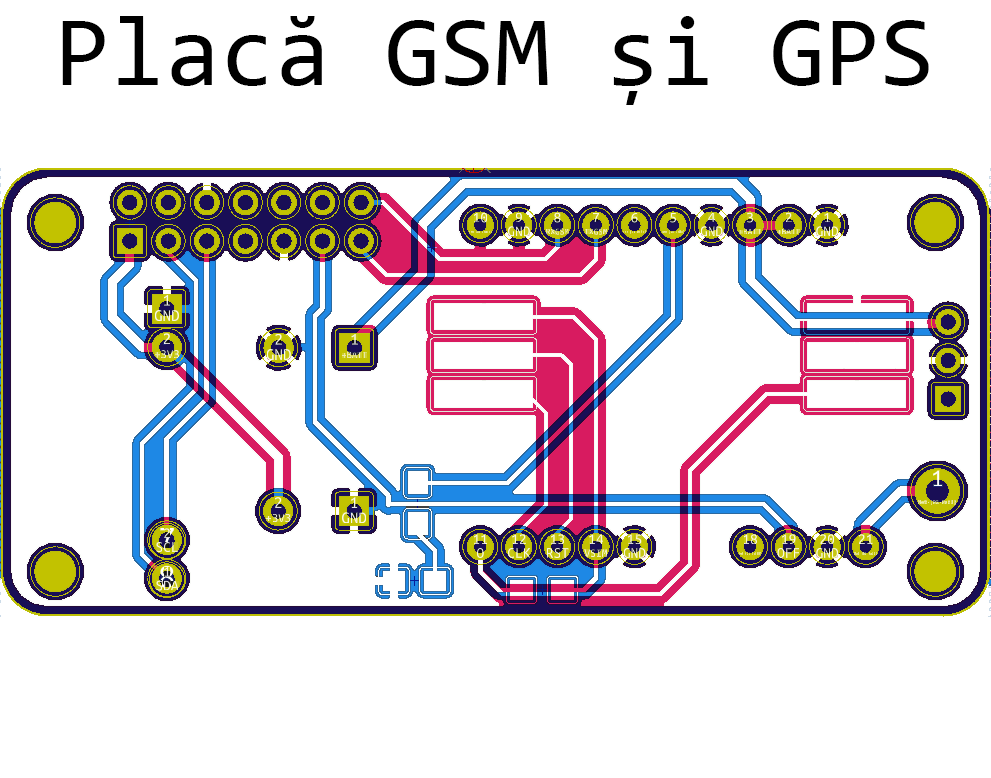
## PCB-uri

Fața plăcii Spatele plăcii

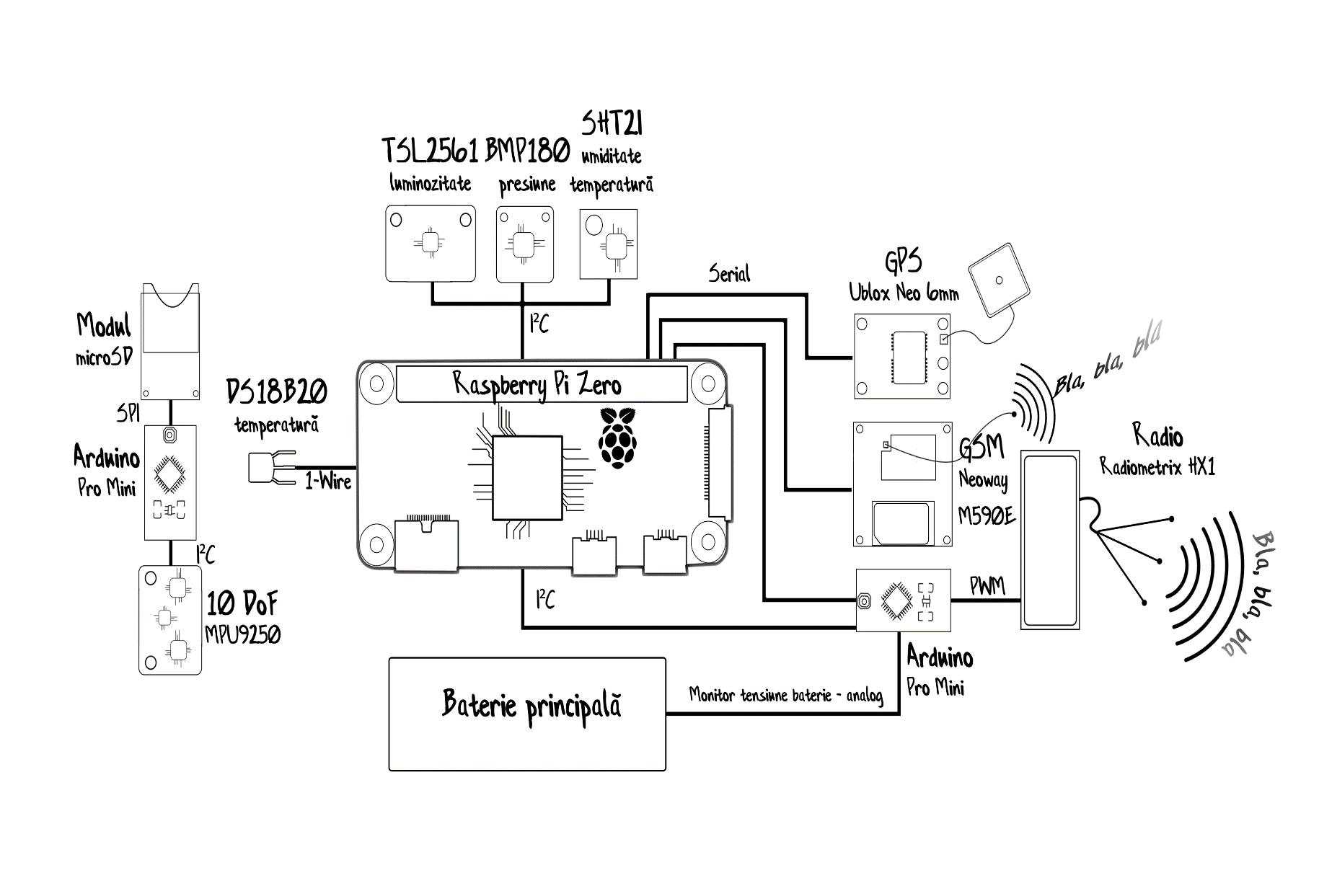








# Comunicarea internă



*Figura TREBUIE NUMEOTATE – Protocoalele de comunicare între componente*

## Protocoale de comunicare

## I²C:

* Între Raspberry Pi și:
  + TSL2561 (0x39)
  + BMP180 (0x77)
  + SHT21 (0x40)
  + Arduino Pro Mini – Monitorizare baterie (0x04)
* Între Arduino Pro Mini independent și:
  + MPU9250 (0x68)

## Serial:

* Între Raspberry Pi și:
  + Arduino Pro Mini – Pachet APRS ca text (baud 9600) – „/dev/ttyS0”
  + Ublox Neo 6m – GPS (b9600) -
  + Neoway M590E – GSM (b9600) -

## 1-Wire:

* Între Raspberry Pi și:
  + DS18B20 – încărcat ca LKM (Linux Kernel Module) și citit ca fișier

## SPI:

* Între Arduino Pro Mini independent și:
  + Modul microSD (0x0A)

## analogRead():

* Între Arduino Pro Mini – Monitor Baterie și:
  + Bateria principală (referință analog = INTERNAL de 1,1V)
    - Tensiunea reala este coborâtă în intervalul (0 – 1,1V) electronic și apoi recalculată cu niște valori de calibrare – detalii la „Scheme” și „Cod”

## PWM:

* Între Arduino Pro Mini – Driver Radio și:
  + Radiometrix HX1 – Radio APRS
    - Arduino modulează (AFSK)[[1]](#footnote-2) pachetul text primit de la RPi0 în semnal PWM ce este trimis la HX1 printr-un pin

# Cod – Programarea radiosondei

Codul dispozitivului propriu-zis este realizat în două limbaje de programare:

* Python pentru codul de pe Raspberry Pi
* C++ pe cele două Arduino.

Pentru a schimba codul radiosondei, este recomandat să te conectezi la Raspberry Pi prin SSH și FTP.[[2]](#footnote-3)

TODO: Baga aici cum anume o programezi

## Codul de pe Raspberry Pi

Urmează o detaliere a codului (cum era la data de 22.09.2018, ultimul *commit* de pe *GitHub – GigaCloud/C4TS[[3]](#footnote-4)*), parcurgând fiecare fișier al codului și explicându-i rolul.

Fișiere de cod:

* Biblioteci:
  + *aprs.py* – conține o singură funcție: *send(TEXT)* ce trimite pe *Serial* /dev/ttyS0conținutul text al pachetului APRS ce ulterior va fi procesat de catre Arduino
  + *batt.py –* asigură comunicarea cu Arduino prin *I²C* și calculează tensiunea totală din baterie și pe cea dintr-o celulă, folosindu-se de valorile de corecție *C0* și *C1*
  + *bmp180.py –* biblioteca standard scrisă de *SparkFun* pentru senzorul BMP180 + o funcție adaugată numită pres2alt(pressure)ce returnează altitudinea (în metri)
  + *ds.py –* conține definiția clasei *DS1820* (sincer, erau suficinete funcțiile, dar așa era scrisă de sursa bibliotecii și nu a mai fost modificat); citesțe temperatura de pe senzorul cu numele clasei și o returnează.

De asemenea, în inițializarea clasei, se încarca modulele LKM w1-gpio și w1-therm

* + *gps.py –* se folosește de *rx\_gps.py* pentru a citi datele de GPS neprelucrate (*propoziții NMEA*[[4]](#footnote-5)), mai apoi prelucrându-le în format zecimal sau APRS[[5]](#footnote-6).

De asemenea, conține funcțiile getLastLocation()și getLastLocationDecimal() ce returnează ultimele coordonate valide ce sunt salvate în fișierele /home/pi/gps.txt /home/pi/gps\_dec.txt

* + *gsm.py* – conține diverse funcții specifice modulului GSM: unele ce trimit diverse comenzi în format AT[[6]](#footnote-7), procedee de pornire/oprire ale modulului, variând tensiunea de pe pinul *Boot* într-o anumită secvență, funcție de citire, etc.

\*Citirea de pe GSM nu se face pe serial numai la apel, ci se face constant, fiind salvată într-un fișier (home/pi/Final/gsm.txt).Funcția de citire descrisă în această biblioteca este pentru ultimele *n* linii din fișier - Rx(nr\_linii = 1)

* + *rx\_gps.py -* are o funcție ce citește în format brut datele de pe *Serial*-ul modulului GPS
  + *sht21.py –* bibliotecă cu funcții standard a senzorului SHT21 – citire I2C și prelucrare
  + *tsl.py –* bibliotecă foarte compactă pentru senzorul TSL2561 (10 linii de cod) – citire I2C
* Fișiere executabile:
  + *rx\_gsm.py* – rulează tot timpul, citind date de pe serial și punându-le in fișierul home/pi/Final/gsm.txt
  + *collect.py* ***–*** citește date de pe toți senzorii, le împachetează în două moduri și le salvează în fișierele /home/pi/Date.txt /home/pi/lastData.txt
  + *sendAPRS.py –* citește date din fișierul /home/pi/lastData.txt, le transpune în format APRS, formând doua pachete distincte. Trimte primul pachet, așteaptă 3 secunde, apoi îl trimite pe urmatorul. Primul pachet are iconiță de *Weather Station*, iar al doilea de balon meteorologic. Schimbul de iconiție este făcut astfel încât site-ul *aprs.fi* să afișeze într-un mod mai ușor de citit datele în timpul zborului.
  + *sendCoord.py –* verifică altitudinea sondei, iar în funcție de aceasta trimite sau nu un SMS cu ultimele coordonate valide. SMS-urile se trimit odată la 10 minute. În caz că sonda tocmai a reintrat sub marja de altitudine, primul SMS se va trimite în maxim 30 de secunde
  + *stopGSM.py –* doar un simplu fișier executabil ce apelează funcția stop() din *gsm.py*
  + *main.cpp (compilat ca „main”) –* rutina sondei. Execută collect.py în mod constant, *sendAPRS.py* la fiecare minut, iar sendCoord.py la fiecare 30 de secunde
  + *start.sh –* se execută la boot. Inițializează biblioteca pigpiod, șterge fișierul /home/pi/Final/gsm.txt, începe execuția în paralel a *rx\_gsm.py*, oprește modulul GSM
  + și pornește rutina *main.*

Figura TREBUIE NUMEROTATE – Rutina radiosondei

## 

O schemă cu legăturile dintre fișierele codului este reprezentată în figura TREBUIE NUMEROTATE.

## 

Figura TREBUIE NUMEROTATE – *Rutina radiosondei*

## Moduri de împachetare a datelor

Fișierul „Date.txt” conține pachete în formatul

*DATĂ ORĂ  
SENZOR: STARE (DATE)*

Spre exemplu:

2018-09-15 13:30:58.385020

BMP:True (26.4, 974.48)

GPS:False ['4656.15N', '02622.95E', '337.5']

DS18:True 26.13

TSL:True (282, 134)

SHT:True(24.94, 52.43)

BAT:False (-1, -1)

\*Unii senzori funcționează OK (fapt redat de stările „True”), mai puțin GPS-ul și Monitorul de Baterie.

GPS-ul are o eroare (fie de citire, fie lipsă de *position fix* – nu este implementată vreo diferențiere între cele două). Între paranteze se află ultimele coordonate valide și altitudinea, salvate in fișierul /home/pi/gps.txt

Monitorul de baterie are o eroare de citire, iar datele sunt înlocuite cu *-1*.[[7]](#footnote-8)

Fișierul „lastData.txt” conține pachete în formatul

TEMP\_BMP,PRESIUNE,LATITUDINE,LONGITUDINE,ALTITUDINE,TEMP\_DS18,LUM,LUM\_IR,TEMP\_SHT,

UMDITATE,TENSIUNE\_CELULĂ,TENSIUNE\_TOTALĂ

CodSenzorStareNumerică;TS Luminozități; DS Temperatură; BAT Tensiuni

Exemplu:

26.7,974.44,4656.15N,02622.95E,337.5,26.38,285,135,24.91,53.76,3.-1,-1

B1G0D1T1S1Ba0;TS285;135;DS26.38;BAT-1;-1

Starea fiecarui senzor este înregistrată într-un șir de caractere în formatul Senzor(litera)Stare(1 sau 0). „B1G0D1T1S1Ba0” reprezintă o funcționare corectă a tuturor senzorilor, mai puțin a GPS-ului și monitorului de baterie. Datele bateriei sunt înlocuite cu *-1* în acest caz.

În mod evident, primul format, cel prezent în „Date.txt” este mult mai ușor de citit. Fișierul „lastData.txt” este folosit de *sendAPRS.py* pentru a lua din el ultimele date citite, a le împacheta și trimite către radio, iar de accea formatul său este compact și greu de citit pentru un om.

1. Audio Frequency Shift Keying [↑](#footnote-ref-2)
2. La data scrierii acestui document (15.11.2018) – user: pi; pass: ayylmao123 [↑](#footnote-ref-3)
3. Codul prezentat se află sub dosarul „*Final”* de pe *GitHub* [↑](#footnote-ref-4)
4. Standard dezvoltat de *National Marine Electronics Association,* folositde mai toate modulele GPS [↑](#footnote-ref-5)
5. APRS101.pdf – „APRS PROTOCOL REFERENCE Protocol Version 1.0” – paginile 23-24 [↑](#footnote-ref-6)
6. Vezi documentația modulului Neoway descris în secțiunea *Hardware* a documentului [↑](#footnote-ref-7)
7. Toate datele ce nu au putut fi citite/sunt eronate sunt înlocuite cu *–1* și sunt trimise așa către APRS [↑](#footnote-ref-8)