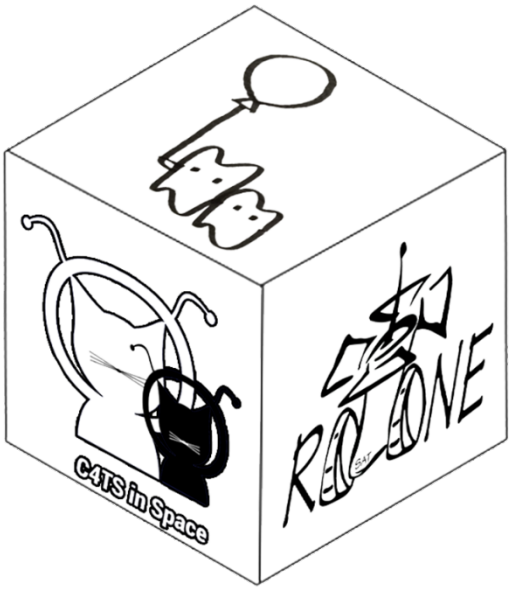
Documentație lacunară dar mai bună decât nimic pentru radiosondă

Primul Balon meteorologic lansat de C4TS in Space pe 16.09.2018

Proiect al asociației RoSatOne



Radu Crăciun – maimuță dactilograf

https://github.com/GigaCloud/C4TS

Cuprins

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:

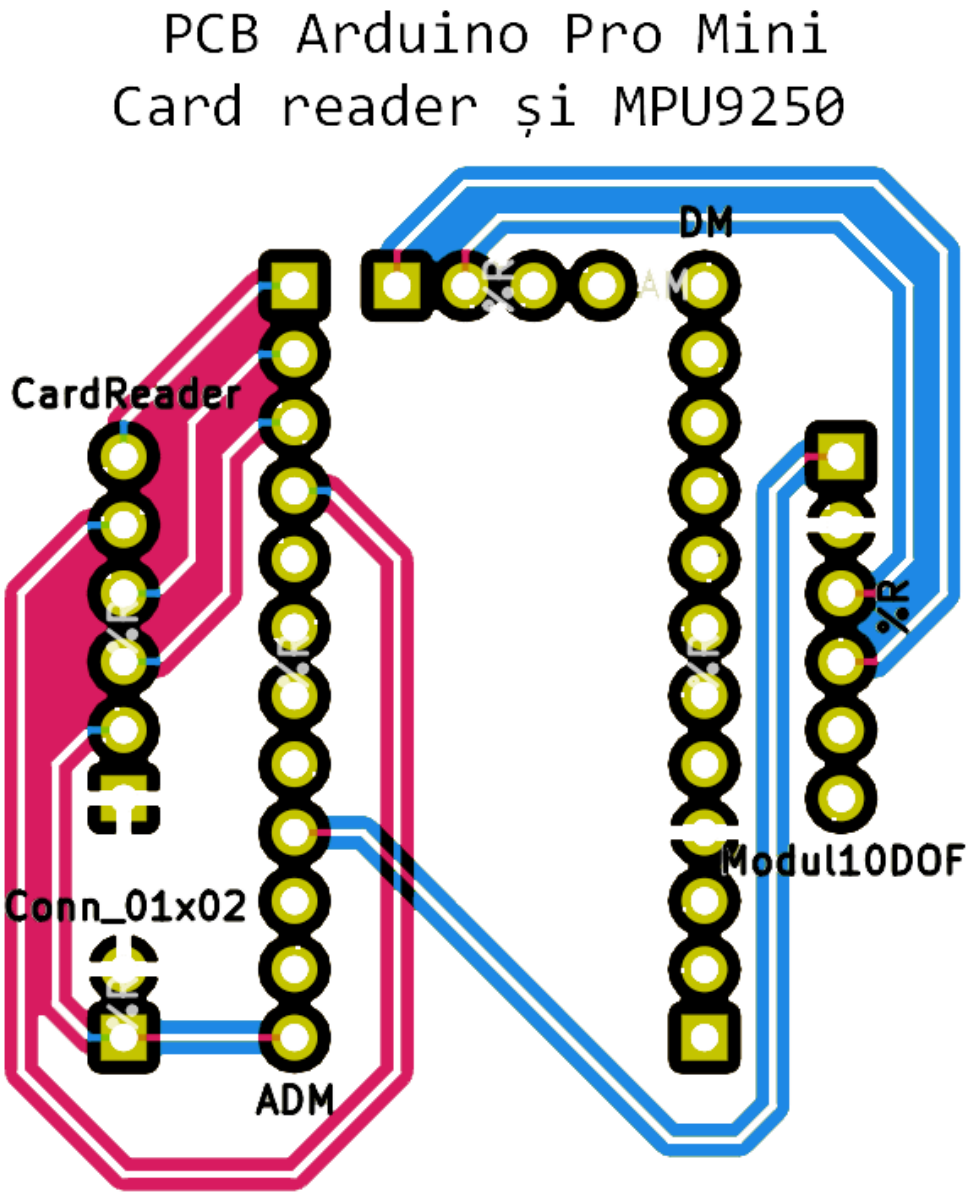
- BMP180 – presiune (și temperatură)
- SHT21 – umiditate si temperatură
- TSL2561 – luminozitate vizibilă și infraroșie
- DS18B20 – temperatură interioară
- MPU9250 – accelerometru + giroscop + magnetometru

Auxiliare:

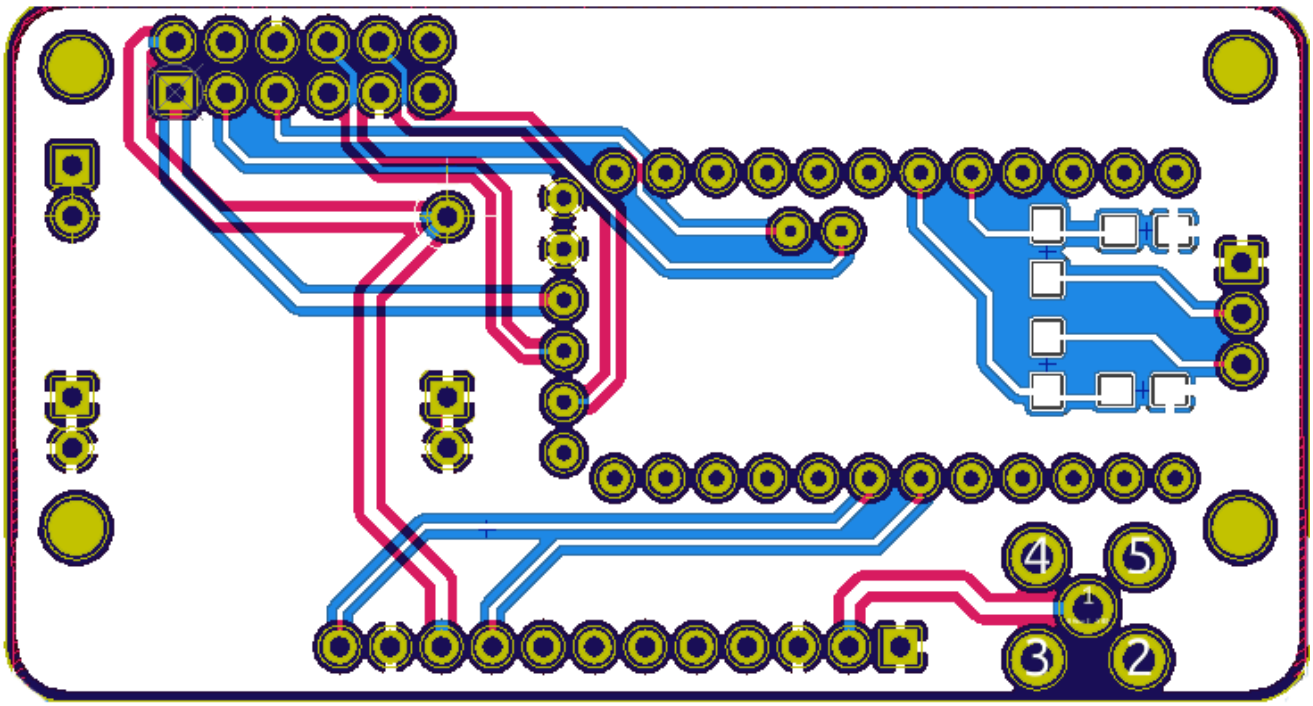
- Ublox Neo 6m – modul GPS
- Radiometrix HX1 144,8 MHz
- Neoway M590E – modul GSM
- Modul SD
- Bate Li-Po principală
- Baterie Li-Po modul GSM și MPU9250 + Arduino
- Întrerupător
- Antenă APRS

PCB-uri

Fața plăcii Spatele plăcii

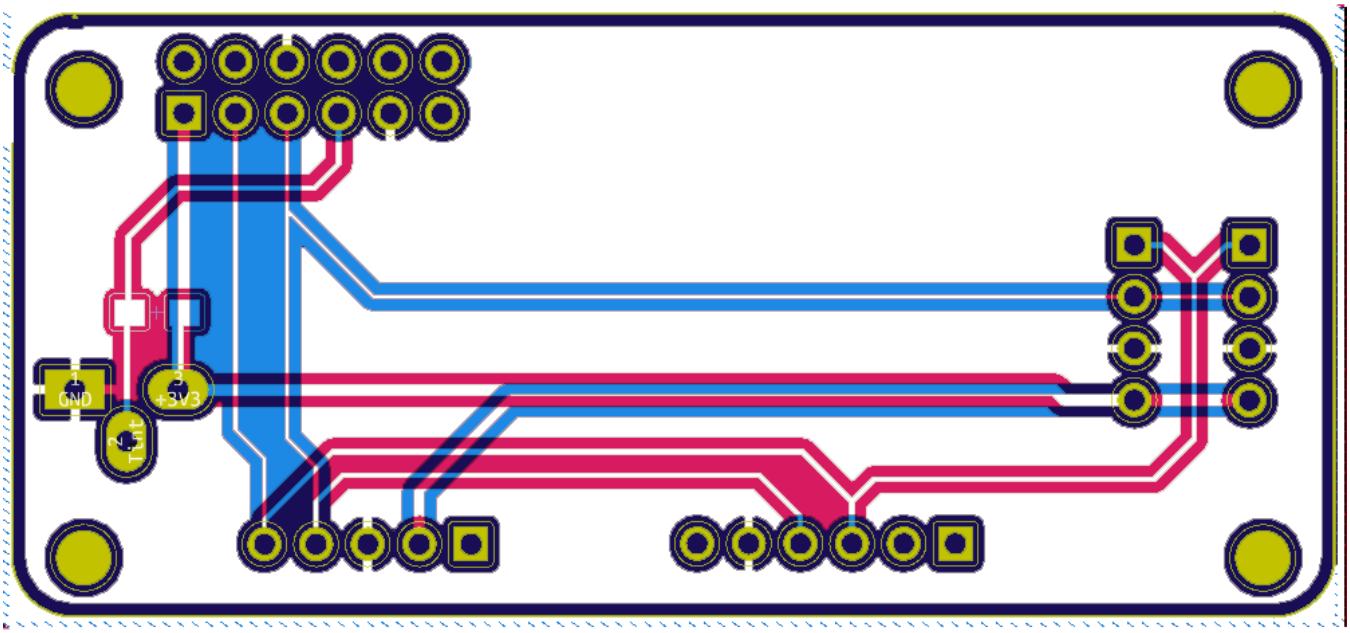


Placă ProMini HX1

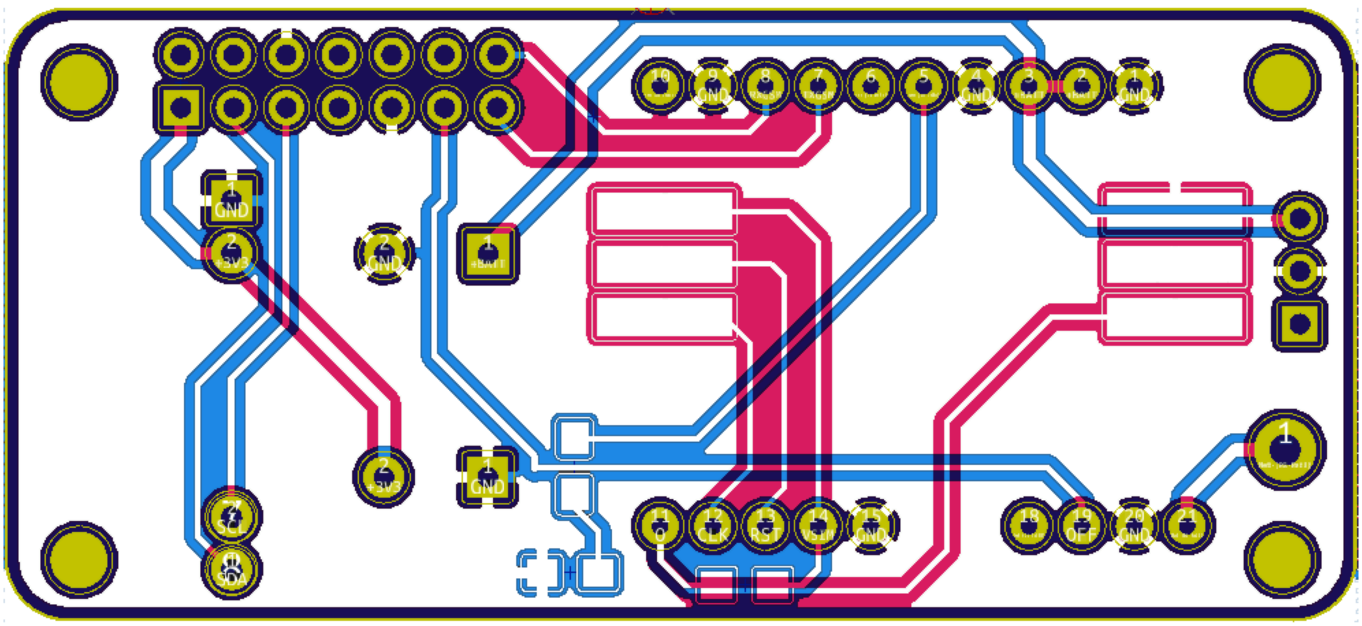


https://github.com/GigaCloud/C4TS

Placă senzori



Placă GSM și GPS



Comunicarea internă
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) - $Pin_{RX}=22$
  - Neoway M590E – GSM (b9600) - $Pin_{RX}=22$

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():

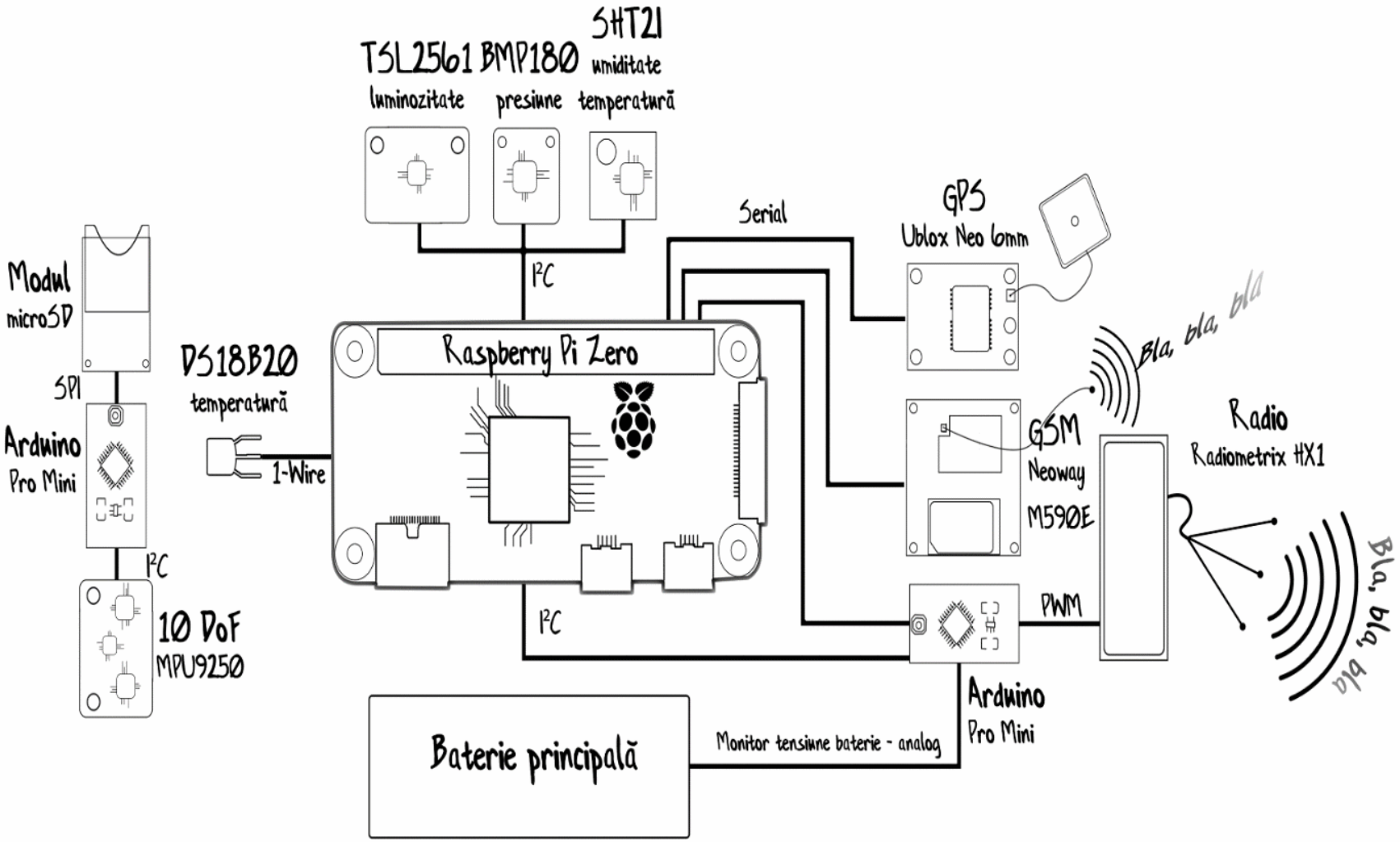
- Între Arduino Pro Mini – Monitor Baterie și:
  - Bateria principală (referință analog = INTERNAL de 1,1V)
    - Tensiunea reala este coborâtă în intervalul (0 – 1,1V) electronic și apoi recalculată cu niște valori de calibrare – detalii la „Scheme” și „Cod”

PWM:

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

[1] Audio Frequency Shift Keying

https://github.com/GigaCloud/C4TS



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 principal al radiosondei, este recomandat să te conectezi la Raspberry Pi prin SSH și FTP.[2]

Codul[3] principal al radiosondei, cel de pe RaspberryPi este în dosarul `/home/pi/Final`

Programarea driverului de radio (Arduino ProMini), cât și a modului SD se realizează prin îndepartarea plăcilor de Arduino pe soclu și utilizarea unui programator de tip FTDI.

[2] La data scrierii acestui document (15.11.2018) – user: pi; pass: ayylmao123. Hotspot - SSID: Omnom; parola: adusadus1

[3] La data rescrierii acestui document (16.03.2019)

Codul

Codul de pe RaspberryPi

Urmează o detaliere a codului (cum era la data de 22.09.2018, ultimul *commit* de pe *GitHub – GigaCloud/C4TS*[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/ttyS0` conținutul 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); citește 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*[5]), mai apoi prelucrându-le în format zecimal sau APRS[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[7], procedee de pornire/oprire ale modulului, variând tensiunea de pe pinul *Boot* într-o anumită secvență, funcție de citire, etc.

[4] Codul prezentat se află sub dosarul „*Final”* de pe *GitHub*
[5] Standard dezvoltat de *National Marine Electronics Association,* folosit de mai toate modulele GPS
[6] APRS101.pdf – „APRS PROTOCOL REFERENCE Protocol Version 1.0” – paginile 23-24
[7] Vezi documentația modulului Neoway descris în secțiunea *Hardware* a documentului

*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.

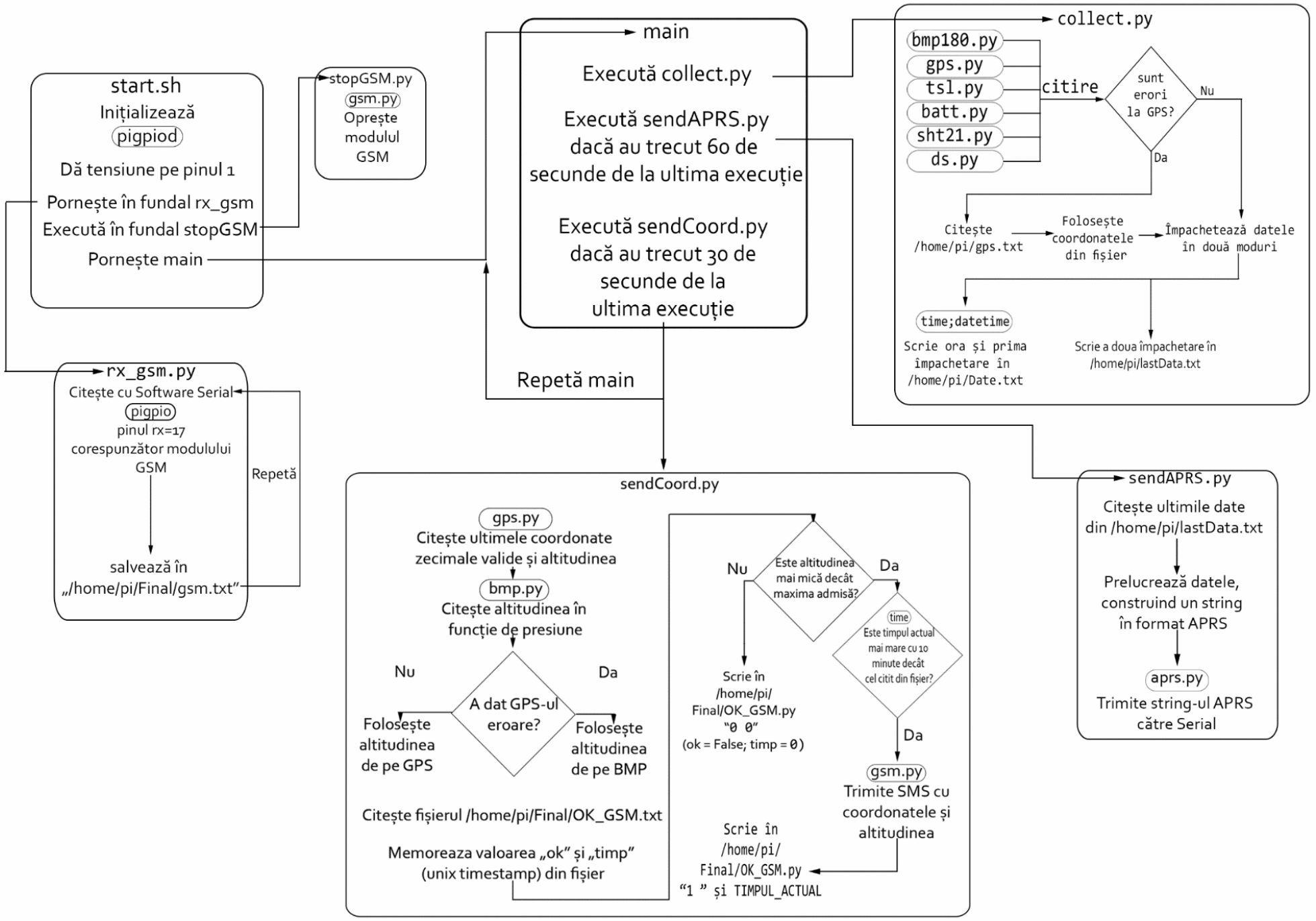
  - *sendCoord.py* – verifică altitudinea sondei, iar în funcție de aceasta trimite sau nu un SMS cu ultimele coordonate valide. SMS-urile se trimit odată la 10 minute. În caz că sonda tocmai a reintrat sub marja de altitudine, primul SMS se va trimite în maxim 30 de secunde

  - *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.*

O schemă cu legăturile dintre fișierele codului este reprezentată în figura următoare


start.sh
Inițializează
pigpiod
Dă tensiune pe pinul 1
Pornește în fundal rx_gsm
Execută în fundal stopGSM
Pornește main
stopGSM.py
gsm.py
Oprește modulul GSM
main
Execută collect.py
Execută sendAPRS.py dacă au trecut 60 de secunde de la ultima execuție
Execută sendCoord.py dacă au trecut 30 de secunde de la ultima execuție
Repetă main
collect.py
bmp180.py
gps.py
tsl.py
batt.py
sht21.py
ds.py
citire
sunt erori la GPS?
Nu
Da
Citește /home/pi/gps.txt
Folosește coordonatele din fișier
Împachetează datele în două moduri
time;datetime
Scrie ora și prima împachetare în /home/pi/Date.txt
Scrie a doua împachetare în /home/pi/lastData.txt
rx_gsm.py
Citește cu Software Serial
pigpio
pinul rx=17 corespunzător modulului GSM
salvează în „/home/pi/Final/gsm.txt”
Repetă
sendCoord.py
gps.py
Citește ultimele coordonate zecimale valide și altitudinea
bmp.py
Citește altitudinea în funcție de presiune
Nu
Da
A dat GPS-ul eroare?
Folosește altitudinea de pe GPS
Folosește altitudinea de pe BMP
Citește fișierul /home/pi/Final/OK_GSM.txt
Memoreaza valoarea „ok” și „timp” (unix timestamp) din fișier
Este altitudinea mai mică decât maxima admisă?
Nu
Da
Scrie în /home/pi/Final/OK_GSM.py “0 0” (ok = False; timp = 0)
time
Este timpul actual mai mare cu 10 minute decât cel citit din fișier?
Da
gsm.py
Trimite SMS cu coordonatele și altitudinea
Scrie în /home/pi/Final/OK_GSM.py “1 ” și TIMPUL_ACTUAL
sendAPRS.py
Citește ultimile date din /home/pi/lastData.txt
Prelucrează datele, construind un string în format APRS
aprs.py
Trimite string-ul APRS către Serial

Codul de pe Driverul Radio

Codul de pe driverul de radio este în mare parte bazat pe proiectul sayden/hispatracker.[8].

Acestuia i s-a mai adăugat partea de comunicare prin I2C și Serial cu RaspberryPi-ul și monitorizarea tensiunii din baterie.

Modificările se găsesc în driver.ino. Adresa de I2C aleasă este 0x04.

Tensiunea este măsurată în intervalul 0-1,1V (analogReference(INTERNAL)).

Codul se găsește în dosarul APRS_Driver+i2C_Battery/driver/.

Codul pentru modulul SD

Se găsește în dosarul mpu_card/ și nu este decât o uniune a codului de driver MPU și codului de driver SD.

Comentariile din acesta ar trebui să fie suficiente.

[8]https://github.com/sayden/hispatracker/tree/master/hispatracker

Date

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.

Dacă 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*.[9]

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:

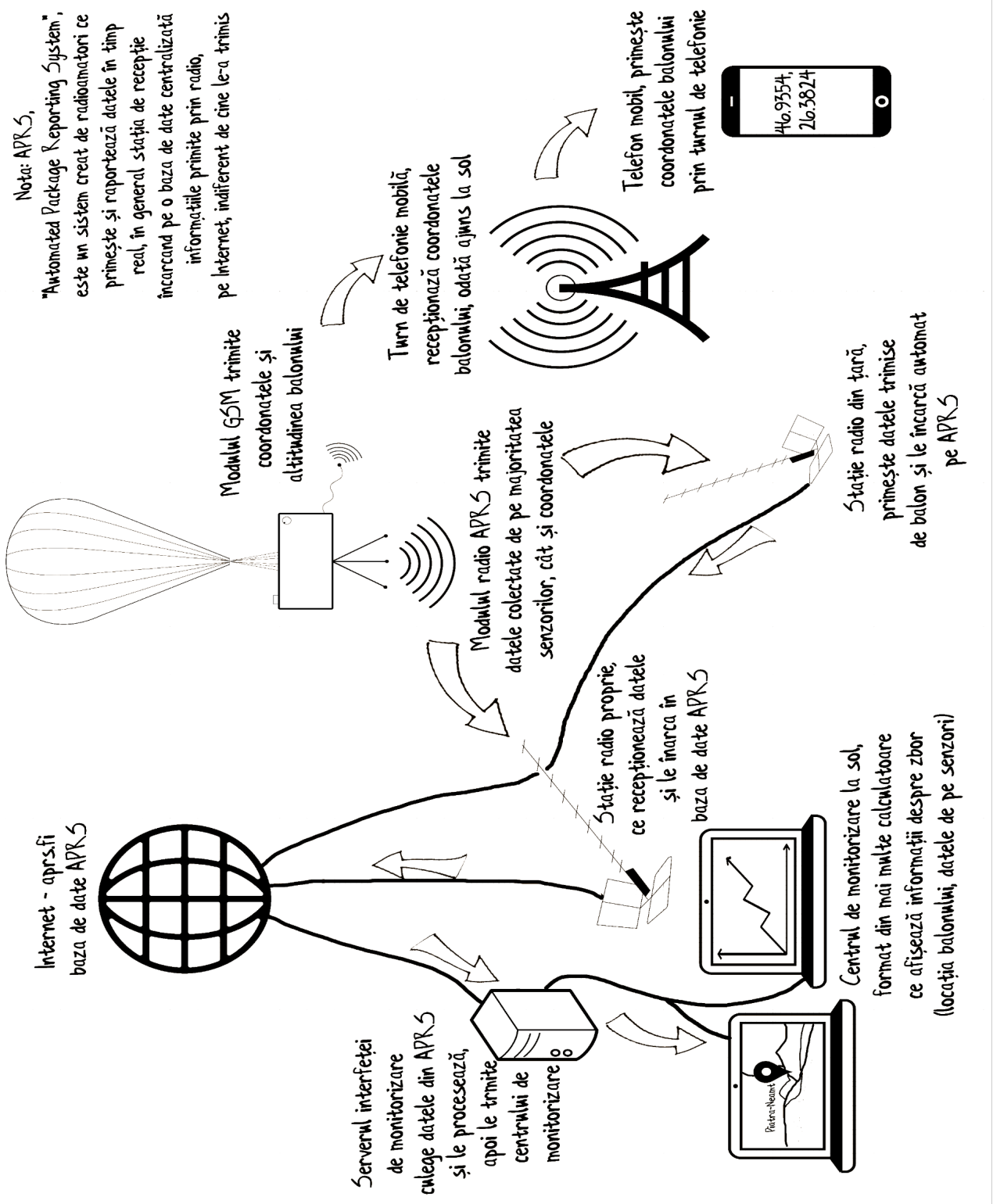
26.7,974.44,4656.15N,02622.95E,337.5,26.38,285,135,24.91,53.76,3.-1,-1
B1G0D1T1S1Ba0;TS285;135;DS26.38;BAT-1;-1

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.

[9]Toate datele ce nu au putut fi citite/sunt eronate sunt înlocuite cu *–1* și sunt trimise așa către APRS

Colectarea datelor din aer



*Centrul de monitorizare de la sol este pe GitHub /GigaCloud/C4TSWeb, dar probabil nu va avea niciodată documentație. Pentru injurii: radu.craciun420@gmail.com

Pentru recepționarea datelor și încărcarea lor în APRS s-a folosit un modul RTL-SDR + Direwolf.

Ca să introduci date în APRS Direwolf-ul trebuie configurat în modul I-GATE (direwolf.conf).

Modulul RTL-SDR poate fi citit cu SDRSharp pe Windows sau cu Gqrx pe Linux.

După ce recepționezi semnalul audio APRS, îl trimiți către Direwolf printr-un sound pipe (Virtual Audio Cable).

Pentru urmarirea pachetelor APRS se poate folosi aprs.fi sau aprsdirect.com

Alte recomandări:

- Google este prietenul tău
- Stackoverflow este prietenul tău
- Github este prietenul tău
- Wikipedia este prietena ta
- Tutorialele de la indieni/chinezi/alți asiatici pe YouTube sunt prietenele tale
- Nu crede că nu merge GSM-ul înainte de a verifica dacă SIM-ul are opțiune activă
- APRS-ul e mai ciudat documentat și sucit decât tentativa asta tristă de documentație
- Mereu asigură-te ca ai antenele conectate înainte de a face emisie/recepție, altfel te alegi cu o sobă cârâitoare
- Nu modifica hardware-ul cât are o sursă de alimentare conectată (doar dacă ești șmecher și-ți place adrenalina)
- Învață C în primul rând ca limbaj, nu ca mijloc de algoritmică (dacă îți place să faci chestii, nu doar să rezolvi probleme, problemele trebuie evitate în viață) – *The C Programming Language*
- Învață Python dacă te-ai săturat să dai la manivelă în C
- Nu-i spune lui Daniel „domnul Daniel” fiindcă sună ciudat; ori „Daniel”, ori „domnul Bejan”
- Nu-ți pune istoricul proiectului DOAR pe Facebook fiindcă e posibil să dispară subit
- Ai rabdare cu Daniel fiindcă și el are răbdare cu tine
- Baftă!