**ROMÂNIA**

**MINISTERUL APĂRĂRII NAȚIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**Facultatea de Sisteme Informatice și Securitate Cibernetică**

**Departamentul de Calculatoare și Securitate Cibernetică**



***Utilizare senzor digital de temperatura si umiditate***

***Si senzor analogic de lumina***

***Platforma de dezvoltare frdm-kl25z***

Std. sg. maj. Ogrean Flaviu Razvan

Std. sg. maj. Mocanu Stefan

Std. sg. maj. Calugaru Diana Alina

Grupa C114C + C114A

**București**

Cuprins

[Cuprins 2](#_Toc123572073)

[1. Prezentarea senzorului DFR0067 3](#_Toc123572074)

[2. Prezentarea senzorului DFR0026 4](#_Toc123572075)

[3. Scop proiect 5](#_Toc123572076)

[4. Conectare senzor – placă de dezvoltare 5](#_Toc123572077)

[5. Abordare problema 7](#_Toc123572078)

[6. Descriere program pentru senzorul DFR0067 7](#_Toc123572079)

[6.1. Descriere program mbed 7](#_Toc123572080)

[6.1.1. Detaliere DHT.h (implementare mbed) 8](#_Toc123572081)

[6.2. Descriere program CMSIS 11](#_Toc123572082)

[6.2.1. Configurare PIT 11](#_Toc123572083)

[6.2.2. Configurare GPIO 12](#_Toc123572084)

[6.2.3. Configurare UART 14](#_Toc123572085)

[6.2.4. Detaliere dht.h 15](#_Toc123572086)

[7. Descriere program pentru senzorul DFR0026 17](#_Toc123572087)

[Functia main 17](#_Toc123572088)

[7.2. Implementarea in python 18](#_Toc123572089)

[8. Analiza rezultatelor pentru senzorul DFR0067 19](#_Toc123572090)

[8.1. Rezultate implementare mbed 19](#_Toc123572091)

[8.2. Rezultate implementare cmsis 20](#_Toc123572092)

[8.3. Rezultate python (implementare mbed) 21](#_Toc123572093)

[9. Analiza rezultatelor pentru senzorul DFR0026 22](#_Toc123572094)

[9.1. Rezultate python 23](#_Toc123572095)

# Prezentarea senzorului DFR0067

Senzorul DFR0067 este un senzor digital pentru temperatura si umidate.

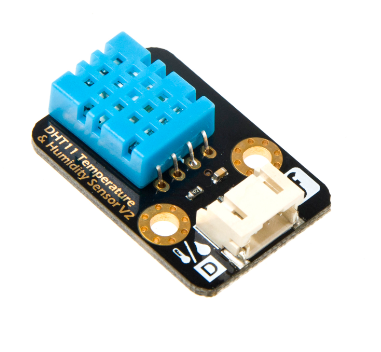


Figura 1. Senzorul DFR0067

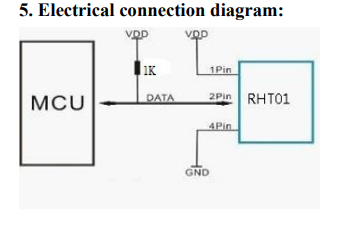


Figura 2. Circuitul senzorului DFR0067

Comunicarea se realizeaza pe un singur fir de date si incepe de indata ce microcontrolerul termina de terminat semnalul de start. Senzorul va raspunde cu 40 biti de date astfel:

* 16 biti umiditate (parte intreaga<8biti> + parte fractionara<8biti>)
* 16 biti temperatura (parte intrega<8biti> + parte fractionara<8biti>)
* 8 biti suma de control

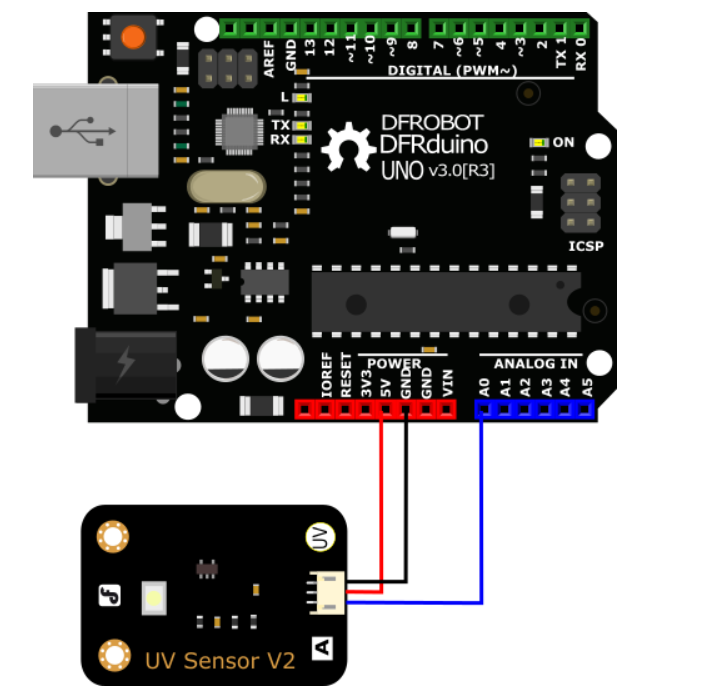
Descrierea detaliata a comunicarii va fi evidentiata mai jos odata cu atasarea segmentelor de cod

# Prezentarea senzorului DFR0026

Senzorul DFR0026 este un senzor analogic pentru lumina.



*Figura 3. Senzorul DFR0026*



*Figura 4. Exemplu conectare senzor*

# Scop proiect

Scopul acestui proiect este acela de a capta valorile furnizate de senzor pentru umiditate, temperatura si lumina, urmand a fi afisate intr-o interfata grafica pentru a putea fi analizate.

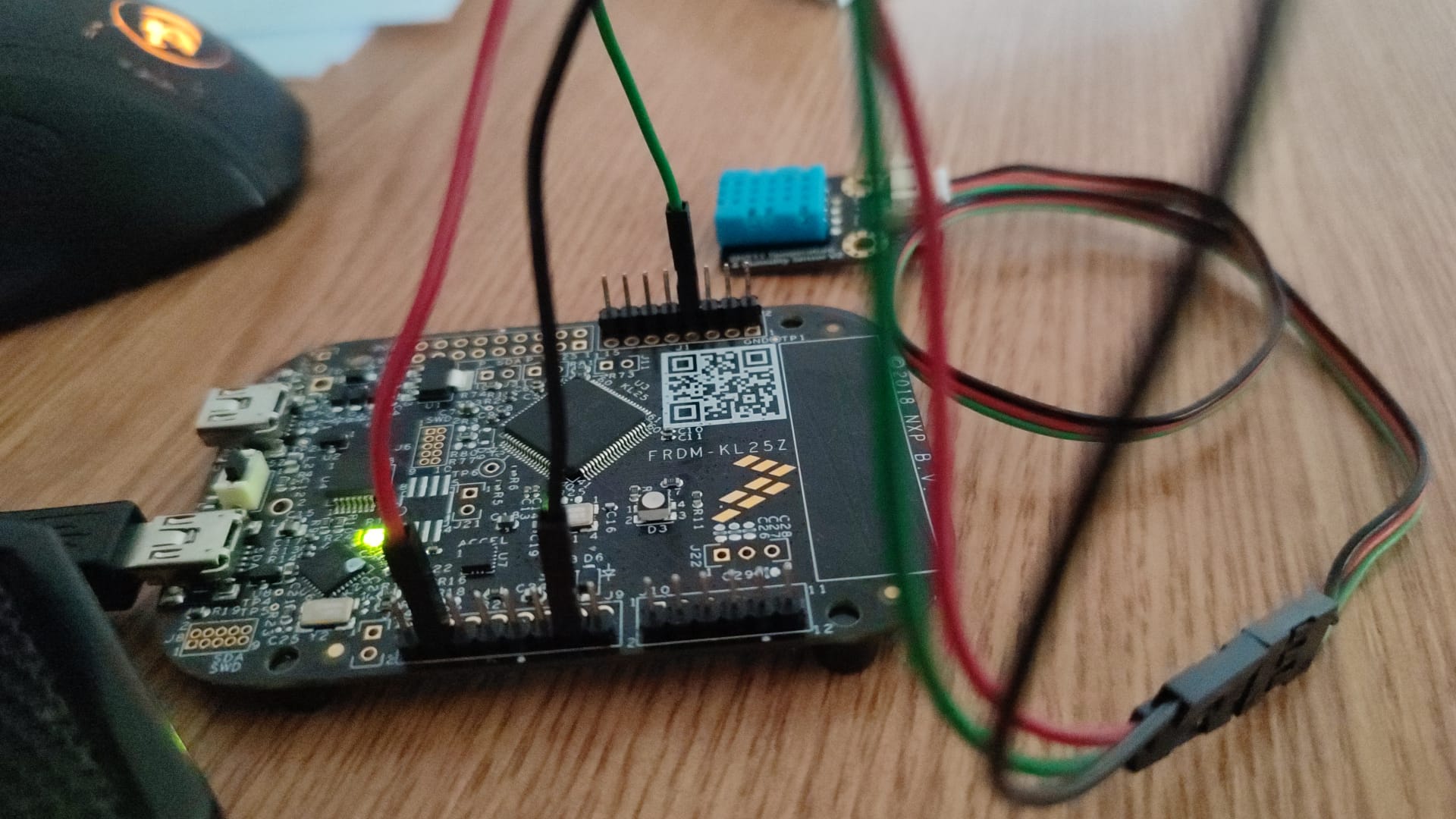
In cazul senzorului de umiditate si temperatura se va dezvolta un program care, o data la fiecare secunda, va verifica ieșirea senzorului curent si va converti valorile in unitatile de masura dorite pentru umiditate – RH – si temperatura – grade Celsius. De asemenea, se va transmite prin UART către PC valoarea ieșirii senzorului pentru a se putea realiza un grafic în timp real în care să se observe variatia valorilor de interes. Graficul va fi implementat folosind python.

In cazul senzorului de lumina prin intermediul programului dezvoltat datele sunt preluate (am setat ca datele sa fie preluate la fiecare 0,3 secunde) si transmise catre PC pentru a putea fi reprezentate sub forma unui grafic implementat folosind python, care sufera schimbari in timp real in functie de inputul primit. Pe langa reprezentarea sub forma unui graf a valorilor obtinute, trebuie ca ledul de pe target sa reactioneze in functie de fluctuatiile datelor.

# Conectare senzor – placă de dezvoltare

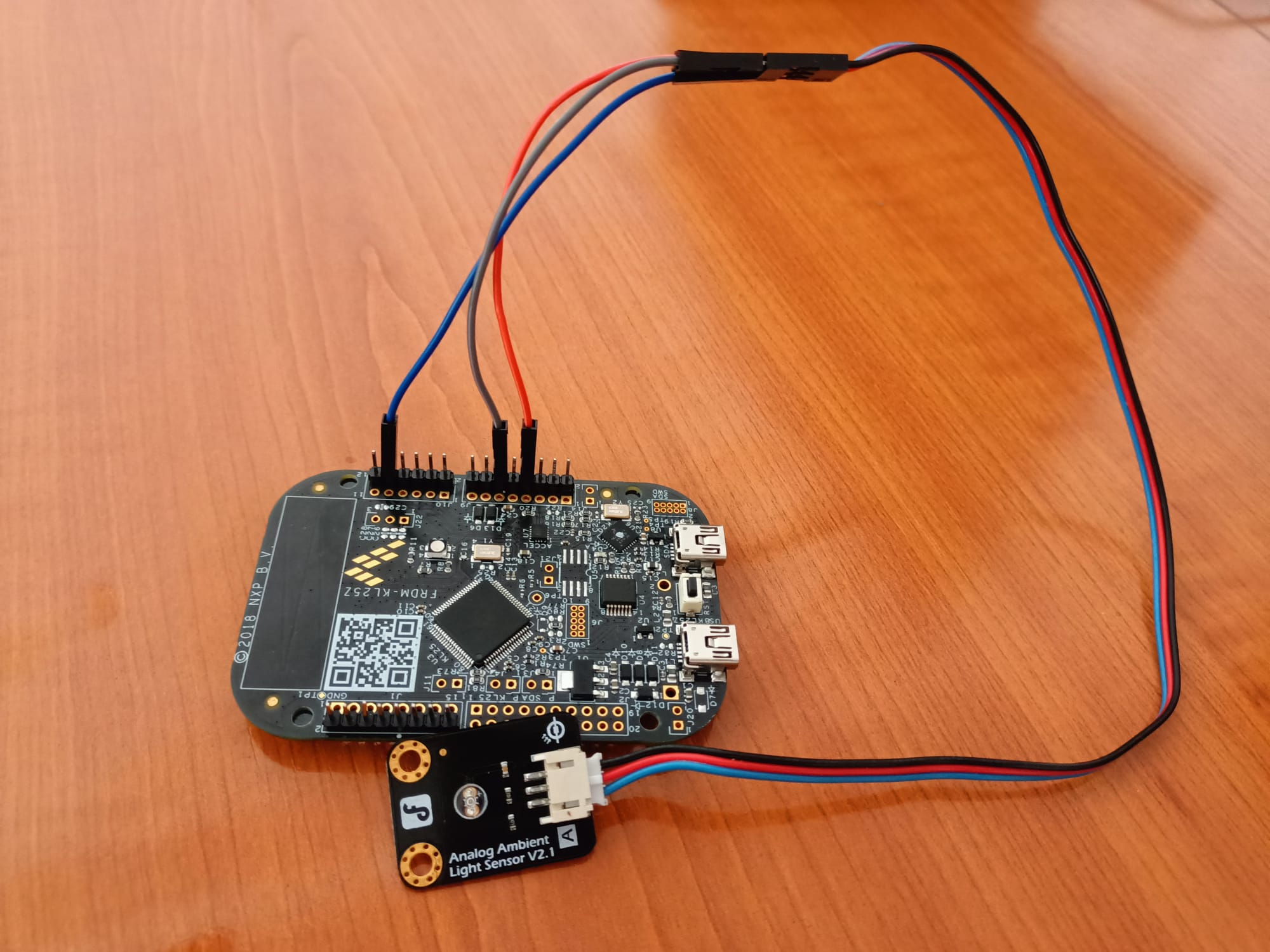
Vom conecta senzorul DFR0067 astfel:

* Firul verde conectează senzorul la PTA12
* Firul roșu conectează senzorul la 3.3V
* Firul negru conectează senzorul la GND



Vom conecta senzorul DFR0026 astfel:

* Firul albastru conectează senzorul la PTC2
* Firul roșu conectează senzorul la 3.3V
* Firul gri conectează senzorul la GND



# Abordare problema

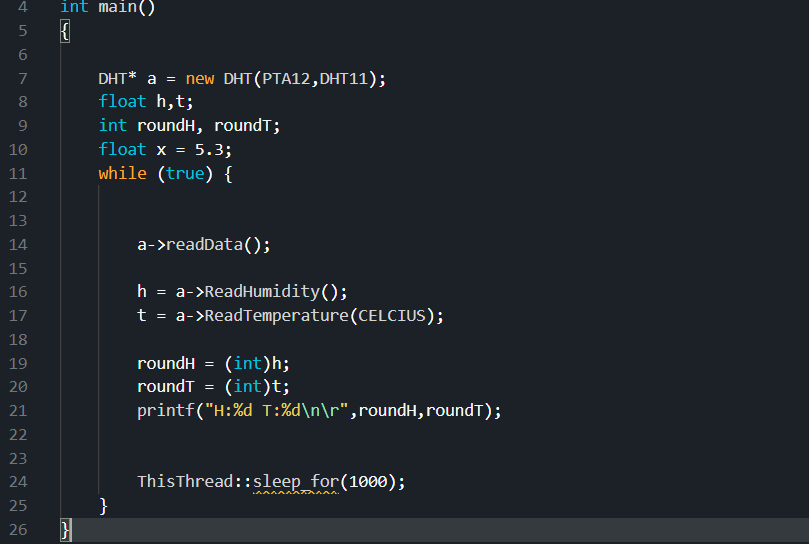
In cazul senzorului DFR0067 am incercat rezolvarea problemei prin 2 metode: folosind strict CMSIS, apoi, folosind biblioteca mbed.h. Prima varianta nu a fost dusa la final cu succes. Am reusit captarea unui calup de date relevant de la senzor, insa nu am reusit mentinerea constanta a comunicatiei. Folosind cea de a doua varianta am realizat cu succes proiectul. Ambele variante urmeaza a fi detaliate mai jos, incepand cu cea care foloseste biblioteca mbed.

In cazul senzorului DFR0026 m-am folosit de materialele puse la dispozitie in cadul laboratoarelor, astfel incat inputul analogic este obtinut prin intermediul functiei ADC0\_Read(), iar mai apoi transmis print intermediul functiei UTILS\_Print(), dar nu inainte de a fi transformat semnalul analogic in digital.

# Descriere program pentru senzorul DFR0067

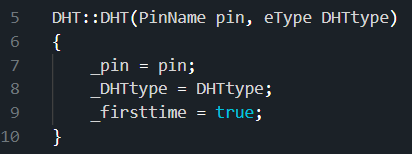
## Descriere program mbed

În fișierul main.c am inclus fișierele header în care sunt declarate funcții și variabile ce urmează a fi folosite: *mbed.h.* si DHT.h.

Logica principală a programului este următoarea: intai se creaza un obiect de tip DHT care urmeaza sa comunice cu senzorul pe pinul PTA12 (tipul senzorului DHT11) ; apoi, in cadrul unui loop infinit, se citesc valorile furnizate de senzor (apelul functie readData() din clasa DHT), se extrag valorile specifice umiditatii (ReadHumidity()) si temperaturii(ReadTemperature), se rotunjesc valorile intrucat zecimalele nu sunt de itneres proiectului actual si se transmit spre host folosind comunicatia seriala uart; inainte de a relua pasii, programul asteapta 1 secunda. 

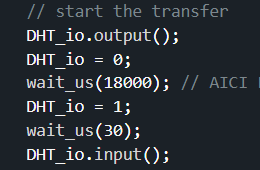
## Detaliere DHT.h (implementare mbed)

Clasa DHT pune la dispozitie un constructor in care se specifica pinul pe care se va comunica si tipul senzorului (DHT11/DHT22).



Functia principala readData() realizeaza etapele necesare comunicarii mentionate in datasheet-ul senzorului.

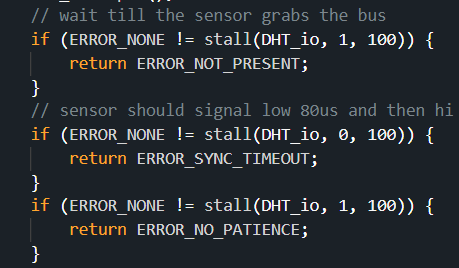
1). Se transmite semnalul de start: 18ms LOW, urmat de 30us HIGH.

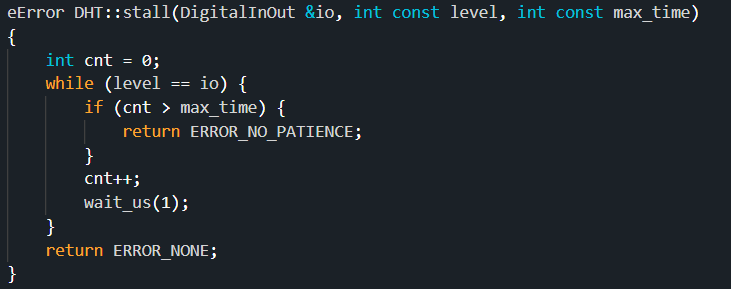


2). Cand senzorul va detecta semnalul de start va transmite pe canalul de date LOW timp de 80us, urmate de alte 80us HIGH.

Pentru a capta aceasta secventa de date, asteptam cat timp pe canalul de date inca se transmite HIGH. Odata cu inceperea semnalului LOW inseamna ca senzorul isi incepe secventa de 80us LOW.

Facem pooling pe datele transmise pe canalul de comunicatie pentru maxim 100us, urmarind schimbarea LOW -> HIGH. Apoi repetam procesul urmarind schimbarea LOW->HIGH care reprezinta confirmarea din aprtea senzorului ca a primit semnalul de start.

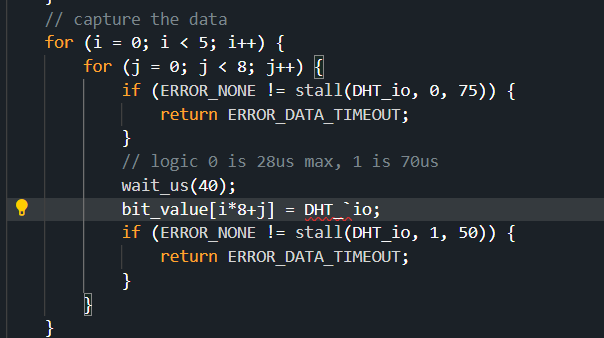




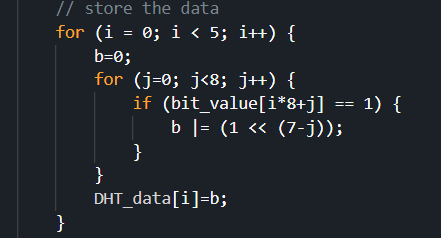
3). Urmeaza ca cei 40 de biti de date sa fie transmisi astfel: inainte de fiecare bit senzorul transmite LOW 50us, apoi transmite HIGH. Daca transmite HIGH mai putin de 28us inseamna ca bitul are valoare 0, daca transmite mai mult (70us) inseamna ca bitul are valoarea 1.

Asa ca, in cod, vom executa de 4 ori urmatoarea seccevnta:

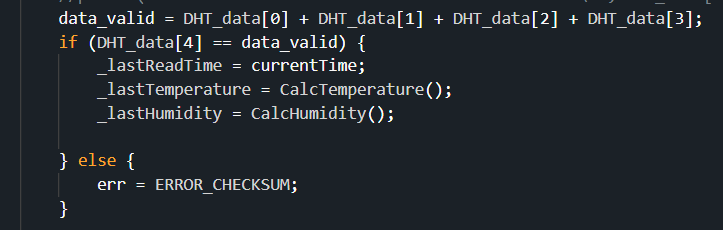
* Asteptam maxim 75us trecerea de la LOW->HIGH
* Asteptam 40us
* Verificam daca dupa cele 40us inca se mai transmite HIGH (in caz afirmativ am primit ‚1’, in caz negativ am primit ‚0’)
* Asteptam finalizarea semnalului HIGH maxim de inca 50us



4). Stocam datele in cadrul unui mebru al clasei DHT (datele primate sunt reprezentarile binare pentru valorile RH, respective Celsius; deci nu necesita conversii).



5). Verificam suma de control si stocam valorile de itneres in variabilele clasei care vor fi expuse in afara acesteia.



Si astfel am prelual valorile de la sensor si le-am stocat in cadrul clasei, urmand ca prin intermediul functiilor ReadHumidity() si ReadTemperature() sa poate fi accesate.

## Descriere program CMSIS

Pentru a implementa aceleasi functionalitati fara biblioteca mbed, am incercat refacerea functiilor high level din cadrul proiectul mbed prezentat mai sus si din cadrul unui alt proeict arduino care indeplineste acelasi scop.

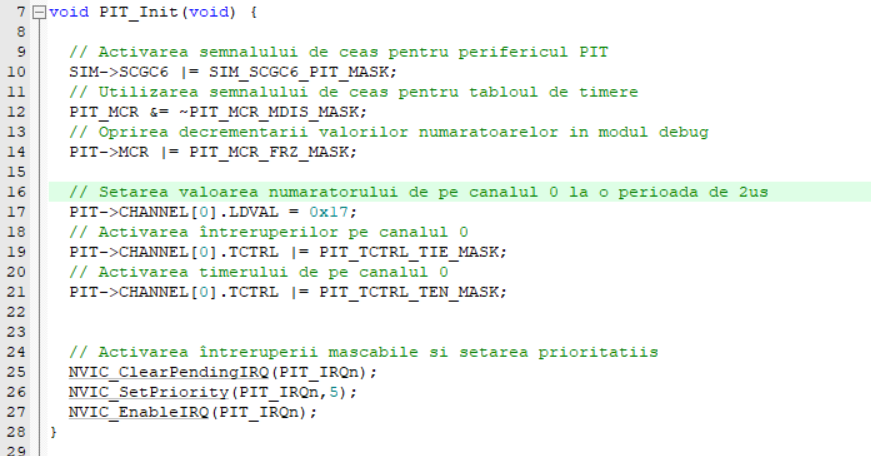
Am avut nevoide de configurarea modulelor:

* PIT, implementarea unui timer
* GPIO, comunicarea efectiva cu senzorul
* UART, transmiterea seriala spre host a datelor primite de la senzor

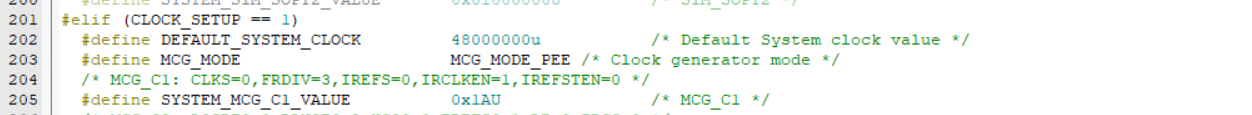
## Configurare PIT

Modulul PIT ne permite generarea de intreruperi dupa o durata fixa de timp, astfel ne permite implementarea unor timere.

Configurarea modului:



In cadrul fisierului header system\_MKL25z4.h am setat CLOCK\_SETUP egal 1.



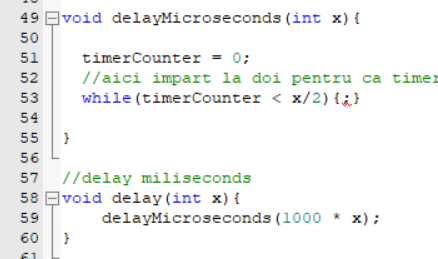
Apoi, am calculat valoarea necesara pentru a incarca numaratorul intern dupa formula:

X = nr.Sec \* bus\_clock\_freq – 1 à 0x17 = 2 \* 10^(-6) \* 48 \* 10^(6) (LDVAL register)

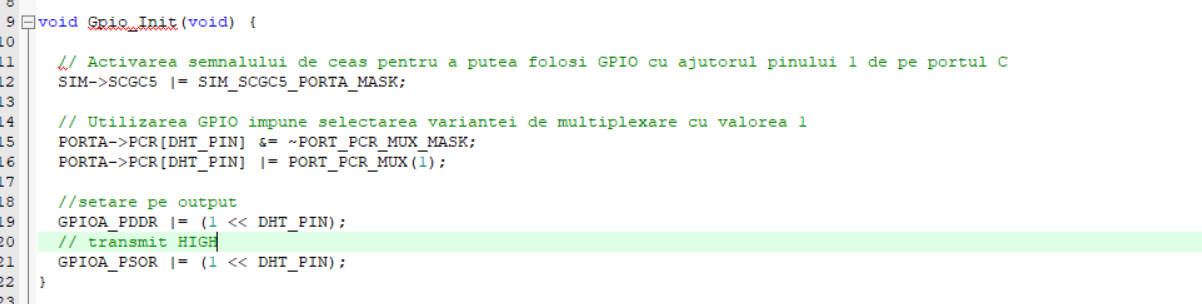
In cadrul interuperii realizate de PIT la fiecare 2us exista 2 variabile incrementate incontinuu:

* tiemrCounter; valoare resetata de fiecare data cand vrem sa incepem incepem o cronometrare
* microCounter; valoare care retine nuamrul de microsecunde de la inceputul rularii programului

In headerul Pit.h, pe langa configurarea modului, am adaugat 2 functii care fac busy waiting un anumit numar de microseconds/miliseconds

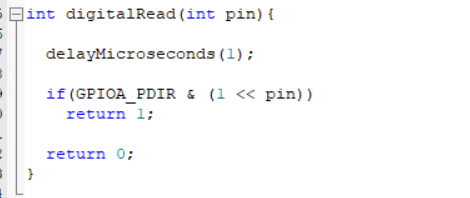


## Configurare GPIO

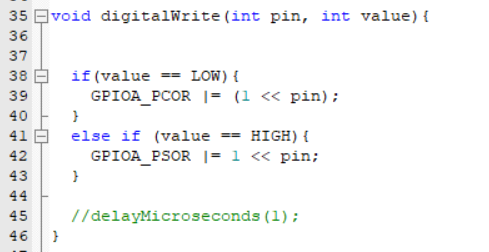
Folosim modulul GPIO pentru a comunica cu senzorul. Initalizam modulul:

Implementare metode pentru:

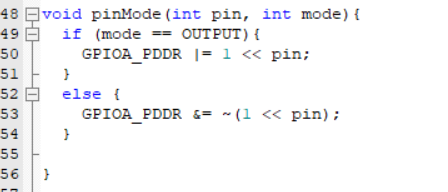
* Citirea datelor din registrul de input specific unui anumit pit



* Scrierea datelor in registrul de output specific unui anumit pin

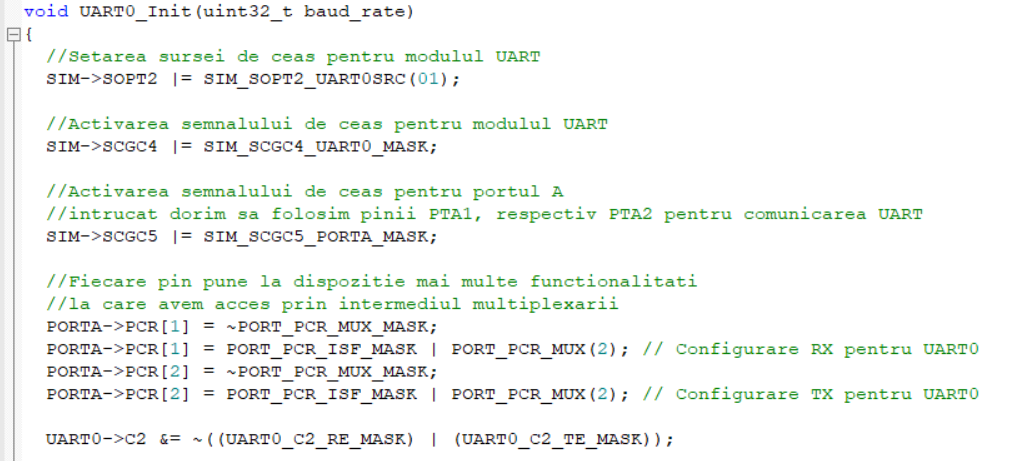


* Setarea modului de lucru (Input/Output)

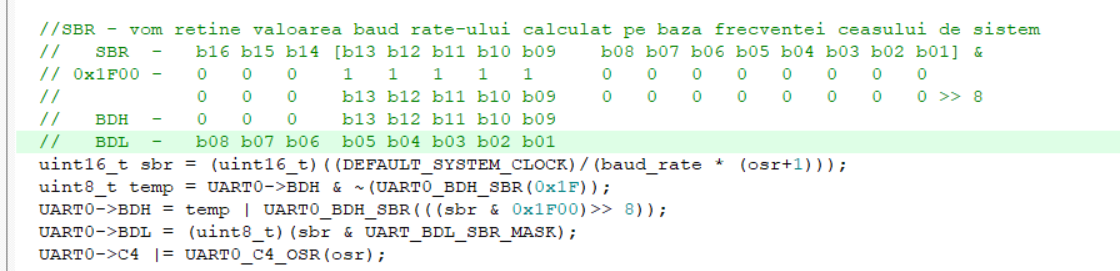


## Configurare UART

Folosim UART0 pentru comunicatia seriala in transmiterea datelor de interes spre host si pentru facilitarea procesului de depanare.

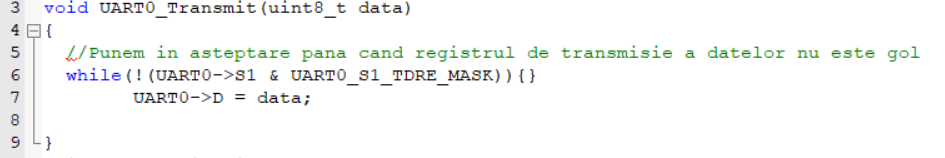
Initializare:  


Setare baud-rate:

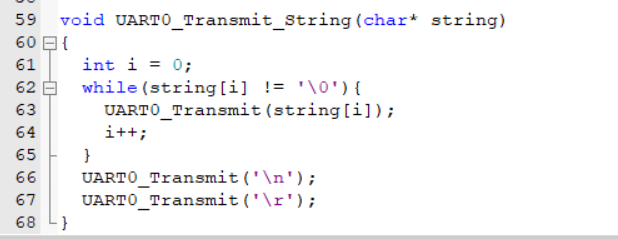


Am implementat 2 metode pentru transmiterea de date. Nu avem nevoie sa primim date.

Trimtierea unui caracter:



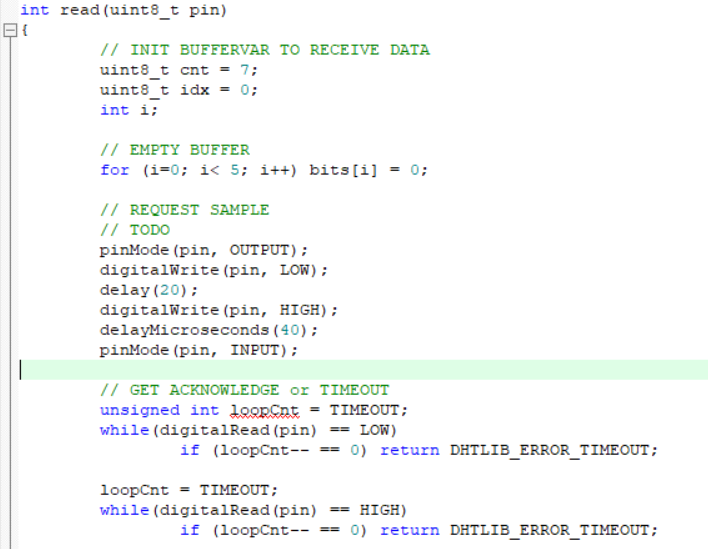
Trimiterea unui sir de caractere:



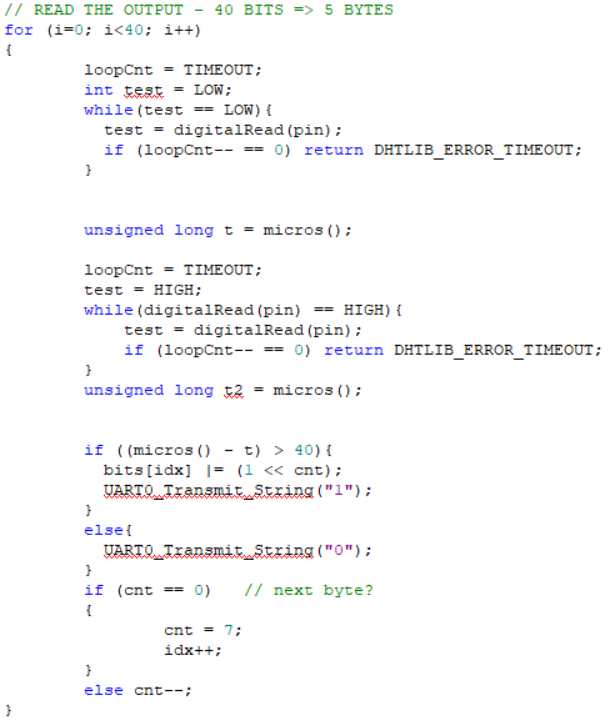
## Detaliere dht.h

Headerul dht.h parcurge aceeasi pasi secventiali descrisi anteriori in capitolul despre implementarea folosind mbed. Diferenta este aceea ca in acest proiect folosim functiile descrise mai sus din cadrul modulelor pit, gpio si uart pentru realizarea obiectivelor.

Semnalul de start si confirmarea primirii acestuia de catre senzor:



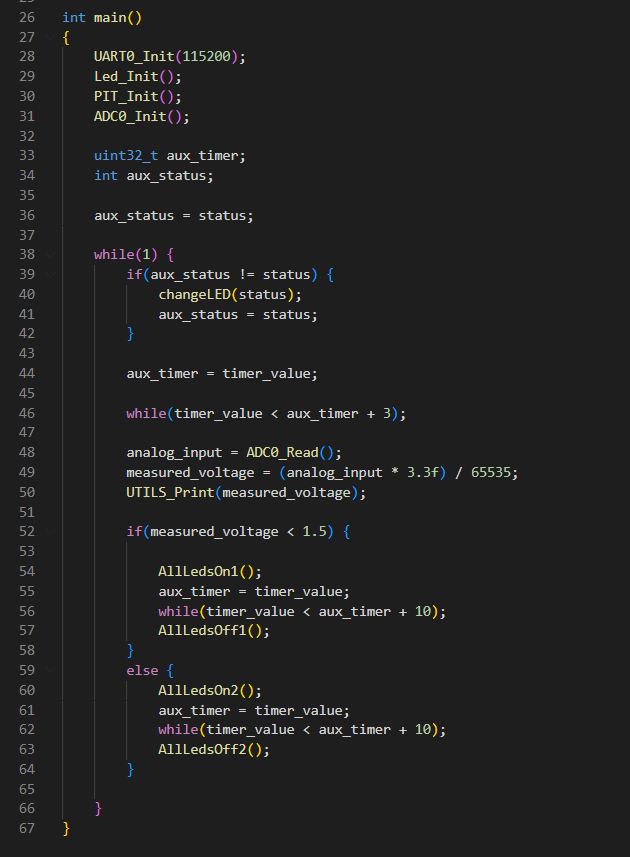
Primirea celor 40 de biti de interes:



# Descriere program pentru senzorul DFR0026

## Functia main

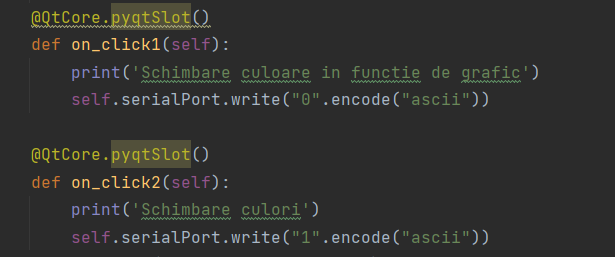
In fisierul main.c am inclus fisierele header Uart.h, gpio.h, adc.h, si pid.h. Dupa cum se poate observa si in imaginea atasata mai jos am apelat functiile de initializare UART0\_Init, Led\_Init(), PIT\_Init() si ADC0\_Init(). Apoi intr-un while infinit, la primirea unui semnal de la PC la target (functie dezvoltata in codul python, astfel incat la apasarea unui buton de pe grafic acesta sa tarnsmita un semnal care sa fie receptionat de target) sa se apeleze functia changeLed() care are rolul de a schimba culoarea ledului de pe placa (in cazul nostru culorile schimbate sunt rosu si galben). In continuare se face o adunare cu o marja de 0,3 ms la transmiterea datelor, deoarece am identificat o problema in faptul ca initial erau transmise prea repede, astfel graficul nu era o reprezentare a datelor relevanta pentru timpul curent. In variabila analog\_input se pune rezultatul de la functia ADC0\_Read() , apoi se transforma semnalul analog in digital si se transmite prin intermediul functiei UTILS\_Print(). Codul se continua cu o verificare, astfel daca measured\_voltage este mai mare decat valoarea 1.5, atunci se apeleaza functia AllLeadsOn1() care are rolul de a aprinde lentul doar in culoarea rosu, in caz contrar se apeleaza functia AllLeadsOn2() care are rolul de aprinde ledul in culoarea verde.



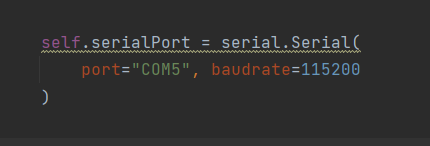
In cazul functiilor de initializare acestea sunt preluate din materialele puse la dispozitie pe parcursul laboratoarelor.

## Implementarea in python

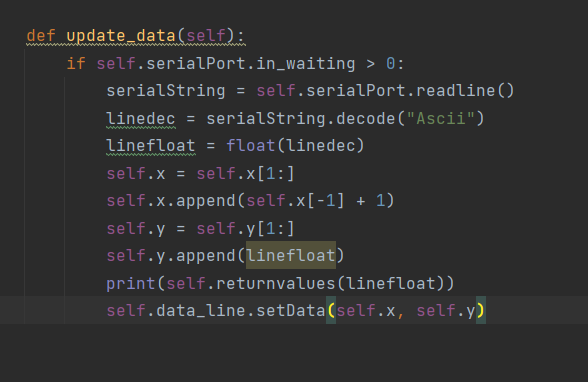
In cadrul implemnetarii programului in python, initial am construit graficul unde urmau sa se reprezinte datele colectate in timp real. Apoi am continuat cu implementarea a doua butoane, care prin click aveau rolul de a controla aprinderea ledurilor in diferite culori, astfel prin apasarea butonului Change All se transmitea un semnal echivalent cu valoarea “1” care avea drept echivalenta in codul din keil culorile galben si rosu pentru led, in schimb daca apasam butonul Change atunci se transmitea un semnal echivalent cu valoarea “0” care nu avea o functionalitate propriu-zis, astfel se continua codul din functia main din keil, iar culoarea ledului alterna intre verde si rosu in functie de valorile primite de la senzor (valorile mai mari decat 1,5 erau corespondente culorii rosu la led, iar valorile mai mici erau corespondente culorii verde).



Conectarea la target din codul python se realizeaza prin intermediul portului si a vitezei, astfel:



Pentru a face update mereu la datele receptionate de catre senzorul de lumina, am implementat functia update\_data() care initial verifica conexiunea, apoi se face conversia in ascii, iar dupa in float pentru a putea fi reprezentate datele. In continuare se fac modificari in ceea ce priveste reprezentarea datelor pe grafic, astfel mereu se sterge primul element s se adauga la final urmatorul, pentru a exista mereu o continuitate. Iar la finalul functiei se face update la datele primite.

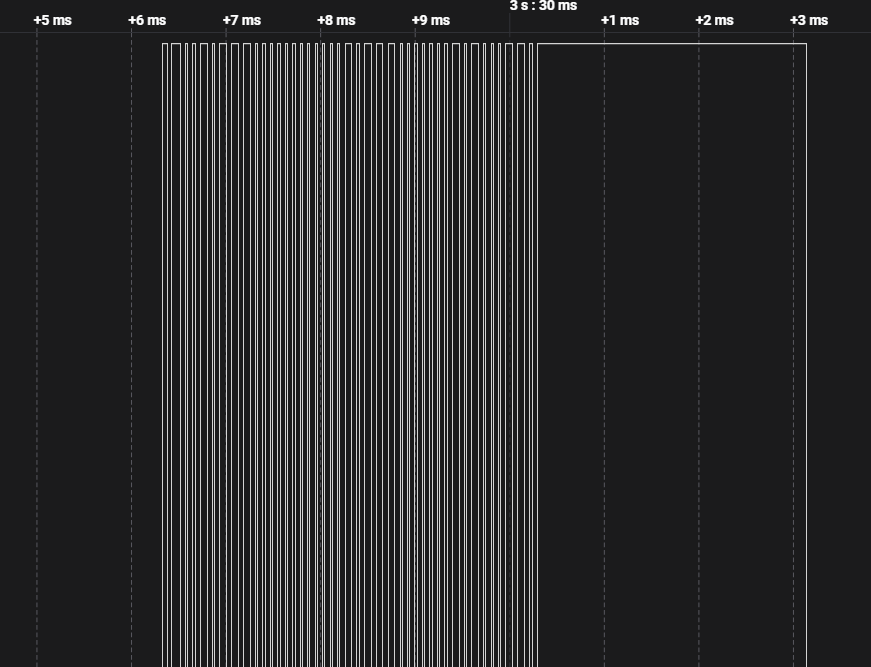


# Analiza rezultatelor pentru senzorul DFR0067

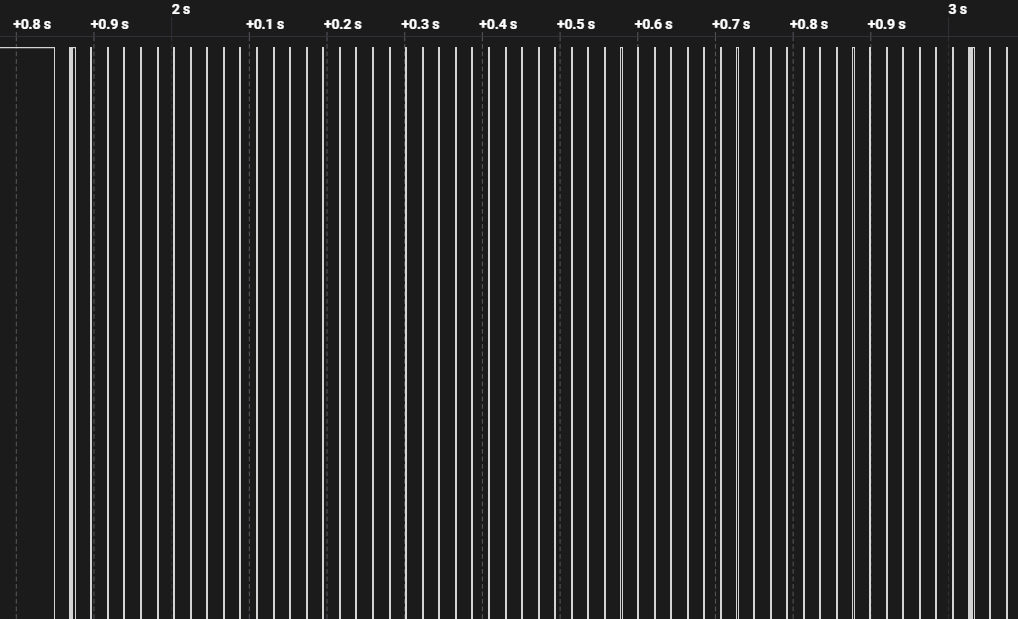
Pe langa folosirea comunicarii seriale pentru a capta datele, am folosit si un dispozitiv extern pentru a capta semnalul de date transmis.

## Rezultate implementare mbed

Transmiterea unui calup de date:

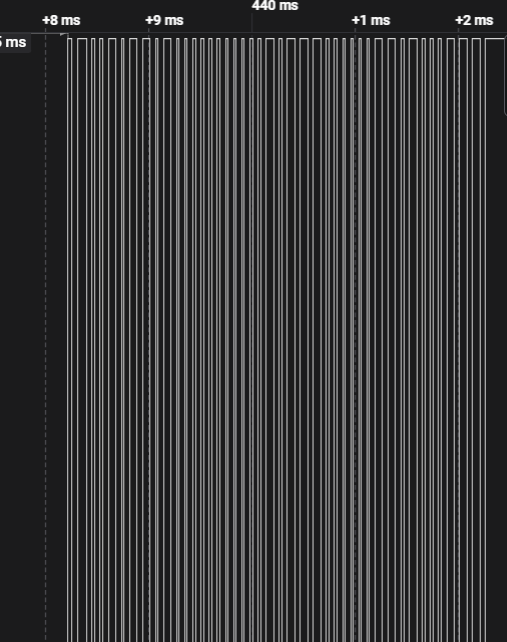


Secventa de mai sus se repeta la fiecare secunda:



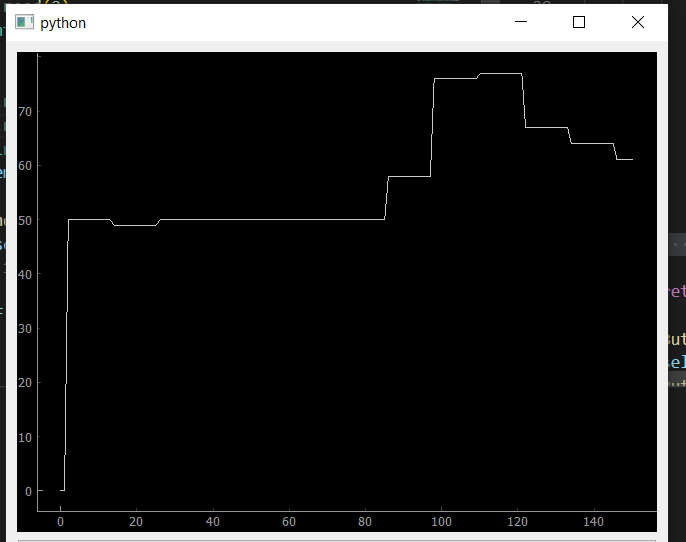
## Rezultate implementare cmsis

Observam captarea unui singur calup de date relevante (repetarea epriodica a comunicarii lipseste)



## Rezultate python (implementare mbed)

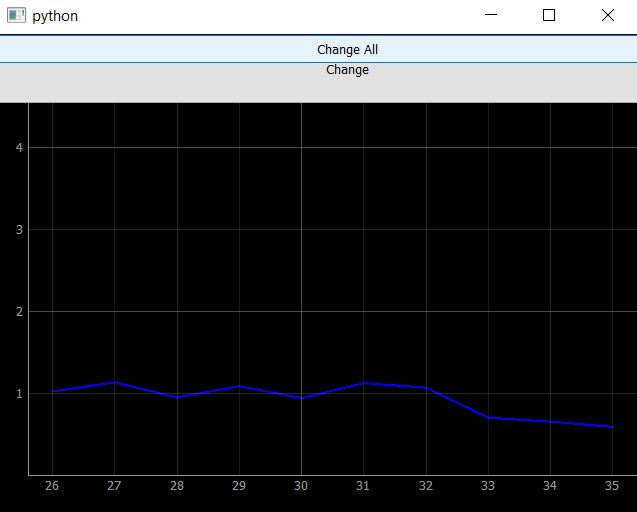
Grafic pe care este reprezentat valoarea umiditatii la fiecare secunda. Pentru variatia valorii, am acoperit senzorul cu mainile aseptandu-ne sa vedem o crestere a umiditatii.



# Analiza rezultatelor pentru senzorul DFR0026

## Rezultate python

In figural de mai jos se observa faptul ca datele primite sunt reprezentate sub forma unui grafic care se modifica in functie de valorile primite. Datele reprezentate au fost receptionate initial in format ASCII, iar pentru a le putea reprezenta grafic a mai existat o conversie in float.



Pentru a testa si observa mai bine datele primite, in cod am facut si un print ale acestora, totodata la apasarea butoanelor am printat si un text sugestiv care sa marcheze acest lucru.

