**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 vibrații***

***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

[1. Prezentarea senzorului DFR0067 3](#_Toc122347773)

[2. Scop proiect 4](#_Toc122347774)

[3. Conectare senzor – placă de dezvoltare 4](#_Toc122347775)

[4. Abordare problema 5](#_Toc122347776)

[5. Descriere program mbed 5](#_Toc122347777)

[5.1. Funcția main (implementare mbed) 5](#_Toc122347778)

[5.2. Detaliere DHT.h (implementare mbed) 6](#_Toc122347779)

[6. Descriere program CMSIS 9](#_Toc122347780)

[6.1. Configurare PIT 9](#_Toc122347781)

[6.2. Configurare GPIO 10](#_Toc122347782)

[6.3. Configurare UART 12](#_Toc122347783)

[6.4. Detaliere dht.h 13](#_Toc122347784)

[7. Analiza rezultatelor 15](#_Toc122347785)

[7.1. Rezultate implementare mbed 15](#_Toc122347786)

[7.2. Rezultate implementare cmsis 16](#_Toc122347787)

[7.3. Rezultate python (implementare mbed) 17](#_Toc122347788)

# Prezentarea senzorului DFR0067

Senzorul DFR0067 este un senzor digital pentru temperatura si umidate.

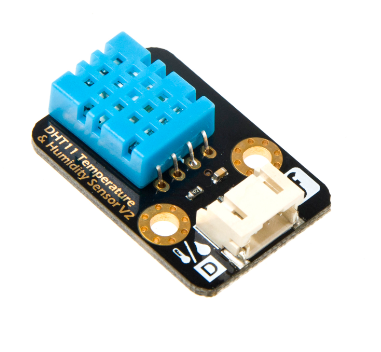


Figura 1. Senzorul DFR0067

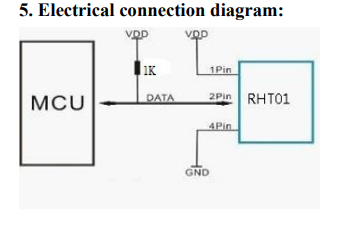


Figura 2. Circuitul senzorului DFR0067

Comunicarea se realizeaza pe un singur fir de date si incepe de indata ce microcontrolerul termina de terminat semnalul de start. Senzorul va raspunde cun 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

# Scop proiect

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

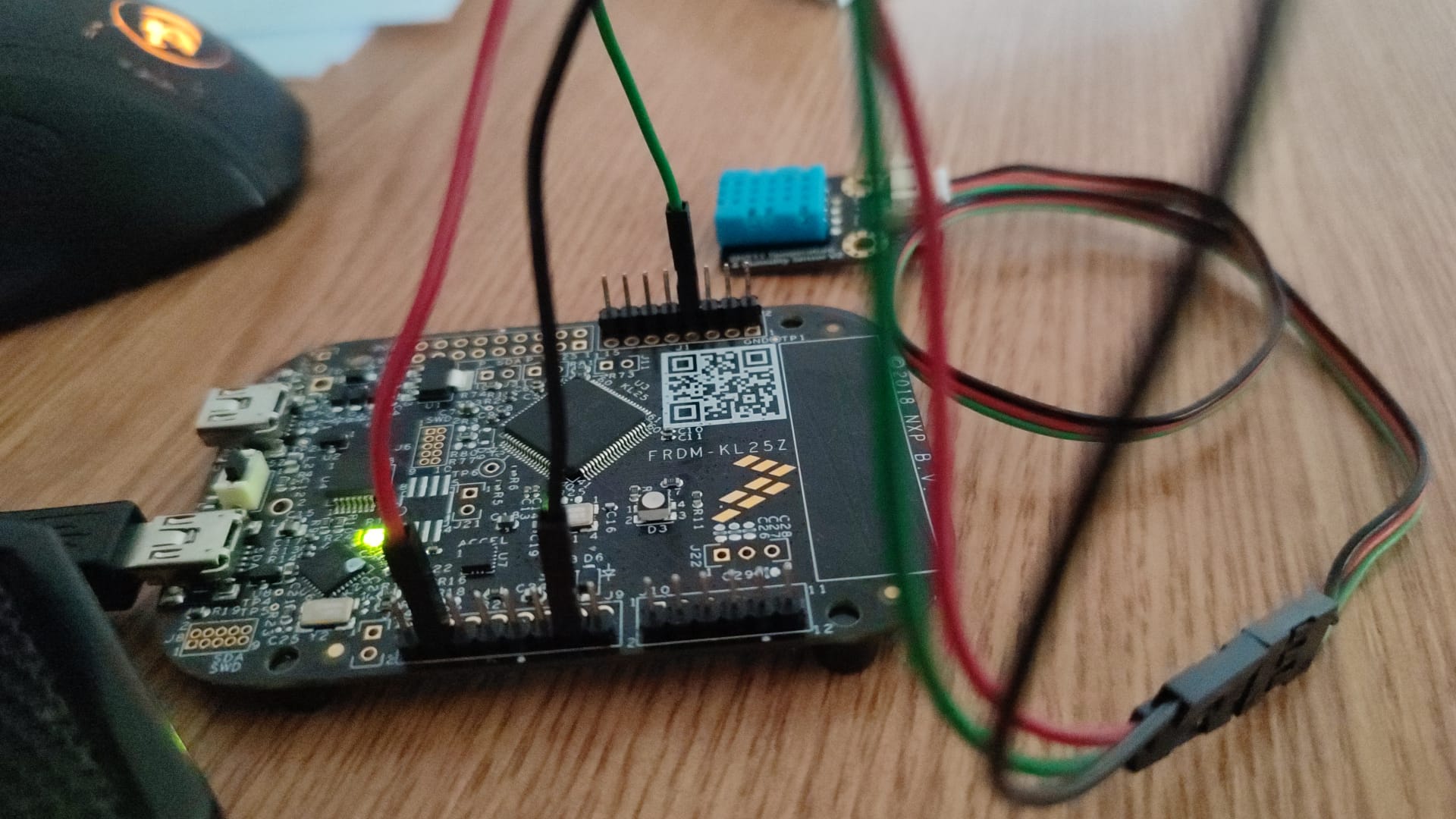
Se va dezvolta un program care,o data la fiecare secunda, va verifica ieșirea senzorului conectat si va converti valorile in unitatile de masura dorite pentru umiditate – RH – si temperatura – grade Celsius.

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 itneres. Graficul va fi implementat folosind python.

# Conectare senzor – placă de dezvoltare

Vom conecta senzorul astfel:

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



# Abordare problema

Am incercat rezolvarea problemei prin 2 metode: folosind strict CMSIS Api, folosind biblioteca mbed.h

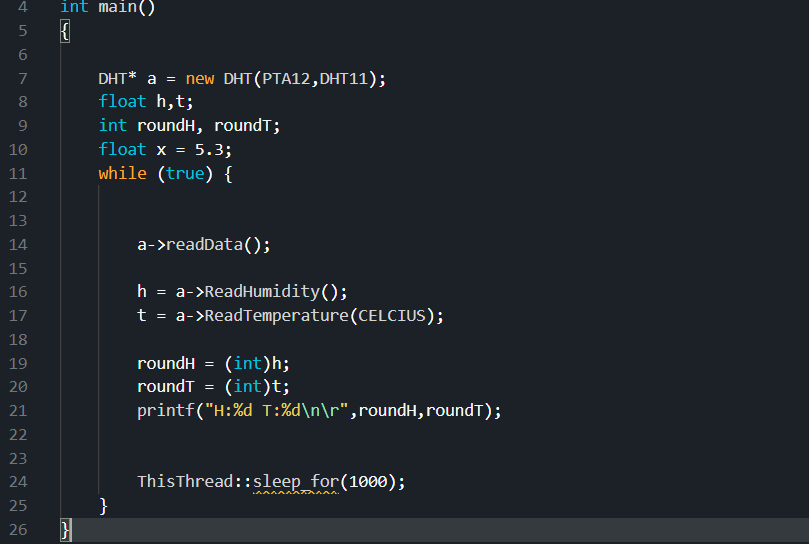
Prima varianta nu a fost dusa la final cu succes. Am reusit captarea unui calup de date relevant de la senzor, insa nu am reusit mentinerea constanta a comunicatiei. Folosind cea de a doua varianta am realizat cu succes proiectul. Ambele variante urmeaza a fi detaliate mai jos, incepand cu cea care foloseste biblioteca mbed.

# Descriere program mbed

## Funcția main (implementare mbed)

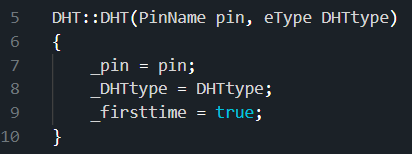
În fișierul main.c am inclus fișierele header în care sunt declarate funcții și variabile ce urmează a fi folosite: *mbed.h.* si DHT.h.

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.



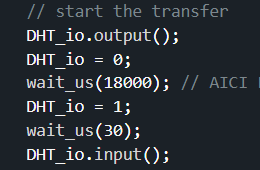
## Detaliere DHT.h (implementare mbed)

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.

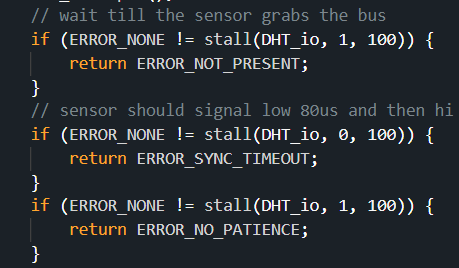
1). Se transmite semnalul de start: 18ms LOW, urmat de 30us HIGH.

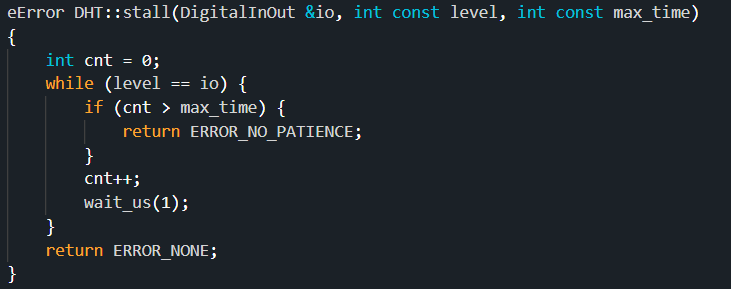


2). Cand senzorul va detecta semnalul de start va transmite pe canalul de date LOW timp de 80us, urmate de alte 80us HIGH.

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.

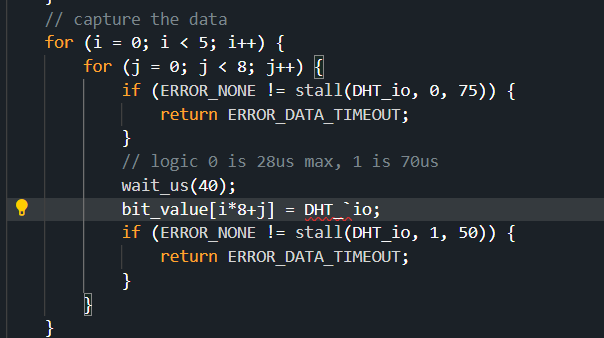




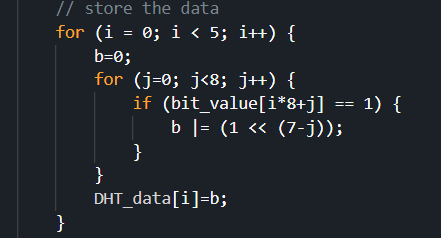
3). Urmeaza ca cei 40 de biti de date sa fie transmisi astfel: inainte de fiecare bit senzorul transmite LOW 50us, apoi transmite HIGH. Daca transmite HIGH mai putin de 28us inseamna ca bitul are valoare 0, daca transmite mai mult (70us) inseamna ca bitul are valoarea 1.

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

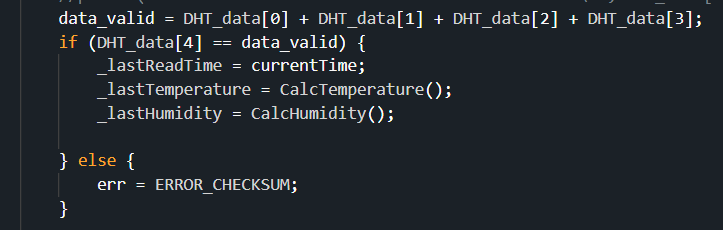
* Asteptam maxim 75us trecerea de la LOW->HIGH
* Asteptam 40us
* Verificam daca dupa cele 40us inca se mai transmite HIGH (in caz afirmativ am primit ‚1’, in caz negativ am primit ‚0’)
* Asteptam finalizarea semnalului HIGH maxim de inca 50us



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.

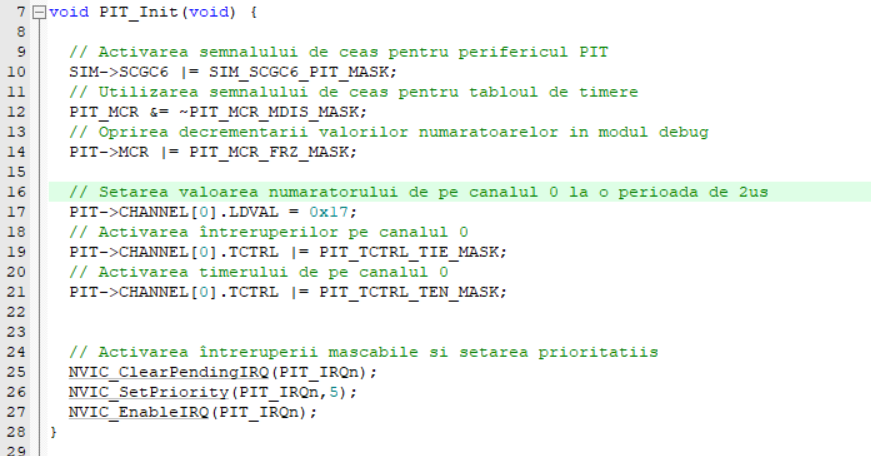
Am avut nevoide de configurarea modulelor:

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

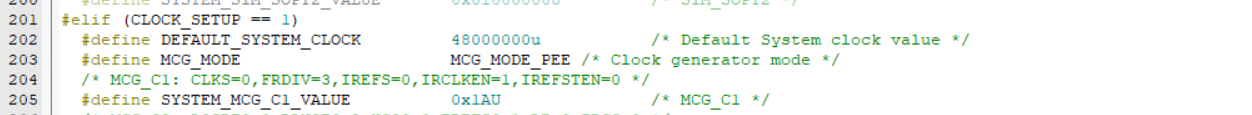
## Configurare PIT

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.



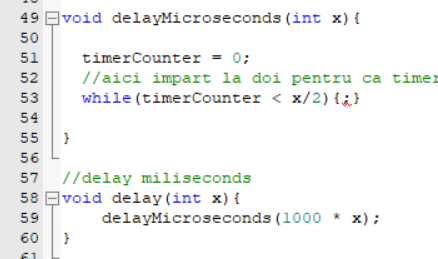
Apoi, am calculat valoarea necesara pentru a incarca numaratorul intern dupa formula:

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

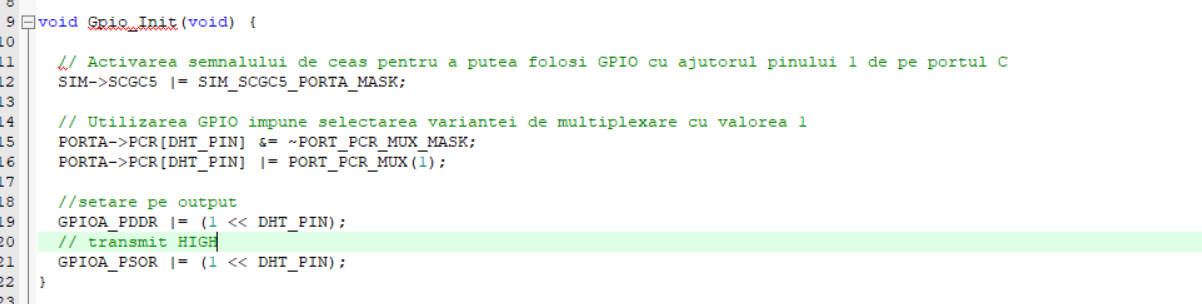
In cadrul interuperii realizate de PIT la fiecare 2us exista 2 variabile incrementate incontinuu:

* 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

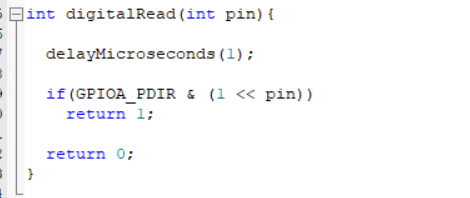


## Configurare GPIO

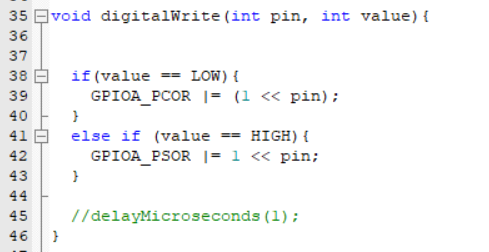
Folosim modulul GPIO pentru a comunica cu senzorul. Initalizam modulul:

Implementare metode pentru:

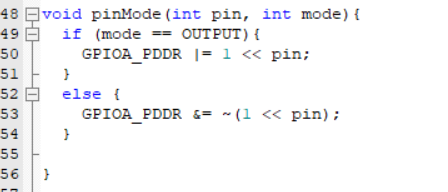
* Citirea datelor din registrul de input specific unui anumit pit



* Scrierea datelor in registrul de output specific unui anumit pin

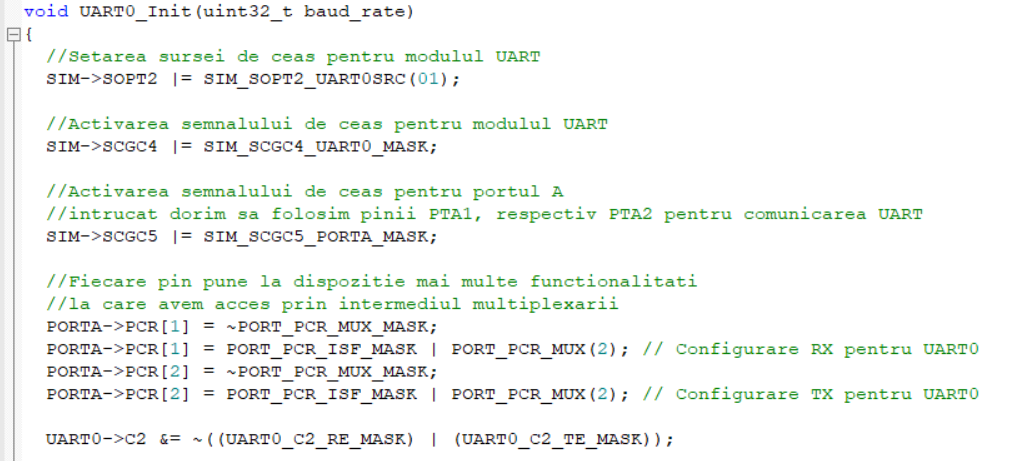


* Setarea modului de lucru (Input/Output)

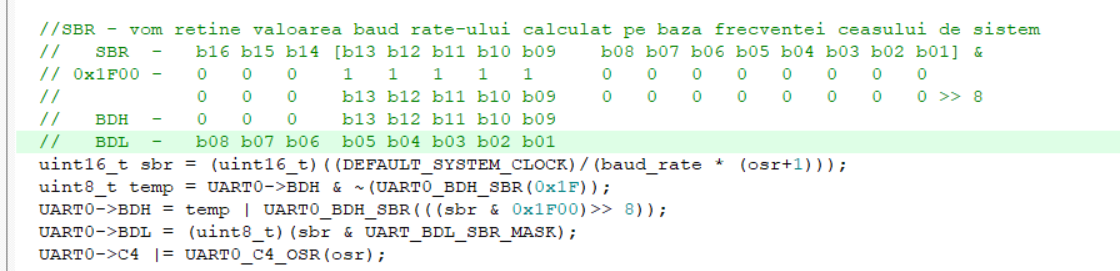


## Configurare UART

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

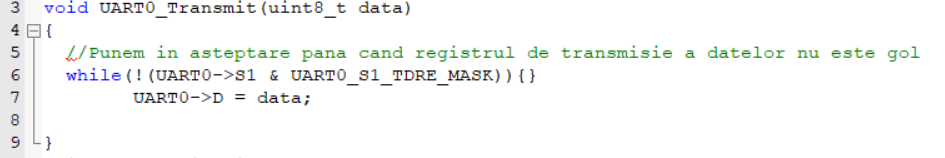
Initializare:  


Setare baud-rate:

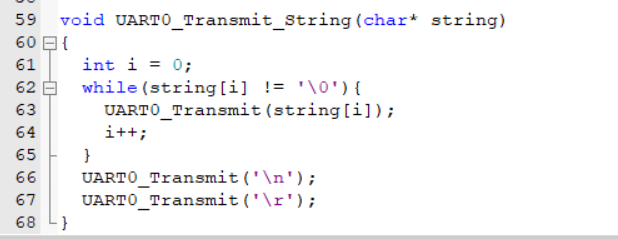


Am implementat 2 metode pentru transmiterea de date. Nu avem nevoie sa primim date.

Trimtierea unui caracter:



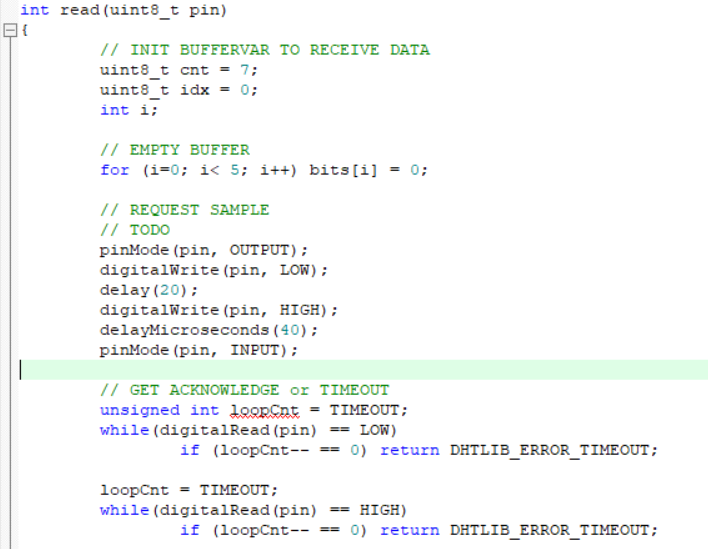
Trimiterea unui sir de caractere:



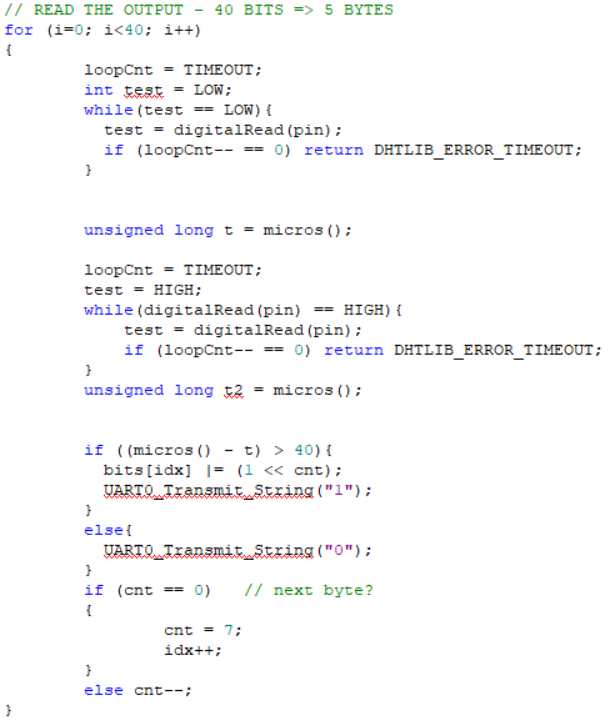
## Detaliere dht.h

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:

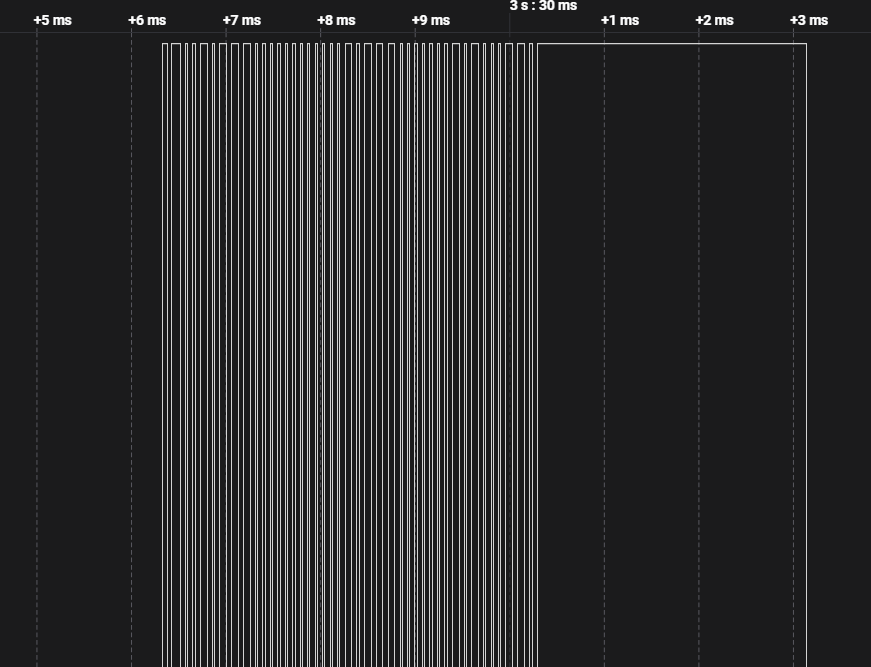


# Analiza rezultatelor

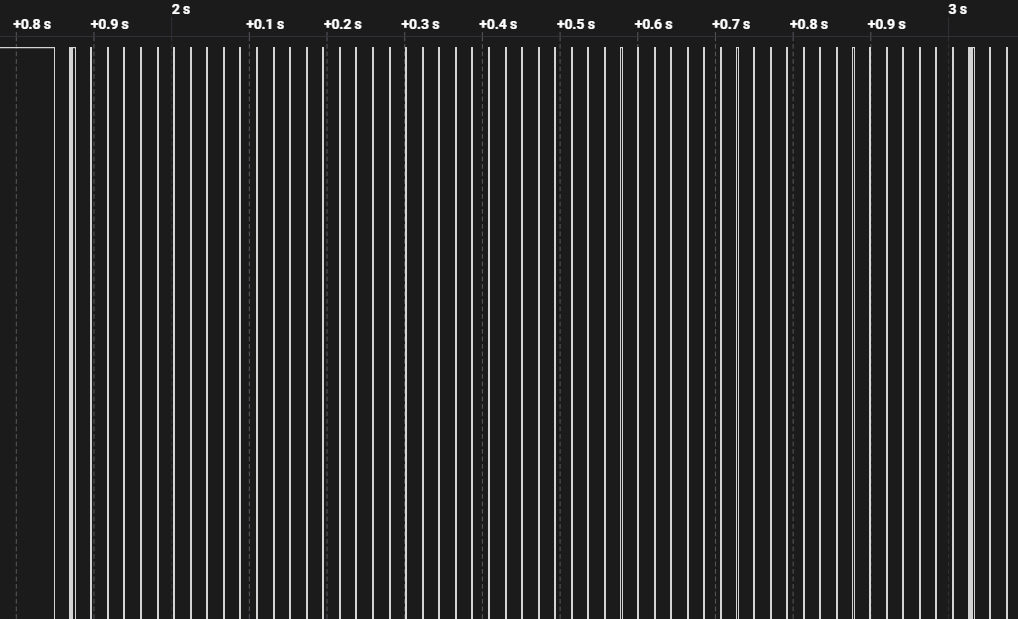
Pe langa folosirea comunicarii seriale pentru a capta datele, am folosit si un dispozitiv extern pentru a capta semnalul de date transmis.

## Rezultate implementare mbed

Transmiterea unui calup de date:

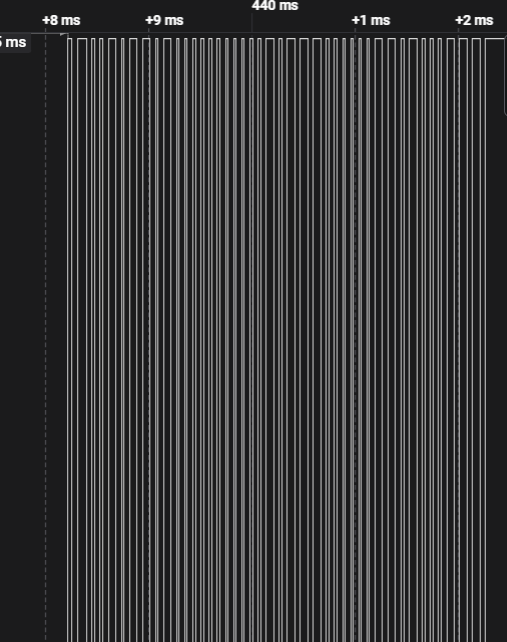


Secventa de mai sus se repeta la fiecare secunda:



## Rezultate implementare cmsis

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



## Rezultate python (implementare mbed)

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

