**ROMÂNIA**

**MINISTERUL APĂRĂRII NAȚIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ**

**Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională**



***Rotation Sensor + BCD***

***Platforma de dezvoltare FRDM-KL25Z***

Student Sg. Maj. Roiu Răzvan (C114A)

Student Sg. Maj. Bănica Liviu-Marian (C114A)

Student Sg. Maj. Popa Iulian-Vasile (C114E)

Cuprins

[**1**. Prezentarea senzorului de rotație analog V2 3](#_Toc123938734)

[2. Prezentarea senzorului BCD (DFR0089) 4](#_Toc123938735)

[3. Scopul proiectului 5](#_Toc123938736)

[4. Conectarea senzorilor la placa de dezvoltare FRDM-KL25Z 5](#_Toc123938737)

[4.1. Conectarea senzorului de rotație analog V2 5](#_Toc123938738)

[4.2. Conectarea modulului LED 6](#_Toc123938739)

[5. Descrierea programului 8](#_Toc123938740)

[5.1. Funcția main 8](#_Toc123938741)

[5.2. Modulul ADC 8](#_Toc123938742)

[5.2.1. Inițializarea modului ADC 9](#_Toc123938743)

[5.2.2. Funcția de calibrare 10](#_Toc123938744)

[5.2.3. ADC0\_Read 11](#_Toc123938745)

[5.2.4. Tratarea întreruperi în funcția ADC0\_IRQHandler 11](#_Toc123938746)

[5.3. Modulul GPIO 12](#_Toc123938747)

[5.3.1. Inițializarea ledurilor din componența modulului BCD 12](#_Toc123938748)

[5.4. Modulul UART 14](#_Toc123938749)

[5.4.1. Inițializare UART 15](#_Toc123938750)

[5.4.2. Transmiterea valoriilor folosind UART 16](#_Toc123938751)

[6. Generare grafic în python în timp real 16](#_Toc123938752)

[6.1. Librăriile necesare 16](#_Toc123938753)

[6.2. Variabile globale 16](#_Toc123938754)

[6.3. Preluarea datelor din portul serial 16](#_Toc123938755)

[6.4. Inițializare și afișare grafic 18](#_Toc123938756)

[6.5 Final 19](#_Toc123938757)

[7. Dificultăți întâmpinate 20](#_Toc123938758)

[8. Referințe 21](#_Toc123938759)

**1**. Prezentarea senzorului de rotație analog V2

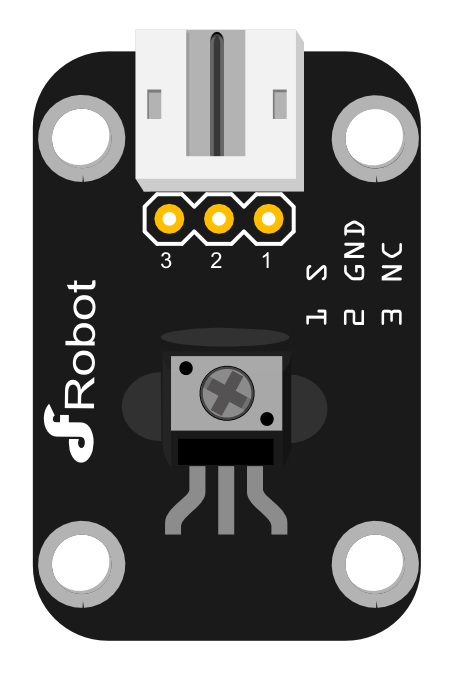
Senzorul de rotație analog V2 este un senzor analog care se poate roti până la 300 de grade având la bază un potențiometru.



*Figura 1. Senzorul de rotație analog V2*

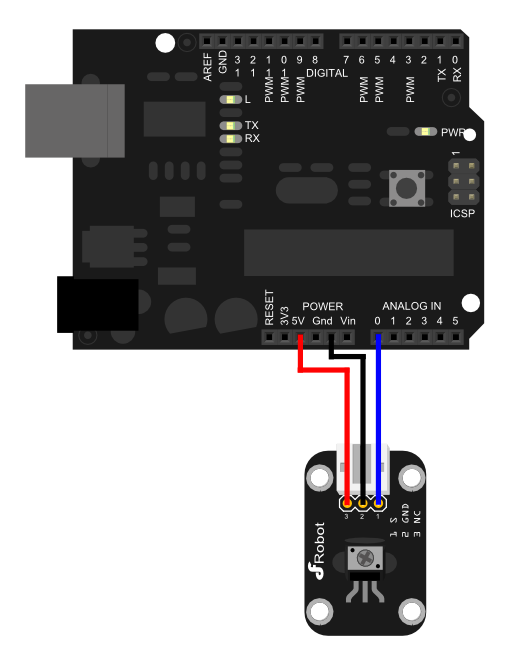
Pinii senzorului:

1. Signal output
2. GND
3. Power

**

*Figura . Pinii senzorului*

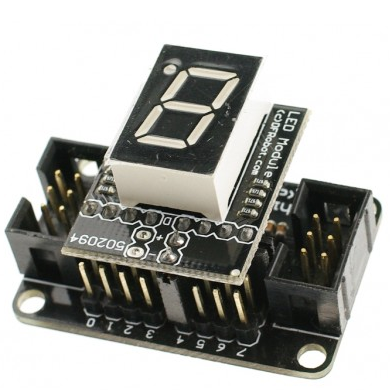
Exemplu de conectare la placa de dezvoltare:



*Figura 2. Exemplu de conectare senzor*

2. Prezentarea senzorului BCD (DFR0089)

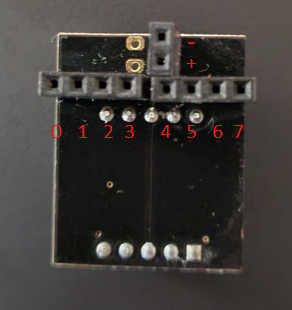
Senzorul BCD este defapt un modul de afișare LED foarte simplu și mic care are nevoie de 5V pentru alimentare.



*Figura 3. Senzorul BCD (DFR0089)*

Pinii modului LED:

1. Pinul pentru masă (-)
2. Pinul pentru tensiune (+)
3. Pinii 0-7 corespunzători fiecărui segment

**

*Figura 4. Pinii modului LED*

# 3. Scopul proiectului

Scopul proiectului este acela de a converti valoarea obținută prin acționarea senzorului de rotație într-o valoare afișabilă pe BCD (funcționalitate asemănătoare cu cea a unui buton de volum).

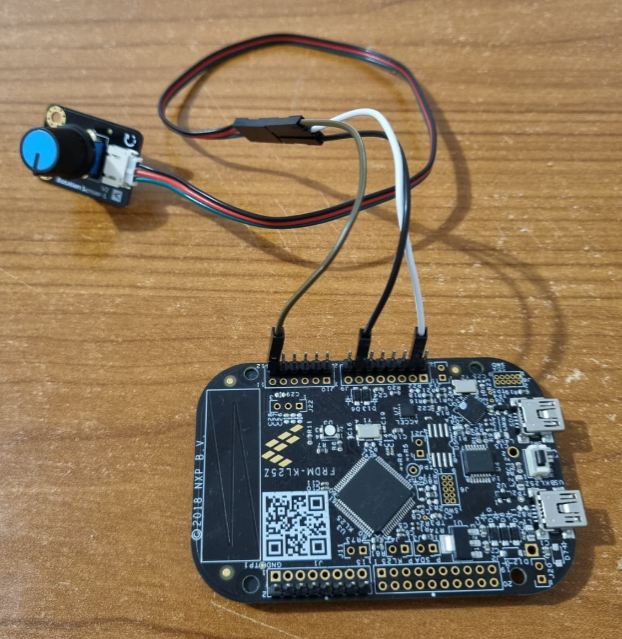
Proiectul dispune de un grafic în timp real cu valoarile preluate de la senzorul de rotație.

# 4. Conectarea senzorilor la placa de dezvoltare FRDM-KL25Z

Vom conecta senzorii pe rând:

4.1. Conectarea senzorului de rotație analog V2

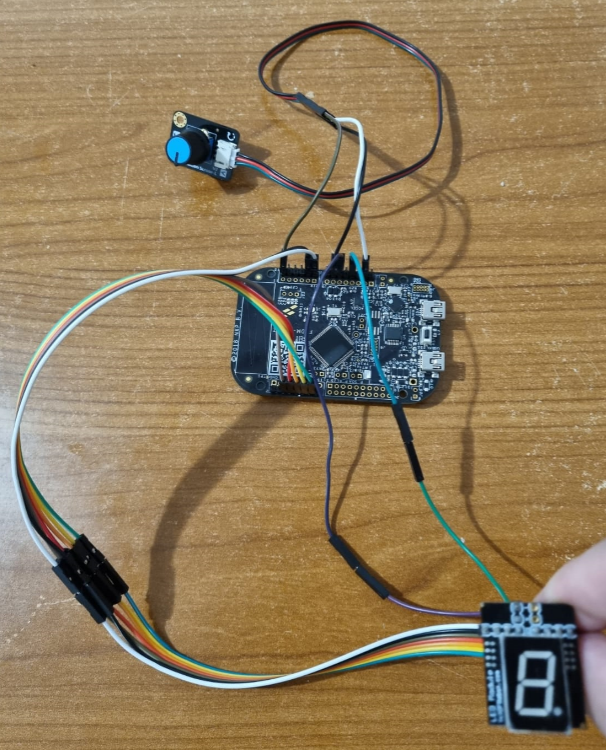
* Firul alb - roșu conectează senzorul la P3V3
* Firul maro – negru conectează senzorul la PTC1
* Firul negru – verde conectează senzorul la GND



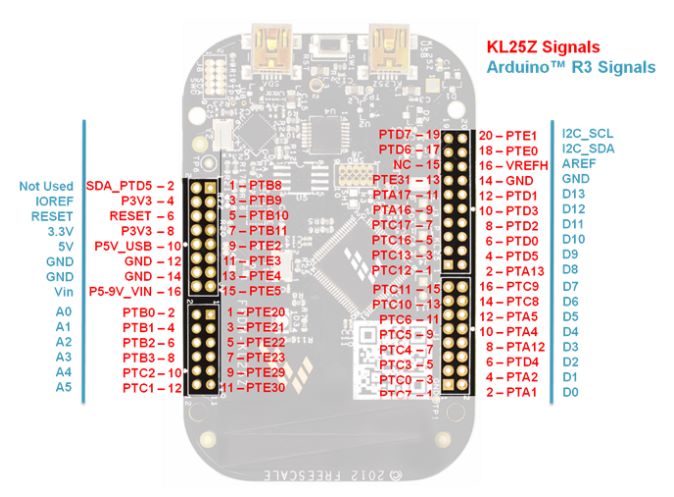
*Figura 5. Conectare sensor de rotație analog V2*

4.2. Conectarea modulului LED

* Firul verde – turcoaz conectează modulul la P3V3
* Firul mov – mov conectează modulul la GND
* Firul alb – alb conectează modulul la PTB0
* Firul negru – negru conectează modulul la PTB1
* Firul maro – maro conectează modulul la PTD4
* Firul roșu – roșu conectează modulul la PTA12
* Firul portocaliu – portocaliu conectează modulul la PTA4
* Firul galben – galben conectează modulul la PTA5
* Firul verde – verde conectează modulul la PTC8



*Figura 6. Conectare modul LED*



*Figura 7. Pini placă dezvoltare*

5. Descrierea programului

5.1. Funcția main

În fișierul main.c am inclus fișierele header în care sunt declarate funcții și variabile ce urmează a fi folosite: *adc.h* , *uart.h,gpio.h* Dupa care inițializez modulul UART și modulul ADC.

5.2. Modulul ADC

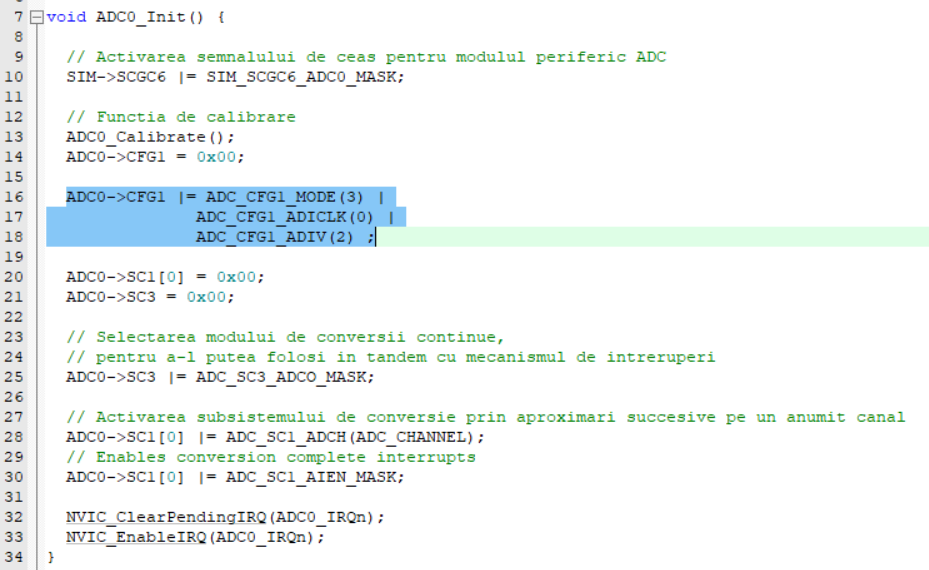
Vom folosi modulul ADC. Modulul ADC (Analog-to-Digital Converter) convertește semnale analogice, cum ar fi tensiunile, în semnale digitale pe care un computer le poate procesa. Acest lucru permite unui sistem să măsoare și să interpreteze cantități fizice, cum ar fi temperatura sau presiunea, și să le transforme într-o formă care poate fi manipulată de un computer sau alt dispozitiv digital. Modulul ADC poate fi folosit pentru a interfata senzori analogici sau alte dispozitive analogice cu un microcontroler sau alt sistem digital. Modulele ADC pot include caracteristici precum canale multiple, rate de eșantionare configurabile și amplificatoare de câștig programabile. Ele pot include, de asemenea, memorie la bord pentru stocarea datelor convertite și alte funcții, cum ar fi compensarea temperaturii sau filtrarea.

Vom folosii portul PTC1 pentur citirea semnalului analog astfel:

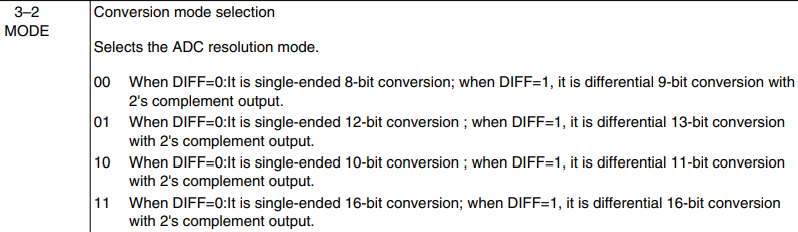




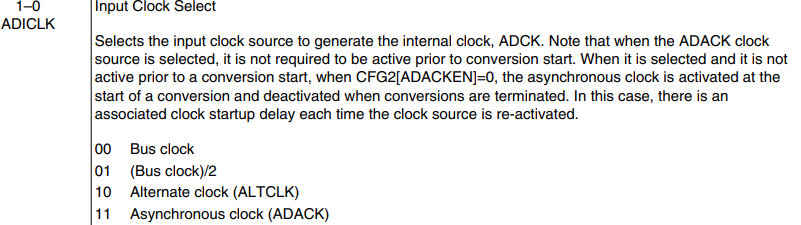
### 5.2.1. Inițializarea modului ADC



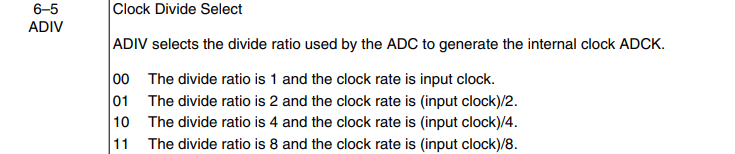
În registrul ADC\_CFG1 setăm campul MODE pentru a folosii modul de conversi p1 16 biti



Tot in același registru setam modul ADICLK pentru a genera un semnal de ceas intern



Iar modul ADIV pentru a selecta rate de divizare folosit de periferic pentru generarea ceasului intern



### 5.2.2. Funcția de calibrare

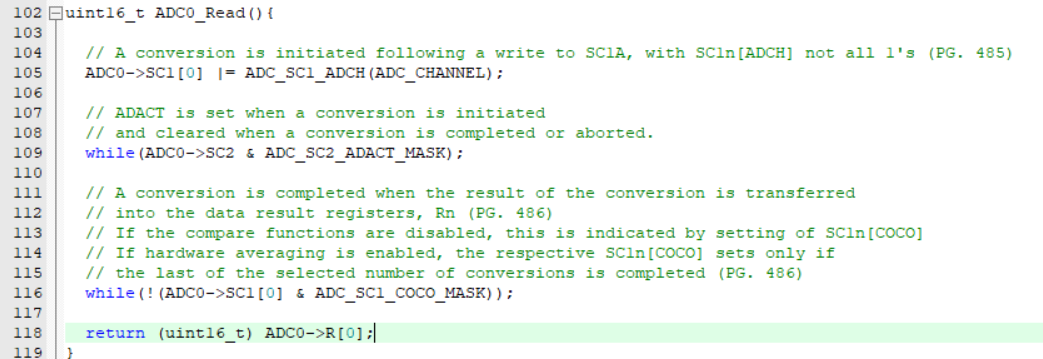
ADC\_Calibrate este o funcție care este utilizată pentru a calibra un ADC. Calibrarea unui ADC implică ajustarea parametrilor ADC-ului, cum ar fi gain-ul și offset-ul acestuia, pentru a se asigura că acesta convertește cu acuratețe semnalele analogice în formă digitală.

Există mai multe motive pentru care se dorește calibrarea unui ADC:

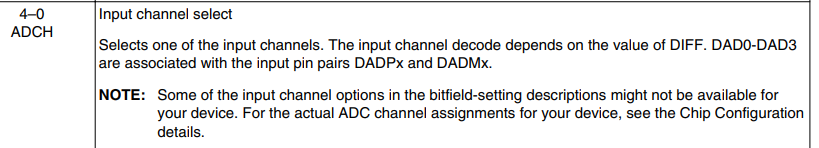
* Pentru a compensa modificările în performanța ADC-ului în timp.
* Pentru a îmbunătăți acuratețea ADC-ului.
* Pentru a corecta erorile care pot fi introduse de factori externi, cum ar fi fluctuațiile de temperatură sau tensiune.

Funcția ADC\_Calibrate este de furnizată ca parte a bibliotecii și este utilizată pentru a efectua calculele și ajustările necesare pentru a calibra ADC-ul.

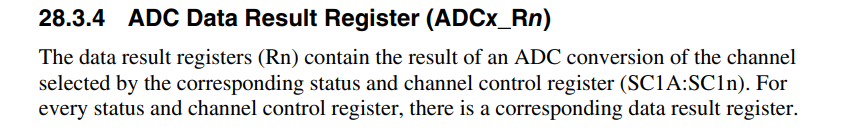
### 5.2.3. ADC0\_Read



În registrul ADC\_SC1 (Status and Control Register) vom seta modul ADCH pentu a selecta canalul de citire:

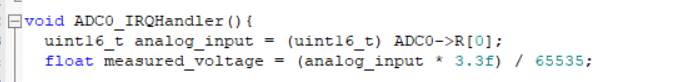


Iar in registrul ADC\_R[0] contine rezultatul conversiei ADC:



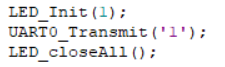
### 5.2.4. Tratarea întreruperi în funcția ADC0\_IRQHandler

Acestă funcție o folosim odată cu apriția unei intreruperi, pentru tratarea acesteia și respectiv citirea valorilor din ’Data Register’.



Odată aflată această valoare o putem convertii și folosii modulul GPIO pentru afișarea acesteia pe BCD.

Exemplu pentru afisarea valorii “1”:

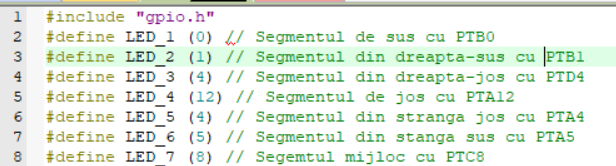


Aici intervine modulul GPIO după cum urmează in capitolul următor.

## 5.3. Modulul GPIO

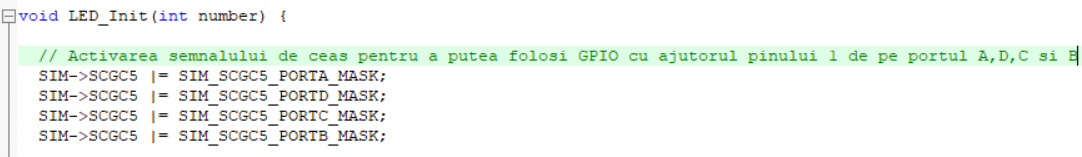
GPIO vine de la “General Purpose Input-Output”. Se referă la o interfață programabilă pe un dispozitiv care permite să transmitem și să primim date cu alte dispozitive sau componente. De exemplu,putem folosi pinii GPIO pentru a citi date de la senzori, a controla motoarele sau chiar a comunica cu alte dispozitive prin internet. In proiectul nostru am folosit Modulul GPIO pentru a comunica cu Senzorul BCD.

Declararea pinilor care vor fi folositi pentru comunicarea BCD:

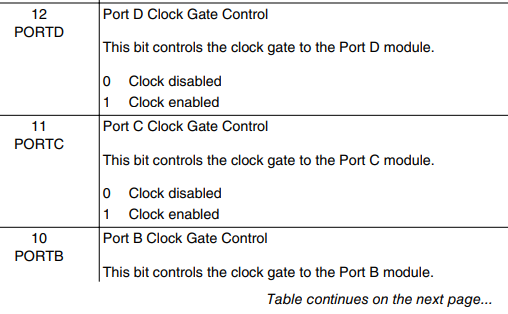


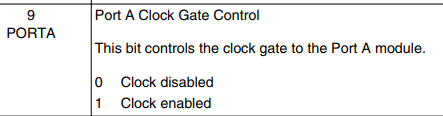
### 5.3.1. Inițializarea ledurilor din componența modulului BCD

Folosim registrul SCGC5 din cadrul SIM (System Integration Module) pentru a active semnalele de ceas pentru a putea folosii pinii declarati anterior.

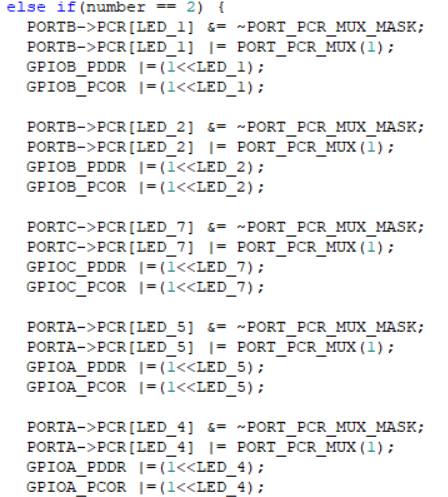


Iar porturile folosite sunt porturile A,D,C și B





Cele 7 segmente din componența Modulului BCD sunt aprinse pentru a putea afișa valorile citite de noi utilizând Senzorul ADC. Exemplu pentru afișarea numărului “2”.



Setăm registrului PORTB->PCR care ia ca parametru pinul de utilizat. Și registrul GPIOB->PCOR pentru a aprinde led-ul respectiv.

Iar pentru stingerea ledurilor respective avem o functie care se ocupă de acest aspect. Utilizăm registrul PSOR din componența modulului GPIO pentru a pentru a stinge ledul.

Exemplu pentru a stinge ledul 3 din compnența BCD-ului:

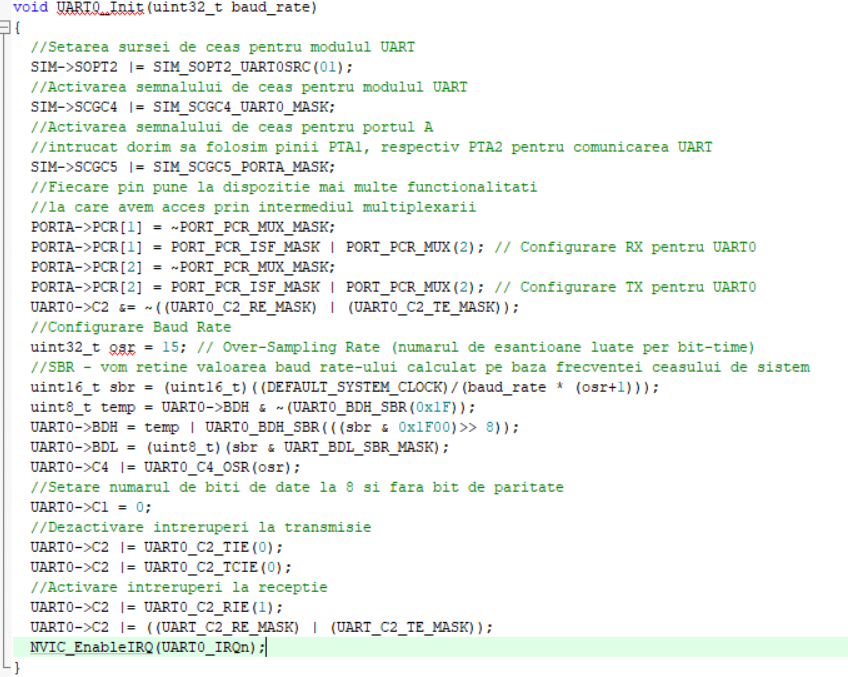


## 5.4. Modulul UART

UART înseamnă Universal Asynchronous Receiver/Transmitter. Este un tip de protocol de comunicație serială care permite dispozitivelor să trimită și să primească date printr-o serie de fire sau cabluri. Protocolul UART este utilizat pe scară largă în computere, periferice și microcontrolere pentru a transmite și a primi date. UART-urile sunt utilizate în mod obișnuit pentru a conecta dispozitive precum imprimante, scanere și modemuri la un computer sau pentru a conecta dispozitive între ele, cum ar fi un microcontroler la un computer sau un modem satelit la un receptor de satelit. Protocolul UART este relativ simplu și ușor de utilizat, ceea ce îl face o alegere populară pentru multe aplicații care necesită comunicare în serie.

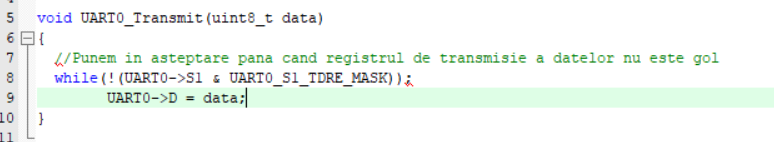
### 5.4.1. Inițializare UART

Utilizăm funcția UART\_Init pentru a inițializa modulul UART si pentru a specifica baud rate-ul dorit comunicării.



### 5.4.2. Transmiterea valoriilor folosind UART

Aceste valorii vor fi mai apoi folosite pentru a genera graficul.



6. Generare grafic în python în timp real

6.1. Librăriile necesare

Vom începe prin a importa bibliotecile necesare:

* **serial**, care este folosit pentru a comunica cu portul serial
* **numpy**, care este folosit pentru a lucra cu array-uri de date
* **tkinter**, care este folosit pentru a crea interfața grafică (GUI)
* **FigureCanvasTkAgg** - importă clasa FigureCanvasTkAgg din biblioteca **matplotlib**, care este utilizată pentru a afișa un grafic într-o fereastră tkinter
* **Figure** - importă clasa Figure din biblioteca **matplotlib**, care este utilizată pentru a crea un grafic

6.2. Variabile globale

Pentru a stoca datele primite de la portul serial vom definii un array gol, denumit **data**.

6.3. Preluarea datelor din portul serial

Definim funcția **plot\_data** care va fi folosită pentru a actualiza graficul cu noi date. Aceasta începe prin a citii o linie de date de la portul serial, de a o converti la un șir de caractere și de a o curăța de orice caractere goale sau spații.

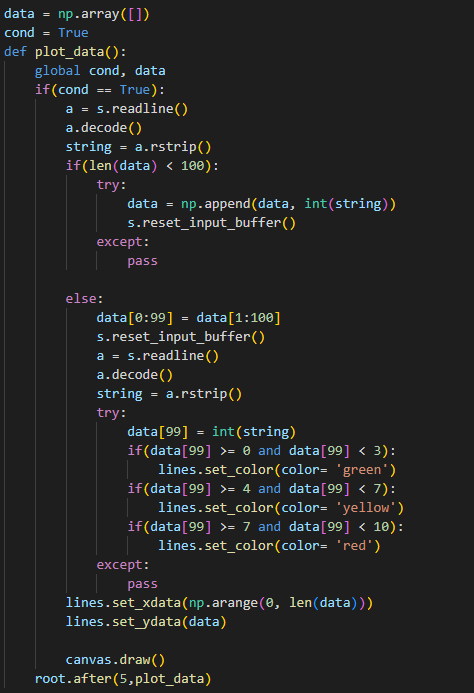
Apoi verificăm lungimea array-ului **data** și, dacă este mai mică decât 100, adaugăm valoarea în array. Dacă array-ul **data** este deja la lungimea sa maximă de 100, funcția shiftează la stânga toate punctele din array cu o poziție și adaugă noua valoare la sfârșitul array-ului.Dacă valoarea noului punct de date este între anumite limite, schimbă culoarea liniei trasate în grafic:

* Dacă valoarea primită din portul serial se află între 0 și 3,inclusiv 0, atunci culoarea liniei trasate va fi **verde**
* Dacă valoarea primită din portul serial se află între 3 și 7, atunci culoarea liniei trasate va fi **galben**
* Dacă valoarea primită din portul serial se află între 7 și 10, inclusiv 7, atunci culoarea liniei trasate va fi **roșu**

Deoarece frecvența datelor venite din portul serial este foarte mare, vom reseta buffer-ul de intrare de la portul serial reprezentat de **s** după fiecare citire.Ne vom folosi de funcția **reset\_input\_buffer** al api-ului oferit de biblioteca serial.De asemenea, vom încerca sa convertim șirul de caractere(care ar trebuii să fie defapt un caracter, o cifră) într-un întreg folosind funcția de conversie a unui string la int **int** și adaugăm rezultatul la array-ul **data**. Vom încapsula această “încercare” într-un bloc **try catch**.În cazul unei excepții vom trece mai departe.

În final, actualizăm datele care sunt trasate pe grafic și redesenăm canvasul,cu ajutorul funcției **draw** oferită de o instanța a clasei FigureCanvasTkAgg declarată ulterior în “main”-ul aplicației.Pentru a cauza efectul de interpretare a datelor in timp real ne vom folosi de funcția **after** pusă la dispoziție de api-ul bibliotecii **tkinter** și vom seta ca funcția **plot\_data** să se reapeleze după un interval de 5 milisecunde.

În captura de ecran de mai jos este ilustrat rezultatul celor de mai sus

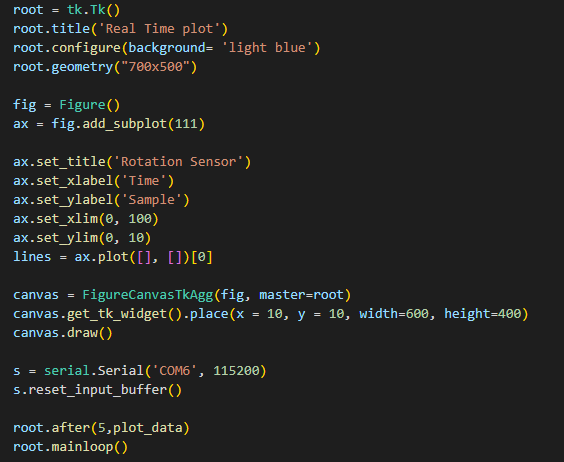


6.4. Inițializare și afișare grafic

Vom creea o fereastră grafică cu ajutorul tkinter.Ne vom folosi de funcția **Tk** care returnează un obiect de Tk și are mai multe metode, câteva exemple : **title**, **configure**, **geometry**.

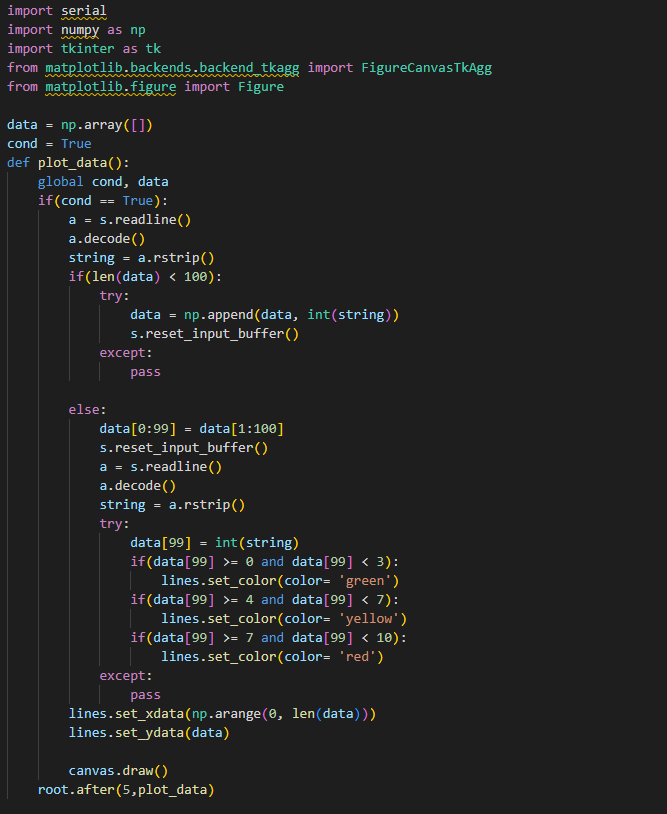
Apoi, se creează o figură și un ax (un subplot) pentru aceasta. Se setează titlul, etichetele pentru axa X și Y și limitele pentru acestea. Apoi, se creează o linie de plotare vidă și se adaugă la ax.

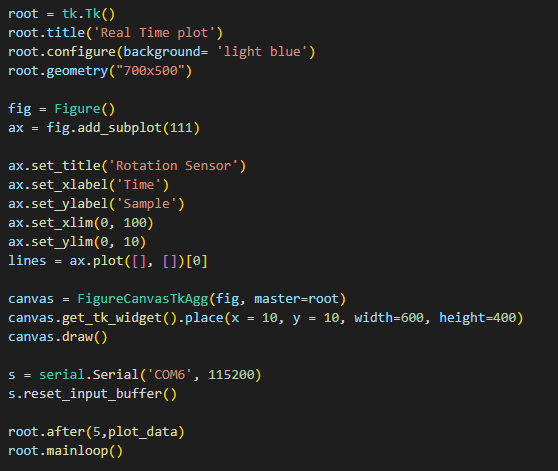
Se creează un canvas pentru a afișa figura și se plasează în fereastră. Se deschide apoi un port serial și se șterge buffer-ul de intrare al acestuia. În final, se face un apel la funcția **plot\_data** după 5 milisecunde și se lansează bucla principală a ferestrei.



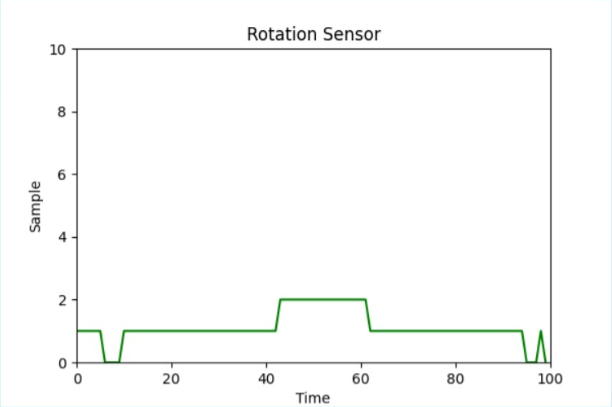
6.5 Final

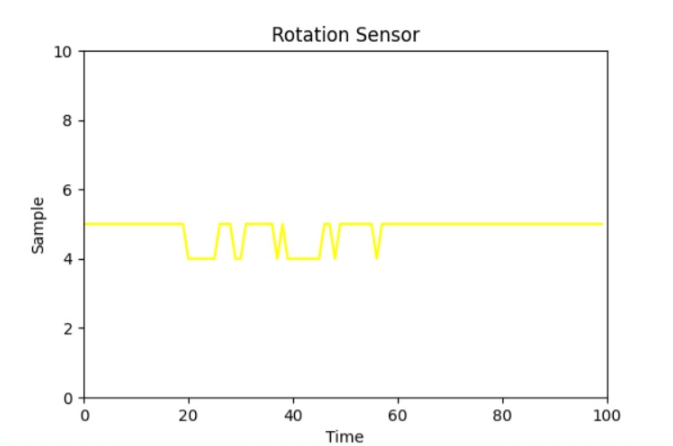
Codul rezultat este:

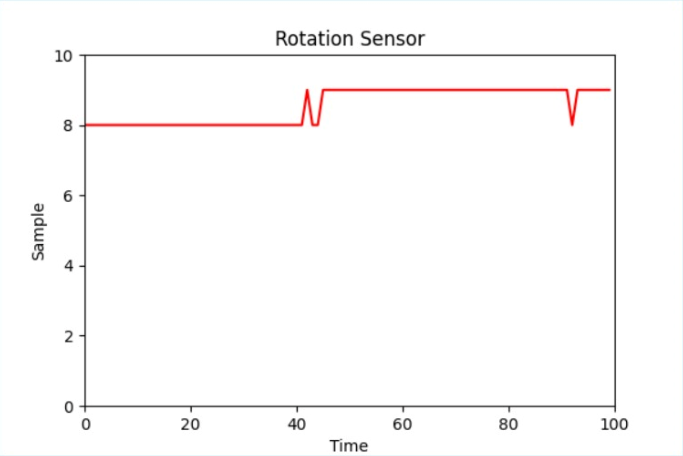




După rulare:







7. Dificultăți întâmpinate

În faza incipientă a proiectului am conectat senzorul de rotație si modulul LED la target, excluzând modulul UART.Am încercat sa conectăm modulul UART la target, acesta folosind porturile PTA1 si PTA2, porturi folosite deja de modulul LED.

Pentru a rezolva această problemă a trebuit să schimbăm porturile folosite pentru primele două LED-uri ale modului LED cu porturile PTB0 și PTB1.

# 8. Referințe

* FRDM-KL25Z\_Pinouts.pdf
* FRDM-KL25Z\_ReferenceManual.pdf
* FRDM-KL25Z\_UserManual.pdf
* <https://wiki.mta.ro/c/4/ssmp/start>
* <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0089_Web.pdf>
* <https://wiki.dfrobot.com/Analog_Rotation_Sensor_V1__SKU__DFR0054_>
* <https://github.com/WaveShapePlay/Arduino_RealTimePlot/blob/master/Part2_RealTimePlot_UsingClass/ArduinoRealTimePlot.py>
* <https://www.youtube.com/watch?v=z14l33paZGU>