**ROMÂNIA**

**MINISTERUL APĂRĂRII NAȚIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**Facultatea de Sisteme Informatice și Securitate Cibernetică**

**Departamentul de Calculatoare și Securitate Cibernetică**



***Utilizare senzor Analog de flacără***

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

Std. sg. maj. Tudor MIRON-ONCIUL

Std. sg. maj. Toraș-Mihnea JIPIANU

Grupa C114A și Grupa C114B

**București**

**2024**

Curpins

[**1.** **Prezentarea Senzorului DFR0076** 3](#_Toc155617893)

[**1.1** **Specificații** 3](#_Toc155617894)

[**1.2** **Aplicații** 3](#_Toc155617895)

[**1.3** **Conectare** 4](#_Toc155617896)

[**2.** **Scop Proiect** 4](#_Toc155617897)

[**3.** **Conectare Senzor – Placă de Dezvoltare** 5](#_Toc155617898)

[**4.** **Descriere Program** 6](#_Toc155617899)

[**4.1** **Fișierul main.c** 6](#_Toc155617900)

[**4.2** **UART** 8](#_Toc155617901)

[**4.2.1** **Inițializare** 8](#_Toc155617902)

# **Prezentarea Senzorului DFR0076**

Senzorul de flacără poate fi folosit pentru a detecta focul sau alte lungimi de undă cuprinse între 760 nm și 1100 nm.



*Figura 1. Senzorul DFR0076*

Unghiul de detectare al senzorului de flacără este de 60 de grade, având o sensibilitate specială la spectrul flăcării. Dispune de două găuri de montare M3 pentru a stabiliza modulul și a preveni rotirea acestuia.

Temperatura de funcționare a senzorului de flacără este cuprinsă între -25 de grade Celsius și 85 de grade Celsius. În timpul detecției flăcării, este important să se mențină o distanță corespunzătoare pentru a evita deteriorarea senzorului.

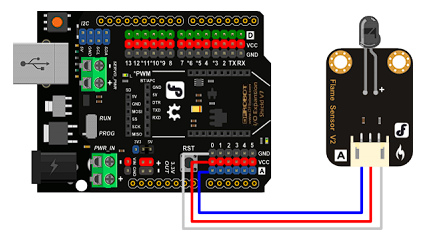
## **Specificații**

* Tensiune de alimentare: 3,3V până la 5V
* Distanța de detectare: 20cm (4,8V) până la 100cm (1V)
* Lungimi de undă: 760nm to 1100nm
* Timp de răspuns: 15 microsecunde
* Interfață: Analogică
* Dimensiuni: 22x30mm

## **Aplicații**

* Alarmă incendiu
* Robot specializat în detectarea focului

## **Conectare**



*Figura 2. Exemplu de conectare senzor*

Definire cabluri:

* Cablu albastru – analog
* Cablu Roșu – VCC
* Cablu Gri – Ground

# **Scop Proiect**

Scopul proiectului constă în iluminarea a 3 leduri dispuse pe o placă experimentală (breadboard), în funcție de valorile furnizate de senzorul de flacără. Valorile detectate de senzor vor fi categorisite în 3 intervale distincte. În funcție de intervalul în care se încadrează valoarea detectată de senzor, se vor activa 1, 2 sau 3 leduri.

De asemenea, prin intermediul interfeței UART, valorile detectate de senzorul de flacără vor fi transmise către o aplicație dezvoltată în Python. Această aplicație are rolul de a citi o dată la 1 secundă valoarea transmisă de senzor și de a afișa un grafic care reflectă aceste valori. Graficul va prezenta ultimele 25 de valori, actualizându-se la fiecare jumatate secundă cu noua valoare recepționată. Fiecare bară din barchart va fi colorată în funcție de intervalul în care se încadrează valoarea respectivă, respectiv:

* 0% - 33% - verde
* 33% - 66% - galben
* 66% - 100% - roșu

Pe placa de dezvoltare, ledurile vor fi activate urmărind secvența: alb, verde, galben, stins. Interfața dezvoltată în Python va conține un buton care va permite inversarea acestei secvențe.

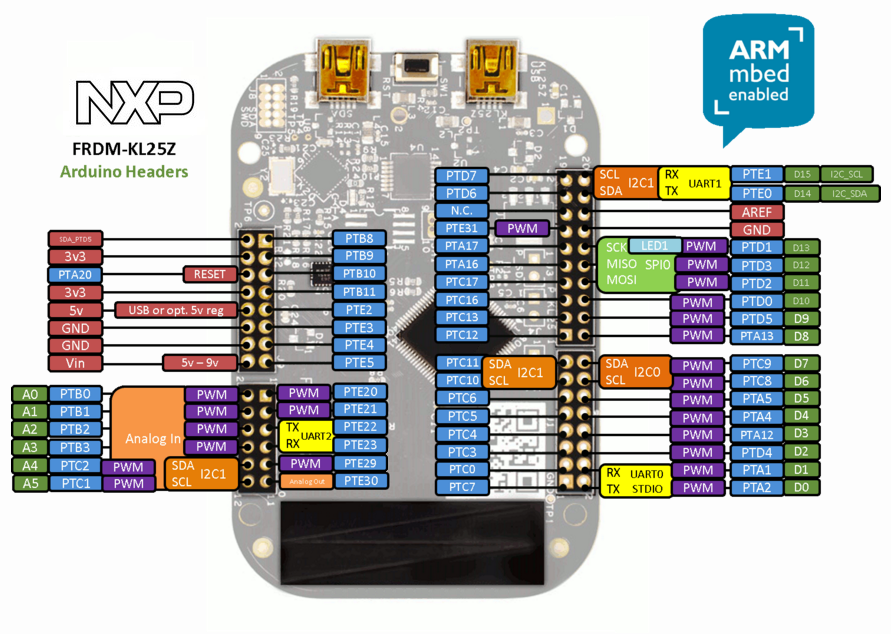
# **Conectare Senzor – Placă de Dezvoltare**

Senzorul este conectat la placuță astfel:

* Firul albastru – gri conectează senzorul la PTC2
* Firul roșu – roșu conectează senzorul la 5V
* Firul negru – negru conectează senzorul la GND

\*POZA CU SENZORUL CONECTAT LA PLACUTA\*

*Figura 3. Conectare senzor*



*Figura 4. Pini placă de dezvoltare[[1]](#footnote-1)*

# **Descriere Program**

## **Fișierul main.c**

1. #include "Gpio.h"

2. #include "Uart.h"

3. #include "Adc.h"

4. #include "Pit.h"

5.

6. uint8\_t flag = 0;

7.

8. void changeLedSeq(void)

9. {

10. if(UART0->S1 & UART0\_S1\_RDRF\_MASK)

11. {

12. char c = UART0->D;

13. if(flag == 0)

14. flag = 1;

15. else

16. flag = 0;

17. ledDirection = flag;

18. }

19. else

20. return;

21. }

22.

23. int main(void)

24. {

25. float measured\_voltage;

26. uint8\_t parte\_zecimala;

27. uint8\_t parte\_fractionara1;

28.

29. UART0\_Init(38400);

30. RGBLed\_Init();

31. ADC0\_Init();

32. PIT\_Init();

33.

34. for(;;)

35. {

36. changeLedSeq();

37.

38. measured\_voltage = (analog\_input \* 6.0f) / 65535;

39.

40. parte\_zecimala = (uint8\_t) measured\_voltage;

41. parte\_fractionara1 = ((uint8\_t)(measured\_voltage \* 10)) % 10;

42.

43. ChangeColorFromFlame(parte\_zecimala+(parte\_fractionara1)/10);

44.

45. UART0\_Transmit('V');

46. UART0\_Transmit('a');

47. UART0\_Transmit('l');

48. UART0\_Transmit('u');

49. UART0\_Transmit('e');

50. UART0\_Transmit(' ');

51. UART0\_Transmit('=');

52. UART0\_Transmit(' ');

53. UART0\_Transmit(parte\_zecimala + 0x30);

54. UART0\_Transmit('.');

55. UART0\_Transmit(parte\_fractionara1 + 0x30);

56. UART0\_Transmit('V');

57. UART0\_Transmit(0x0A);

58. UART0\_Transmit(0x0D);

59.

60. readyToTransmit = 0;

61. }

62. }

63.

În fișierul „*main.c*” sunt incluse fișierele header în care sunt declarate funcțiile și variabilele folosite:

* „Gpio.h” conține funcțiile și variabilele pentru interacțiunea cu ledurile (atât de pe placă cât și de pe breadboard)
* „Uart.h” conține funcțiile pentru a interacșiona cu perifericul UART care este responsabil pentru trimiterea și primirea datelor din aplicația Python.
* „Adc.h” conține funcțiile și variabilele responsabile pentru interacțiunea cu senzorul de flacără
* „Pit.h” conține funcțiile pentru a interacționa cu cronometrul responsabil pentru sincronizarea ledurilor de pe plăcuță

Fluxul programului este următorul:

* În prima parte a programului sunt apelate funcțiile de inițializare ale prifericelor:

* + UART0\_Init(38400) – inițializare periferic UART (pentru transmitere și recepție date de la aplicația Python)
  + RGBLed\_Init() – inițializare perriferic GPIO (pentru becurile de pe plăcuță și breadboard)
  + ADC0\_Init() – inițializare convertor analogic-digital pentru citirea valorilor recepționate de senzorul de flacără
  + PIT\_Init() – inițializare periferic PIT, pentru setarea cronometrelului
* În a doua parte se inițializeaza o buclă infinită în care se vor aștepta întreruperi de la periferice:
  + Funcția *changeLedSeq()* verifică dacă perifericul UART a recepționat date de la aplicația Python pentru a schimba sensul secvenței de aprindere a becurilor LED de pe plăcuță
  + În continuare se preia valoarea pe 16 biți recepționată de senzor și se scaleaza la o valoare *float* cuprinsă între 0 si 6. Acesta este intervalul în care vom include valorile recepționate de senzor.
  + Funcția *ChangeColorFromFlame(parte\_zecimala+(parte\_fractionara1)/10)* este responsabilă pentru aprinderea becurilor de pe breadboard. Parametrul este valoarea în *float.*
  + În continuare se trimite, prin intermediul perifericului UART, valoarea măsurată însoțită de cuvântul *Value.*

## **UART**

### **Inițializare**

1. void UART0\_Init(uint32\_t baud\_rate)

2. {

3. uint32\_t osr;

4. uint16\_t sbr;

5. uint8\_t temp;

6. /\* Setarea sursei de ceas pentru modulul UART\*/

7. SIM->SOPT2 |= SIM\_SOPT2\_UART0SRC(01);

8.

9. /\* Activarea semnalului de ceas pentru modulul UART\*/

10. SIM->SCGC4 |= SIM\_SCGC4\_UART0\_MASK;

11.

12. /\* Activarea semnalului de ceas pentru portul A\*/

13. /\* intrucat dorim sa folosim pinii PTA1, respectiv PTA2 pentru comunicarea UART\*/

14. SIM->SCGC5 |= SIM\_SCGC5\_PORTA\_MASK;

15.

16. /\* Fiecare pin pune la dispozitie mai multe functionalitati \*/

17. /\* la care avem acces prin intermediul multiplexarii\*/

18. PORTA->PCR[1] = ~PORT\_PCR\_MUX\_MASK;

19. PORTA->PCR[1] = PORT\_PCR\_ISF\_MASK | PORT\_PCR\_MUX(2); /\* Configurare RX pentru UART0\*/

20. PORTA->PCR[2] = ~PORT\_PCR\_MUX\_MASK;

21. PORTA->PCR[2] = PORT\_PCR\_ISF\_MASK | PORT\_PCR\_MUX(2); /\* Configurare TX pentru UART0\*/

22.

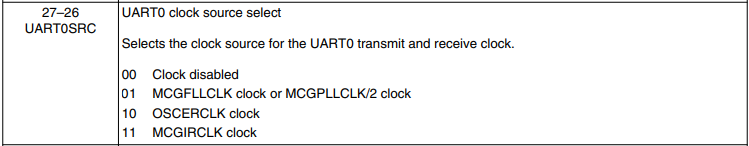
23.

24.

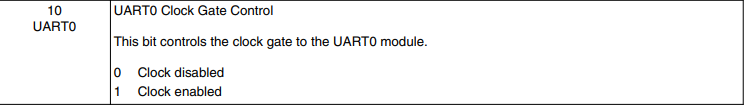
25. UART0->C2 &= ~((UART0\_C2\_RE\_MASK) | (UART0\_C2\_TE\_MASK));

26.

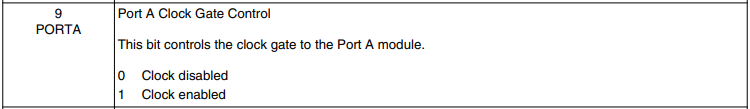
În registrul SIM\_SOPT2 (System Options Register 2) setăm pe 0b01 câmpul UART0SRC (biții 27-26) pentru selectarea ca ceas al modulului MCGFLLCLK anterior configurat.



În registrul SIM\_SCGC4 (System Clock Gating Control Register 4) setăm pe 1 câmpul UART0 (bitul 10) pentru activarea ceasului pentru acest modul.



În registrul SIM\_SCGC5 (System Clock Gating Control Register 5) setăm pe 1 câmpul PORTA (bitul 9) pentru activarea ceasului acestui port.



În regiștrii de control ai pinilor 1 și 2 din portul A (PORTA\_PCR1/2), setăm câmpul MUX (biții 10-8) pe valoarea 0b010, care înseamnă folosirea acestora în modulul de UART0 (RX/TX).



1. /\* Configurare Baud Rate\*/

2. osr = 15; /\* Over-Sampling Rate (numarul de esantioane luate per bit-time)\*/

3.

4. /\* SBR - vom retine valoarea baud rate-ului calculat pe baza frecventei ceasului de sistem\*/

5. /\* SBR - b16 b15 b14 [b13 b12 b11 b10 b09 b08 b07 b06 b05 b04 b03 b02 b01] &\*/

6. /\* 0x1F00 - 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0\*/

7. /\* 0 0 0 b13 b12 b11 b10 b09 0 0 0 0 0 0 0 0 >> 8\*/

8. /\* BDH - 0 0 0 b13 b12 b11 b10 b09\*/

9. /\* BDL - b08 b07 b06 b05 b04 b03 b02 b01\*/

10. sbr = (uint16\_t)((DEFAULT\_SYSTEM\_CLOCK)/(baud\_rate \* (osr+1)));

11. temp = UART0->BDH & ~(UART0\_BDH\_SBR(0x1F));

12. UART0->BDH = temp | UART0\_BDH\_SBR(((sbr & 0x1F00)>> 8));

13. UART0->BDL = (uint8\_t)(sbr & UART\_BDL\_SBR\_MASK);

14. UART0->C4 |= UART0\_C4\_OSR(osr);

15.

16.

17. /\* Setare numarul de biti de date la 8 si fara bit de paritate\*/

18. UART0->C1 = 0;

19.

20. /\* Dezactivare intreruperi la transmisie\*/

21. UART0->C2 |= UART0\_C2\_TIE(0);

22. UART0->C2 |= UART0\_C2\_TCIE(0);

23.

24. /\* Activare intreruperi la receptie\*/

25. UART0->C2 |= UART0\_C2\_RIE(1);

26.

27. UART0->C2 |= ((UART\_C2\_RE\_MASK) | (UART\_C2\_TE\_MASK));

28.

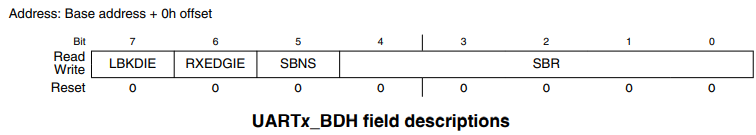
29.

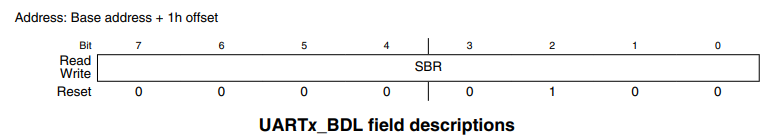
30.

31. }

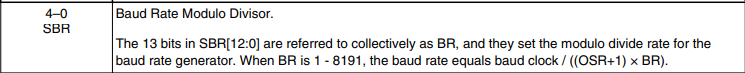
32.

Configuram baud rate în funcție de osr=15.

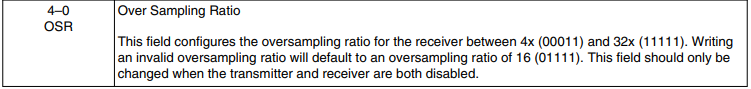




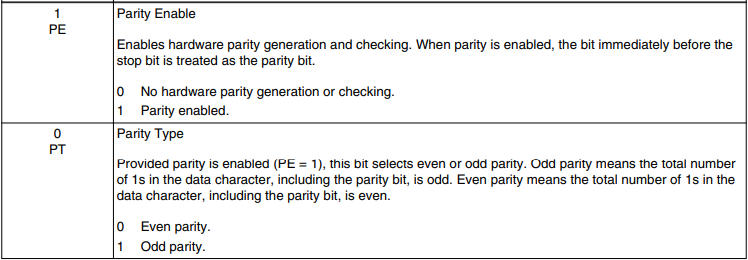
Formula după care se calculează baud rate este: DEFAULT\_SYSTEM\_CLOCK / ((OSR+1) × BR) (pentru SBR între 1 – 8191).

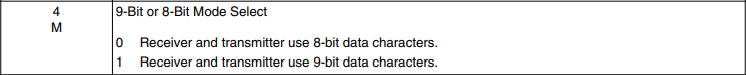


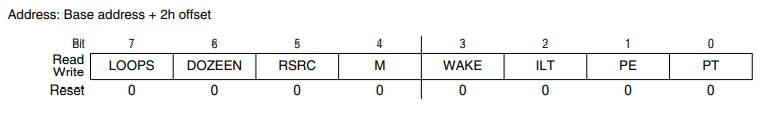
OSR (Over Sampling Ratio) are valoarea implicită 0b01111 adică 15, iar baud clock a fost setat la cel de 48 MHz.



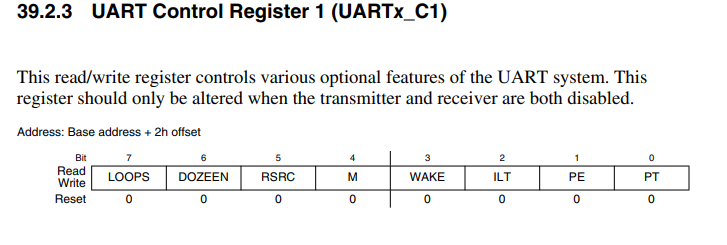
Portul serial este setat implicit cu 8 biți date, niciun bit de paritate și un bit de stop.



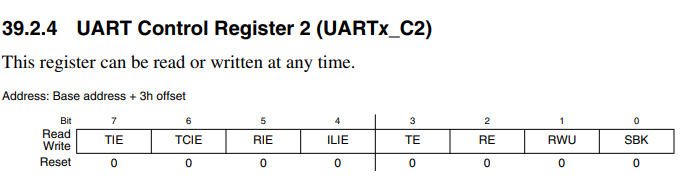


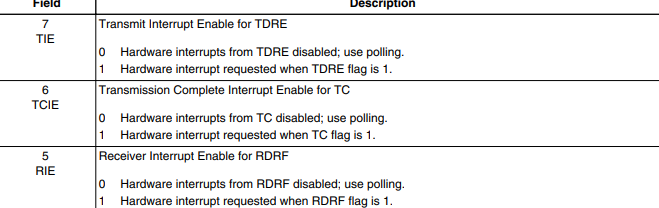


Registrul UARTx\_C1 ( UART Control Register 1 ) se setează pe 0.



Registrul UARTx\_C2





1. https://os.mbed.com/platforms/KL25Z/ [↑](#footnote-ref-1)