ROMÂNIA
MINISTERUL APĂRĂRII NAȚIONALE
ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”

FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ
Specializare: Calculatoare și Sisteme Informatice pentru Apărare și Securitate Națională

SENZOR DE FOC

Echipa: Gheorghe Alina Andreea
Bălășcan Gabriel-Danut
Grupa: C 114B

BUCUREȘTI

Cuprins :

Scopul proiectului

Prezenta lucrare are ca scop dezvoltarea unei aplicații care să detecteze și să răspundă la apariția unor flăcări prezente în proximitatea unui senzor prin afișarea unor grafice în timp real sugestive cu privire la acest aspect. Aplicația își propune preluarea datelor prin intermediul perifericului ADC(analog discovery) de la un senzor de flacără și afișarea valorilor obținute prin transmisia pe interfața UART către PC într-un grafic.

Componente software utilizate

Medii de dezvoltare integrate (IDE):

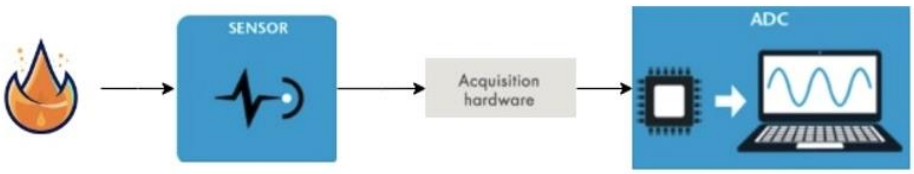
- Arm Keil MDK
- Matlab versiunea 2020b

Preluarea datelor de la senzor se va realiza pe mediul de dezvoltare Arm Keil MDK, iar afișarea rezultatelor obținute sub formă grafică se va realiza prin intermediul aplicației Matlab.

Context

Utilizarea unui senzor de flacără poate fi integrată în realizarea unui sistem de alarmă ce detectează și anunță, prin integrarea cu un sistem de sonorizare, apariția unui incendiu. Sistemele de alarmă de incendiu sunt foarte frecvente în clădirile comerciale și în fabrici, aceste dispozitive conțin de obicei un grup de senzori care monitorizează în mod constant orice flacără, gaz sau incendiu în clădire și declanșează o alarmă dacă detectează pe oricare dintre aceștia.

Diagrama sistemului propus

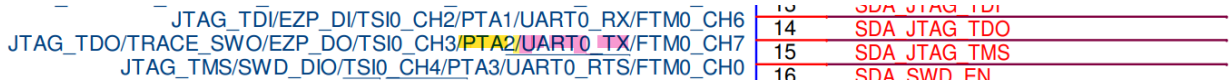


Periferice și regiștrii

În cadrul proiectului au fost necesare folosirea perfiericelor ADC și UART, pe care le-am detaliat în următoarele subcapitole. În continuarea prezentării celor două periferice, am ilustrat regiștrii folosiți și valorile cu care i-am setat.

Perifericul UART

Pentru transmiterea datelor către PC de la convertorul AD, am folosit comunicația UART. Când UART detectează un bit de start, începe să citească biții de intrare la o frecvență specifică cunoscută sub numele de viteză de transmisie sau bauderate. În cadrul acestui proiect am folosit o viteză de transmisie de 57600 biți/secundă. Pentru transmisia UART (setarea pinului UART0_TX) am activat portul A asociat pinului PTA2



Figură 1 Vizualizarea pinului de transmisie UART0_TX în Schematics [1]

În Figura 1 se poate vizualiza o captură din Schematics a amplasării pinului de transmisie și faptul că portul de transmisie e asociat alternativei PTA2.

Perifericul ADC

Modulul ADC suportă până la 24 de intrări single-ended. În cadrul acestui proiect am utilizat o intrare single-ended, deoarece informația e reprezentată sub forma unei diferențe între semnal și GND. Pinul analog folosit pentru conversia single-ended este A0 conectat pe canalul PTB0/ADC0_SE8 (AD8) al ADC**,** conform Schematics.

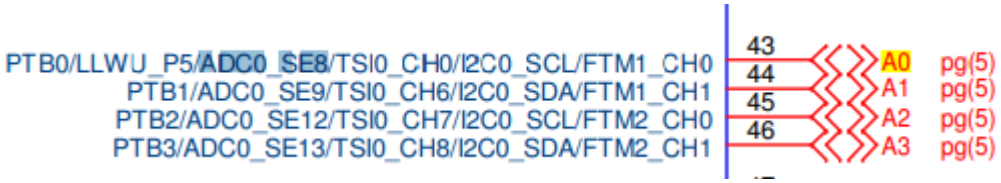


Figure 2 Vizualizarea canalului ADC0_SE8 în Schematics [1]

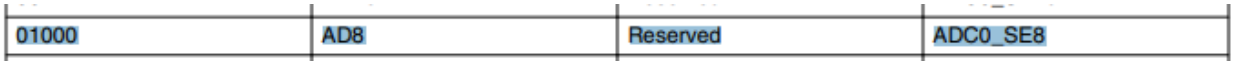


Figure 3 Vizualizarea canalului 8 în Reference Manual [2]

În Figura 2 am ilustrat felul în care am detectat faptul ca pinul A0 poate fi accesat prin itnermediul canalului ADC0_SE8, care, după cum am putut vedea și în Reference Manual (Figura 3), acesta este reprezentat de canalul 8 (1000 binar).

Registrul ADC0_RA

ADC Data Result Register folosit pentru a întoarce și a stoca rezultatul conversiei. [2]

Registrului ADC0_CFG1

ADC Configuration Register 1 este folosit pentru configurarea modulului ADC și anume selectarea modului de operare (în cazul nostru pe 10 biți), sursa ceasului, eșantionarea ceasului (clock divide), low power mode, după cum urmează:

Mode = 2

```
ADC0_CFG1 |= (ADC_CFG1_MODE(2)   |    // 10 bits mode
```

Figure 4 Configurarea ADC pe 10 biți în codul sursă

Figure 5 Explicația alegerii mode =2 conform Reference Manual [2]

Modul 2 e selectat pentru conversia datelor pe 10 biți single-ended și presupune 20 de cicluri de ceas (20 ADCK cycles), conform tabelului *Table 28-72 Base conversion time (BCT)* din *Reference Manual* [2].

Input Clock Select = 1

"Selects the input clock source to generate the internal clock, ADCK. Note that when the ADACK clock source is selected, it is not required to be active prior to conversion start." (Reference Manual, pg. 466) [2]

```
ADC0_CFG1 |= (ADC_CFG1_MODE(2)   |    // 10 bits mode
             ADC_CFG1_ADICLK(1)|    // Input Bus Clock/2 (24 Mhz)
```

Figure 6 Configurarea Input Clock Select =1 în codul sursă

S-a selectat valoarea 1 corespunzătoare unei divizări a frecvenței default a ceasului cu 2, valoare provenită din calculul 2^1=2.

Clock Divide Select = 3

Bitul ADIV setează eșantionarea utilizată de ADC pentru generarea ceasului intern ADCK. *(Reference Manual, pg. 466)*

```
             ADC_CFG1_ADIV(3)   |      // Clock divide by 8 (3 Mhz)
```

Figure 7 Setarea Clock Divide Select cu 3 în codul sursă

Pentru apelarea cu valoarea 3, în Reference Manual se specifică " The divide ratio is 8 and the clock rate is (input clock)/8 " (Figura 7) Practic, am ales o eșantionare cu 8 a frecvenței de ceas.

Low-Power Configuration = 1

Controlează configurația de putere a convertorului de aproximare succesiv. Acest lucru optimizează consumul de energie atunci când nu sunt necesare rate de eșantionare mai mari (*Reference Manual*, pg. 466) [2]

```
                    ADC_CFG1_ADLPC_MASK); // Low power mode
```

1 Low-power configuration. The power is reduced at the expense of maximum clock speed.

Figure 8 Selecția modului Low Power conform Reference Manual [2]

Registrul ADC0_SC1A

ADC Status and Control Registers 1 este folosit în proiect pentru activarea întreruperilor (configurare interrupt enable prin ADC_SC1_AIEN_MASK), setarea modului single-ended (prin ADC_SC1_DIFF_MASK setat pe 0) dar și pentru obținerea unui flag cu statusul conversiilor realizate pe canalul AD8 (ADC_SC1_COCO_MASK).

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_B000	ADC Status and Control Registers 1 (ADC0_SC1A)	32	R/W	0000_001Fh	28.3.1/462

Figure 9 Vizualizarea detaliilor despre registru ADC0_SCA1 [2]

Interrupt Enable = 1
Valoarea 1 reprezintă "Conversion complete interrupt is enabled". [2]
"Enables conversion complete interrupts. When COCO becomes set while the respective AIEN is high, an interrupt is asserted." (Reference Manual, pg. 464) [2]

Differential Mode Enable = 0
Valoarea 0 pentru acest bit reprezintă selectarea conversiei single-ended și a canalului de input.
"Configures the ADC to operate in differential mode. When enabled, this mode automatically selects from the differential channels, and changes the conversion algorithm and the number of cycles to complete a conversion.**"** (Reference Manual, pg. 464) [2]

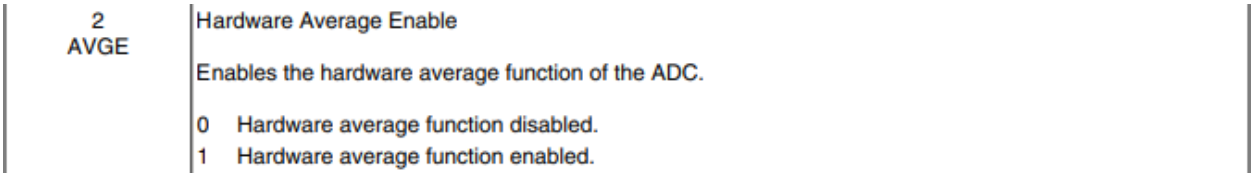
Input channel select = 01000 (8)
Conform Reference Manual, când bitul DIFF este 0, este selectat default canalul AD8.
Input channel select **"**Selects one of the input channels. The input channel decode depends on the value of DIFF.**"** (Reference Manual, pg. 464) [2]

Conversion Complete Flag = 0

Modulul rulează într-un while cât timp acest flag este zero. Valoarea zero anunță faptul că nu s-a temrinat de realizat conversia. **"**This is a read-only field that is set each time a conversion is completed when SC3[AVGE]=0.**"** (Reference Manual, pg. 464) [2]

Registrul ADC0_SC3

ADC Status and Control Registers 3 este folosit în cadrul proiectului pentru calibrare în vederea obținerii unei acurateț̦e cât mai bună a valorilor convertite, pentru setarea conversiei continue și a funcției hardware average enable. *"The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average* ***function is enabled by setting SC3[AVGE]*** *and operates in any of the conversion modes and configurations."(Reference Manual, pg. 483) [2]*



2 AVGE	Hardware Average Enable Enables the hardware average function of the ADC. 0 Hardware average function disabled. 1 Hardware average function enabled.

Figure 10 AVGE flag corespunzător registrului ADC0_SC3 [2]

În cadrul funcției init_ADC() are loc setarea registrului bitului AVGE din ADC0_S3 cu valoarea 0.*"The hardware average function can be enabled by setting SC3[AVGE]=1 to perform a hardware average of multiple conversions." (Reference Manual, pg.492) [2]*

Registrul SIM_SCGC6

System Clock Gating Control Register 6, folosit pentru pornirea semnalului de ceas pentru :

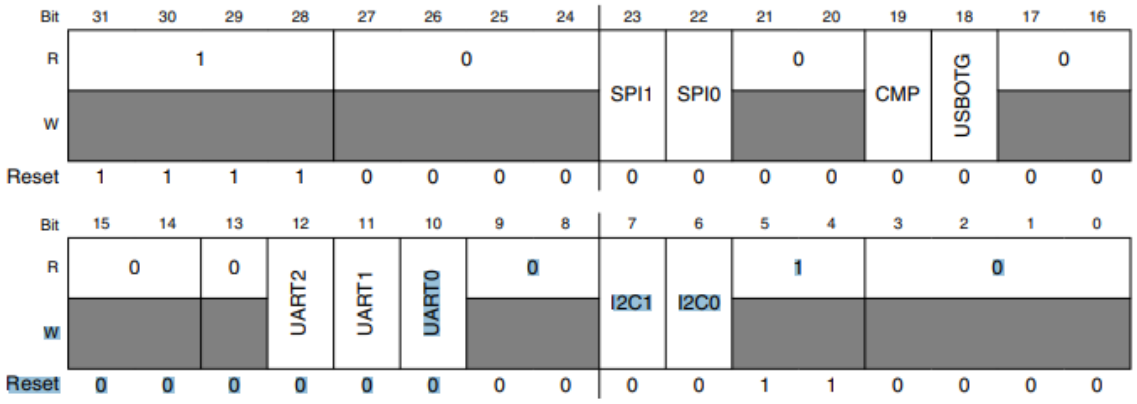
- ADC, prin setarea bitului ADC0 cu 1: "This bit controls the clock gate to the ADC0 module", (Reference Manual, pg. 208) [2]
- PIT, prin setarea bitului PIT „PIT Clock Gate Control": "This bit controls the clock gate to the PIT module." (Reference Manual, pg. 208) [2]

Registrul SIM_SCG4

System Clock Gating Control Register 4, folosit pentru a putea face enable la semnalul de ceas pe UART0. Registrul l-am setat cu valoarea 0x0400 .

12.2.8 System Clock Gating Control Register 4 (SIM_SCGC4)

Address: 4004_7000h base + 1034h offset = 4004_8034h



Figură 11 Vizualizarea registurlui SIM_SCG4 [2]

În Figura 11 se poate vizualiza bitul UART0 care trebuie setat pe poziția 10, corespunzătoare bitului 10. Activarea acestui bit e echivalentă cu activarea semnalului de ceas pentru UART0.

Registrul SIM_SCG5

System Clock Gating Control Register 5 este folosit pentru a putea seta, în cadrul modulului UART, ceasul pentru portul A pe care se face transmisia, corespunzător bitului 9. De aceea, registrul SIM_SCG5 a fost setat cu valoarea 0x0200.

Registrul PIT_TFLG1

Acest registru reține PIT flagul de întrerupere. În întreruperea PIT_IRQHandler(), suprascrisă în main(), am resetat PIT_TFLG1 cu valoarea 1.

Registrul PIT_MCR

Acest registru activează sau dezactivează ceasurile PIT și controlează temporizatoarele atunci când PIT intră în modul Debug. Registrul este setat cu 0, setând astfel oprirea timer-ului când se intră în modul Debug.

Registrul PIT_LDVAL1

Timer Load Value Register. Valoarea numărătorului este încărcată din acest registru. În proiectul nostru, acest registru se inițializează cu valoarea 0x000FA000 corespunzătoare unei întreruperi de ceas la 40 ms pentru frecvența default de 21 mhz a ceasului.

Registrul PIT-TCTRL1

Timer Control Register, acesta controlează fiecare timer(TIE, TEN). Am setat bitul TIE (Time Interrupt Enable) și TCTRL (Timer Enable).

Registrul SIM_SOPT2

System Options Register 2, folosit pentru a putea selecta sursa de ceas UART0. Registrul a fost setat cu valoarea 0x04000000 (ce presupune setarea bitului 27 corespunzătoare flag-ului UART0SRC din registrul SIM_OPT2).
Modulul UART0 are un ceas selectabil. În figura de mai jos se poate vedea cum se genereaza UART0 clock:

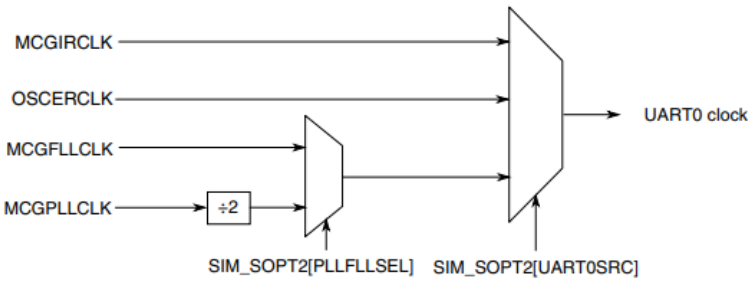


Figure 5-7. UART0 clock generation

Figure 12 Generarea semnalului de ceas pentru modulul UART0

Celelalte două module UART1, UART2 disponibile funcționează pe ceasul de magistrală. (Reference Manual, pg 125). În Figura 12 se poate observa circuitul de generare a semnalului de ceas UART0 influențat de semnalul SIM_SOPT2.

Registrul UART0_C2

UART Control Register 2, inițial setat cu 0 pentru a dezactiva UART0 pe timpul setării configurațiilor.

După setarea baudrate(57600), 8-bit Mode, OSR(over sampling ratio 15), acest registrul l-am setat cu valoarea 0x08, ce presupune setarea bitului 3 Transmitter Enable). Practic, prin intermediul acestui registru am activat transmisia.

Regiștrii UART0->BDL și UART0->BDH

UART Baud Rate Register High și UART Baud Rate Register Low, acest regiștrii controlează divizorul de prescalare pentru baudrate UART. Prin intermediul lui am setat bauderate cu 57600.

Registrul UART0_C4

Acest registru a fost folosit pentru a putea seta pe modulul UART0 câmpul OSR (Over Sampling Ratio la recepție). Acesta a fost setat cu valoarea 0x0F, corespunzătoare valorii 15 default.

Registrul UART0_C1

Acest registru de citire/scriere controlează diverse caracteristici opționale ale sistemului UART. L-am setat cu 0, de interes fiind doar bitul M corespunzător 8-Bit Mode Select, ceea ce presupune că la recepție și la transmisie se folosesc date pe 8 biți.

Registrul PORTA_PCR2

Pin Control Register 2, folosit pentru a putea activa pinul A, atribuindu-i valoarea 0x0200, corespunzătoare setării câmpului MUX cu valoarea binară 100, corespunzătoare alternativei 4. Multiplexarea pinului se poate observa în figura de mai jos:

28	24	19	12	PTA2	DISABLED	TSI0_CH3	PTA2	UART0_TX	TPM2_CH1

Figure 13 Multiplexarea pinului PTA2

În Figura 13 se poate observa faptul că pinul de transmisie UART este asociat alternativei PTA2.

Registrul UART0_S1

UART Status Register 1, prin acest registru se verifică dacă un octet a fost primit complet la transmisie, verificând bitul TDRE(Transmit Data Register Flag) din cadrul registrului.

Registrul UART0_D

UART Data Register, folosit pentru a scrie date în el.

FRDM-KL25Z - vizualizare pini

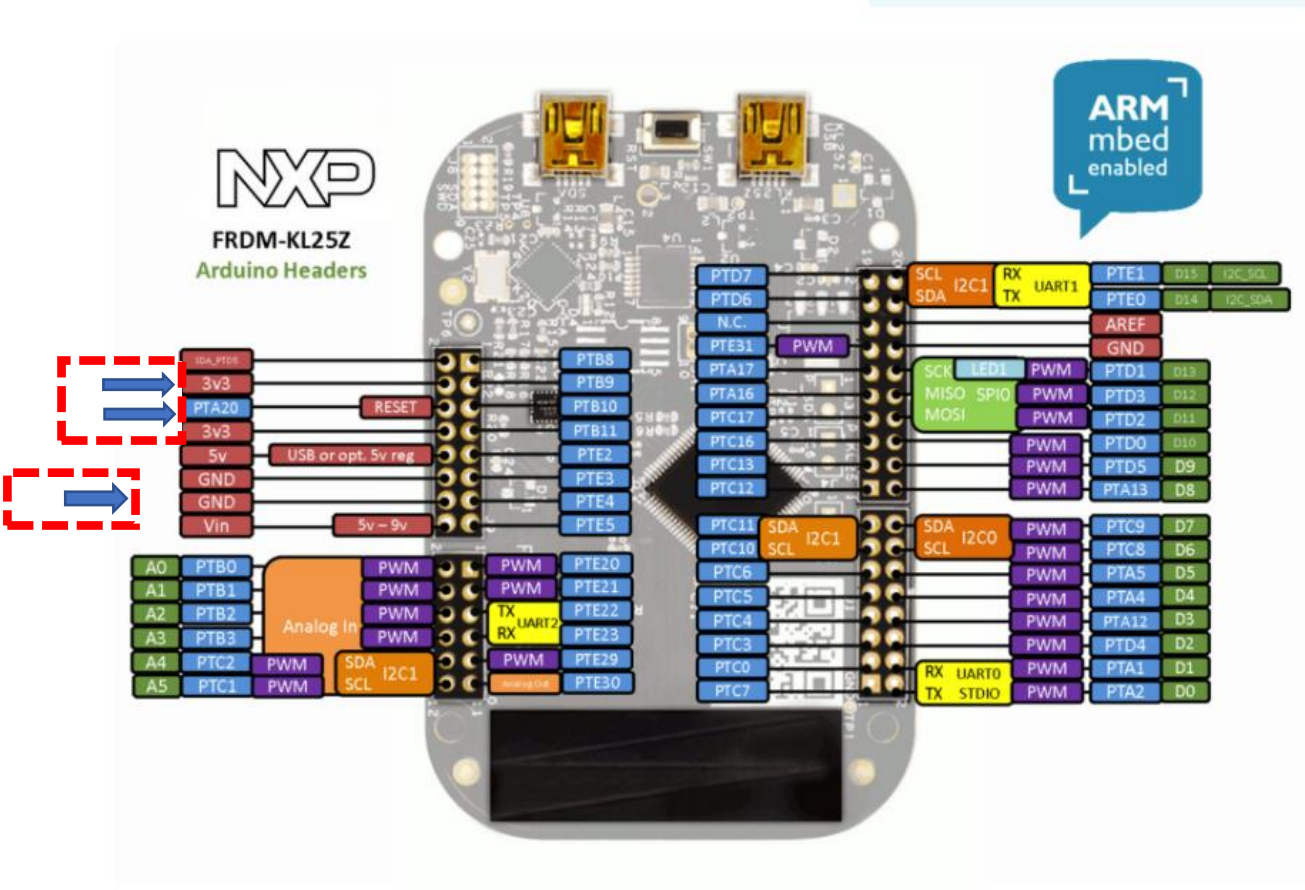


Figure 14 Amplasarea pinilor folosiți pe placă [3]

- PTE20 - Pinul analog input
- PTE3 - Pinul pentru GDN
- PTE2 - Pinul pentru VCC

Conectarea plăcii la senzorul de flacără s-a realizat conform schemei de mai sus prin intermediul pinilor PTE20 (A0), PTE3, PTE2, astfel:
Firul roșu al senzorului, respectiv **verde**, este pentru alimentare – corespunzător pinului PTE2.
Firul negru al senzorului, respectiv **galben**, pentru GND – corespunzător pinului PTE3.
Firele albastre pentru citire date analog in – corespunzător pinului A0.

Acest set-up poate fi vizualizat în pozele realizate mai jos, în două variante zoom-in și zoom-out:

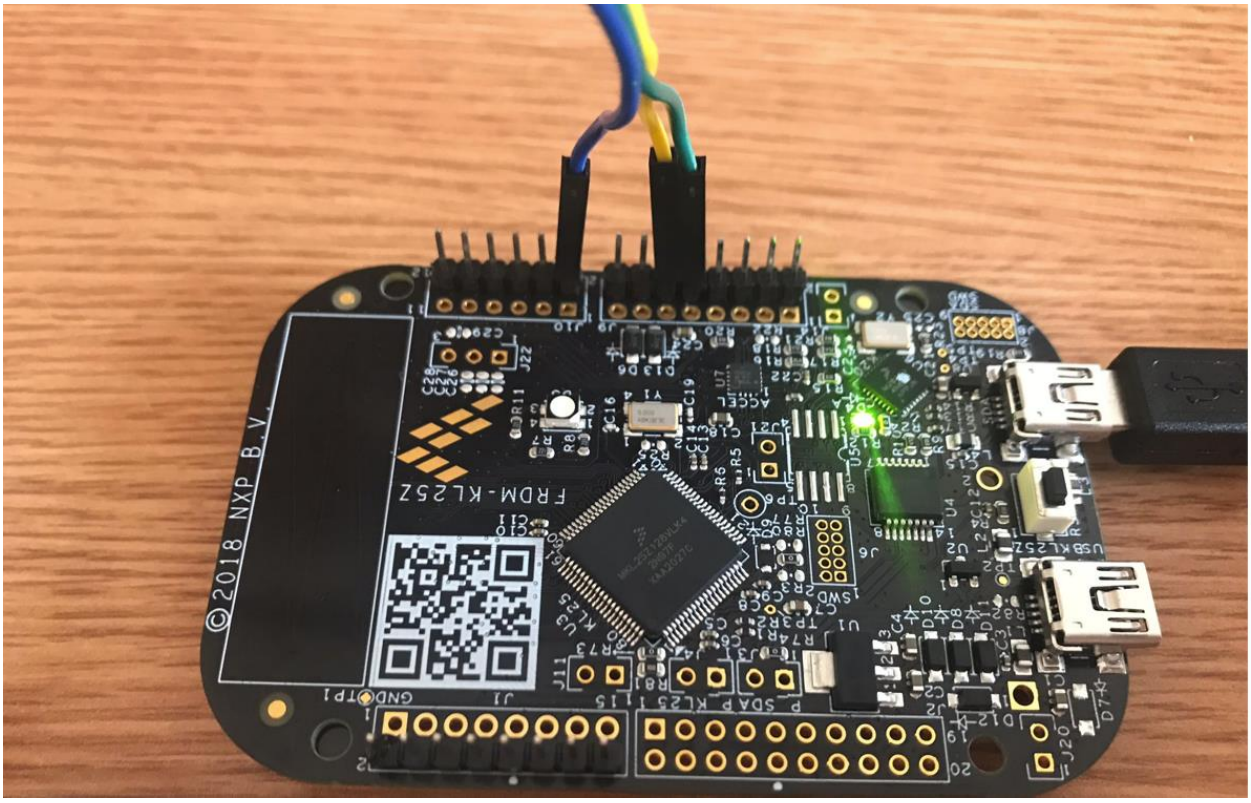


Figure 15 Zoom-In set-up

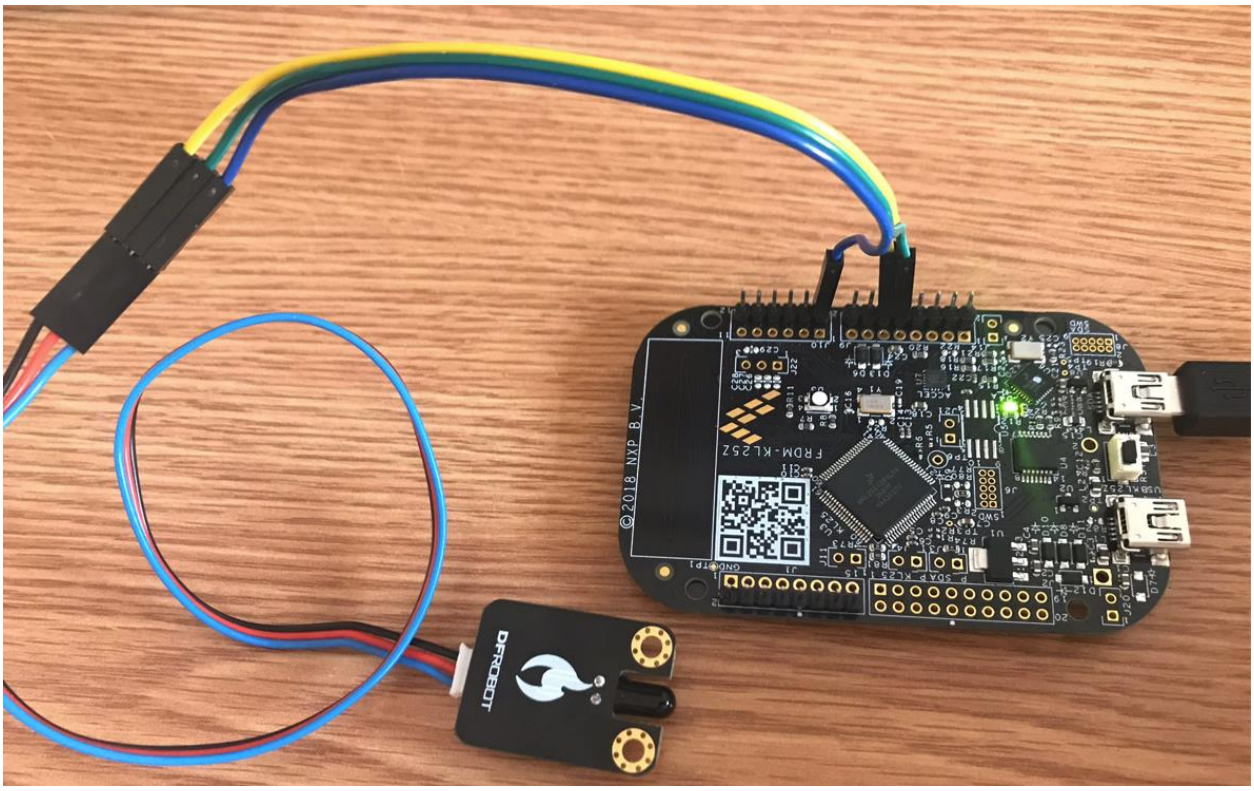


Figure 16 Zoom Out Set-up

Se poate observa modul de conectare al pinilor senzorului de foc la cei de pe placă. În tabelul de mai jos puteți vizualiza descrierea pinilor senzorului de foc:

Table 1 Descrierea pinilor senzorului de foc

PIN	DESCRIERE
VCC	3.3 – 5V
GND	Ground
ANALOG OUT	Analog output voltage proporțional cu lungimea de undă a luminii, practic a radiației emise de o sursă externă.

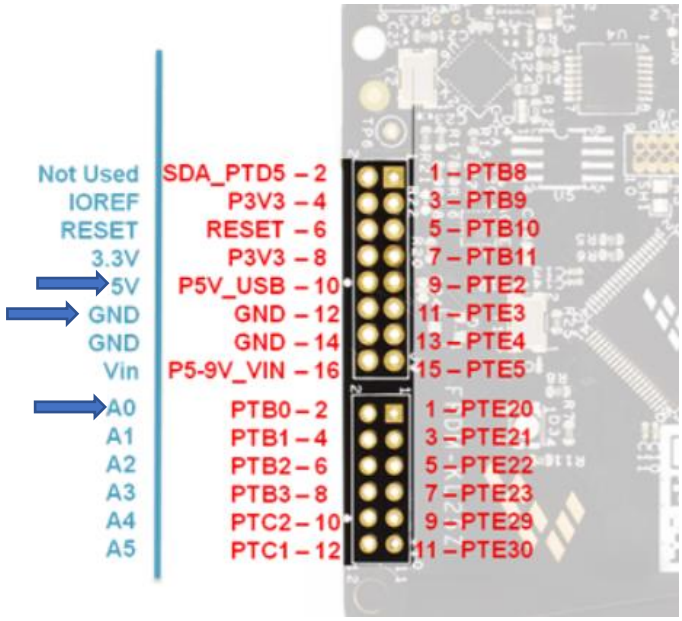


Figure 17 Zoom-in al pinilor folosiți [4]

Configurarea pinilor folosiți a fost realizată conform Schematics, după cum urmează:

- **Pinul A0** este folosit pe canalul AD8, pe care îl dăm ca argument funcției ADC_Read().

Specificații hardware relevante dezvoltării proiectului pentru microprocesorul KL25Z

- **Toți pinii** numerotați (PT_XX) pot fi utilizați și ca interfețe DigitalIn și DigitalOut.
- **Periferice analogice:** periferic analogic 16-bit SAR ADC w/ DMA suport
  - Modului ADC de până la 16 biți cu rezoluție configurabilă, eșantion timp și viteza de conversie/putere.
  - Senzor de temperatura integrată.
- **Periferice de comunicare:** un UART de putere redusă și două module UART standard

- **Timere**: temporizator de întrerupere periodică cu 2 canale (PIT)

Flame sensor SKU DFR0076

Senzorul de flacără poate fi utilizat pentru a detecta focul sau alte lungimi de undă la 760 nm ~ 1100 nm de lumină. Temperatura de funcționare a senzorului de flacără este de **la -25 grade Celsius până la 85 grade Celsius.** Este de remarcat că distanța sondei de flacără nu trebuie să fie prea apropiată pentru a evita deteriorarea (10 - 100 cm). [5]

Când senzorul de flacără detectează flacăra, modificarea numerică poate fi observată pe grafic. Ca senzor analog, citirile sunt afectate de intensitatea flăcării și poziția acesteia (unghi și distanță).
Gama de lățime de bandă spectrală a senzorului este: 760 nm până la 1100 nm, deci poate detecta orice lumină a cărei lățime de bandă spectrală este între 760-1100 nm. [5]
Acesta poate fi acționat nu numai de flacără, ci și de radiații infraroșii (Analog citire: 900), lămpi de neon în sălile de clasă și chiar și de blițul telefonului (Analog citire: 50).

Detalierea modulelor

În această secțiune vom detalia modulele folosite în cadrul programului sursă, pentru dezvoltarea aplicației. Modulele folosite sunt: modulul ADC, modulul UART și modulul PIT.

Modulul ADC [6]

“The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm. To meet accuracy specifications, the ADC module must be calibrated using the on-chip calibration function.” (Reference Manual, pg.482) [2]
Modulul ADC este reprezentat de un convertor analog-digital pe 10 biti pentru frecvența de 21 mhz (la frecv asta default modulul adc poate lcura max pe ADC – el e pe 16 pe placuta dar nu poate functiona) având o valoare maximă de retur de 2^10 - 1 = 1023. Inițializarea acestuia se face in funcția init_ADC() prin pornirea semnalului de ceas. Așadar, valorile primite de la ADC vor fi în intervalul [0,1023].
Pornirea semnalului de ceas se realizează prin setarea registrul **SIM_SCGC6**.
Configurarea modulului ADC se face prin setarea registrului **ADC0_CFG1.**
Implementarea modulului poate fi vizualizată în imaginea de mai jos.

```c
#include "Adc.h"

void init_ADC(void)
{

    // Enable ADC0 Clock
    SIM_SCGC6 |= SIM_SCGC6_ADC0_MASK;


    // Configure ADC
    ADC0_SC1A |= (ADC_SC1_AIEN_MASK);  // Interrupt enabled
    ADC0_SC1A &= ~ADC_SC1_DIFF_MASK;   // Single Ended ADC

    ADC0_CFG1 = 0; // Reset register
    ADC0_CFG1 |= (ADC_CFG1_MODE(2)   |   // 10 bits mode
                  ADC_CFG1_ADICLK(1)|   // Input Bus Clock/2 (24 Mhz)
                  ADC_CFG1_ADIV(3)   |     // Clock divide by 8 (3 Mhz)
                         ADC_CFG1_ADLPC_MASK); // Low power mode

    ADC0_SC3  &= ~ADC_SC3_AVGE_MASK;
}
/*************************************************
 * @function: adc_read
 * @brief:    Read the ADC Module
 * @param:    ch - ADC Channel
 * @return:   none
 *************************************************/
uint16_t ADC_Read(uint8_t ch)
{
    // Write to SC1A to start conversion
    ADC0_SC1A = ((ch & ADC_SC1_ADCH_MASK) |
                    (ADC0_SC1A &
                (ADC_SC1_AIEN_MASK | ADC_SC1_DIFF_MASK)));

    while(ADC0_SC2 & ADC_SC2_ADACT_MASK);    // Conversion in progress
    while(!(ADC0_SC1A & ADC_SC1_COCO_MASK)); // Run until the conversion is complete

    return ADC0_RA;
}
```

Figure 19 Implementarea modului ADC [6]

Modulul PIT

Modulul PIT reprezintă un numărător care scade în funcție de frecvența semnalului de ceas. În implementarea noastră, registrul **PIT_LDVAL1** a fost inițializat cu valoarea 0x000FA000 (1024000) pentru a realiza o întrerupere și o resetare a modulului PIT o dată la 40ms. Frecvența ceasului pentru placă este de 21 mhz. În imaginea de mai jos se poate vizualiza cum am implementat modulul PIT.

```
#include "Pit.h"


void initPIT()
{
  SIM->SCGC6 |= SIM_SCGC6_PIT_MASK;

  PIT_MCR = 0x00;

  PIT_LDVAL1 = 0X000FA000;  //1024000 timer (40 ms)

  PIT_TCTRL1 = PIT_TCTRL_TIE_MASK;
  PIT_TCTRL1 |= PIT_TCTRL_TEN_MASK;

  NVIC_ClearPendingIRQ(PIT_IRQn);
  NVIC_SetPriority(PIT_IRQn,128);
  NVIC_EnableIRQ(PIT_IRQn);
}
```

Figure 20 Implementarea modulului PIT [7]

Când valoare din **PIT_LDVAL1** ajunge la 0 , numărătorul se resetează și este apelată întreruperea **PIT_IRQHandler** suprascrisă in fișierul main.c .

```
void PIT_IRQHandler(void)
{

  PIT_TFLG1 = 0x01;
  UART0_Transmit16bit(valoare);
}
```

Figure 21 Suprascrierea întreruperii PIT [7]

Modulul UART

Modulul UART are ca scop realizarea comunicației seriale. Pentru implementarea noastră am ales un baudrate de 57600 biți/secundă prin setarea regiștrilor **UART0->BDL** si **UART0->BDH,** corespunzători octeților low, respectiv high**.** În fișierul uart.c sunt implementate si funcțiile pentru transmiterea prin comunicație serială a unei valori pe 8 și pe 16 biți.

```
void UART0_Transmit(uint8_t data)
{

    while(!(UART0->S1 & UART0_S1_TDRE_MASK));
            UART0->D = data;

}

void UART0_Transmit16bit(uint16_t data)
{


    uint8_t f8=((uint16_t)data & 0xff);
    uint8_t l8=((uint16_t)data  >> 8 & 0xff);

    UART0_Transmit(f8);
    UART0_Transmit(l8);

}

void uartInit(){
    //Activam ceasul pentru modulul UART0
  SIM->SCGC4 |= 0x0400;   /* enable clock for UART0 */
    SIM->SOPT2 |= 0x04000000;    /* use FLL output for UART Baud rate generator */
    UART0->C2 = 0;          /* turn off UART0 while changing configurations */
    UART0->BDH = 0x00;
    UART0->BDL = 0x17;      /* 57600 Baud */
    UART0->C4 = 0x0F;       /* Over Sampling Ratio 16 */
    UART0->C1 = 0x00;       /* 8-bit data */
    UART0->C2 = 0x08;       /* enable transmit */

    SIM->SCGC5 |= 0x0200;   /* enable clock for PORTA */
    PORTA->PCR[2] = 0x0200; /* make PTA2 UART0_Tx pin */
}
```

Figure 22 Implementarea modulului UART [7]

Implementarea sursei main

Funcția main() inițializează cele 3 module descrise mai sus.
Într-o buclă infinită este citită constant de la modulul de ADC valoare convertită din semnalul transmis de senzor.
O dată la 40ms se apelează întreruperea **PIT_IRQHandler(void),** suprascrisă în fișierul main.c, care are rolul de a transmite prin intermediul modulului UART variabile pe 16 biți.
Implementarea funcției main() și suprascrierea întreruperii sunt ilustrate în imaginea de mai jos.

```c
#include "MKL25Z4.h"
#include "Uart.h"
#include "Adc.h"
#include "Pit.h"


uint16_t valoare;
void PIT_IRQHandler(void)
{

  PIT_TFLG1 = 0x01;
  UART0_Transmit16bit(valoare);
 }


int main(void)
{
  uartInit();
  init_ADC();
  initPIT();

  for (;;)
  {
    valoare = ADC_Read(8);
  }

  return 0;

}
```

Figure 23 Implementarea funcției main()

Rezultate obținute

Scriptul matlab constă în inițializarea unei conexiuni seriale cu baudrate de 57600 biți/secundă (același cu cel setat în sursa uart.c) și în inițializarea unui vector gol în care vor fi stocate valorile primite de la modulul UART al programului.

```
clc
clear all

s = serialport("COM5",57600)

vector=[];
valoare=read(s,1,"uint16")
while true
    valoare=read(s,1,"uint16")
    vector(end+1)=valoare;
    h=area(vector)
    grid on
    set(h(1),'FaceColor',[1 0 0])
    drawnow;
end
```

Figure 14 Scriptul Mathlab

Într-o buclă infinită se citește o valoare de pe conexiunea serială ce se adaugă în vectorul neinițializat și se redesenează graficul. În imaginea de mai jos se pot vizualiza rezultatele obținute.

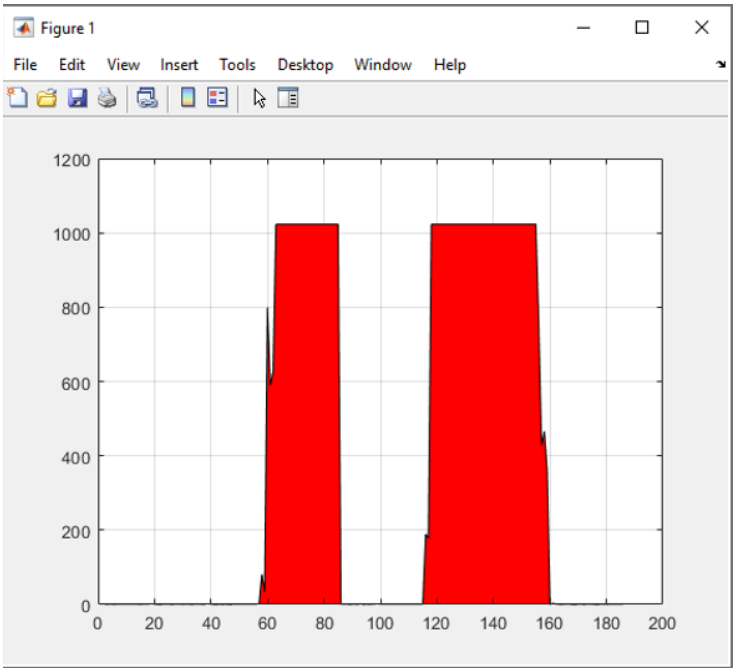
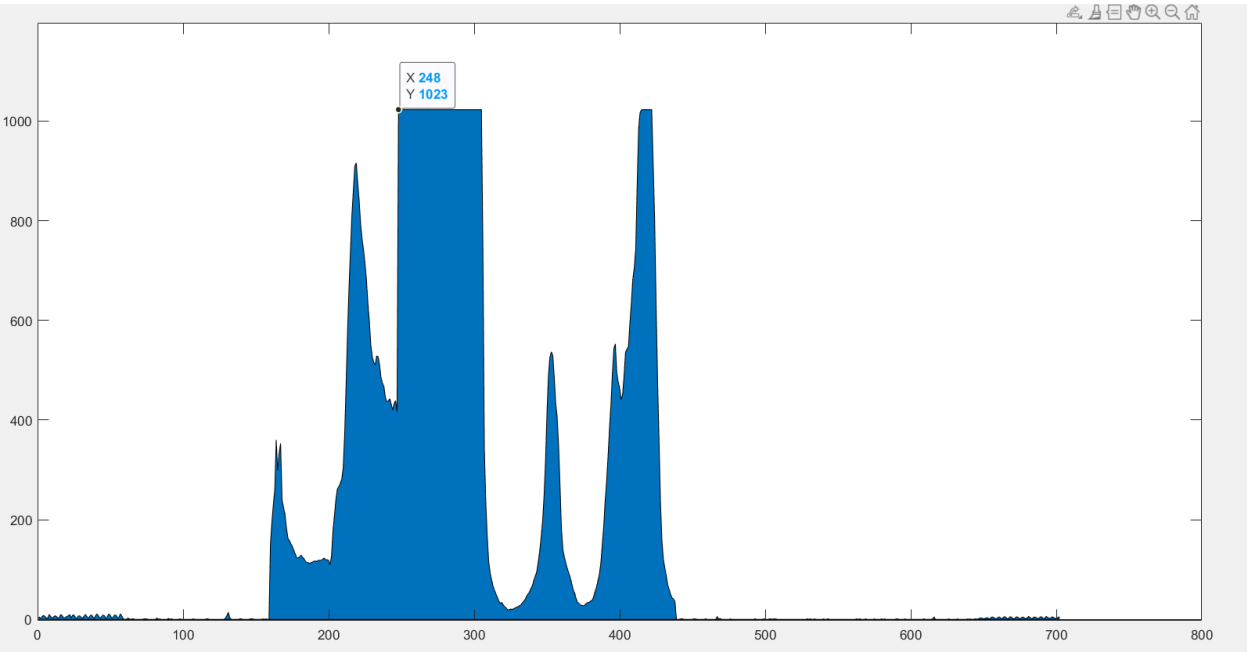


Figure 25 Graficul Valorilor citite de la senzorul de flacără



Figură 26 Vizualizarea valorii maxime 1023

Interpretarea graficelor

Graficul a fost realizat folosind funcția **area()** ce primește ca argument un vector de valori înregistrate de la senzor.

Cele două grafice de mai sus sunt construite în două momente de timp diferite ale rulării proiectului. Primul grafic își propune să evalueze comportamentul senzorului într-o cameră întunecată la aprinderea și stingerea unei surse de foc. Cel de-al doilea grafic își propune să ilustreze comportamentul senzorului într-o camera slab luminată, cu lumină naturală, la apropierea și îndepărtarea, nu foarte rapidă, a unei surse de foc. Întrucât counter-ul folosit în codul sursă este setat la 40 de ms, valorile primite de modulul UART și trimise către afișare scriptului Maltab, vor apărea tot la 40 ms distanță.

Pentru lucrul cu senzorul de foc este indicată o distanță minină de 10 cm. Distanța maximă de detecție a sursei de foc, conform specificațiilor hardware, este de 100 cm.

În Figura 25 de mai sus au fost înregistrate cu aproximare valorile: [40,800,1023,0,1,200,440, 0], care se pot vizualiza pe axa OY, iar pe OX indicii acestora. Valoarea maximă obținută este 1023 și este atinsă atunci când focul este foarte aproape de senzor (aproximativ 9-10 cm), iar restul valorilor depind de radiația ambientală și de unghiul lungimii de undă realizat cu senzorul.

În Figura 26 din cel de-al doilea grafic am reprezentat valorile convertite obținute de la senzor prin apropierea și îndepărtarea unei surse de foc provenite de la o brichetă, în întuneric (fără lumină de la bec). Se poate observa valoarea maximă 1023 în dreptul punctului determinat de coordonatele (248,1023) reprezentat pe grafic.

Observații

Am observat faptul că senzorul nu este acționat doar de flacără, ci și de lumina ambientală din cameră. La o luminozitate puțin mai mare decât medie pe grafic s-au putut citi valori în intervalul 30-40, acestea provenind de la radiația luminii naturale. Pentru lămpile de neon din camera de studiu am interceptat valori din intervalul 40-50 și chiar și de la blițul telefonului (analog citire: 50).
Aceste rezultate demonstrează faptul că valoarea de ieșire analog output voltage e proporțională cu radiația emisă de o sursă de lumină externă și nu doar de o sursă de foc.

Bibliografie

1. https://www.nxp.com/downloads/en/schematics/FRDM-KL25Z_SCH_REV_D.pdf "FRDM-KL25Z Schematics"
2. https://www.apogeeweb.net/upload/pdf/20210903/KL25Z-Reference-Manual.pdf "KL25 Sub-Family Reference Manual"
3. https://os.mbed.com/platforms/KL25Z/ "KL25Z MCU Specifications and Features – detailed illustrations of the location of pines and their purpose"
4. https://www.mouser.com/pdfdocs/FRDM-KL25Z.pdf "FRDM-KL25Z User's Manual "
5. https://wiki.dfrobot.com/Flame_sensor_SKU__DFR0076 "SKU_DFR0076 Flame sensor introductory tutorial"
6. Interfacing LM35 Temperature Sensor with KL25Z Series MCU – Learning Embedded System and Microcontrollers (wordpress.com) "SKU_DFR0076 Flame sensor complete tutorial and sensor features"
7. https://github.com/peterfillmore/frdm-kl25z-exploit-practice-code/blob/master/uart.c " Repository of a similar project"
8. https://mcuoneclipse.com/2013/02/10/tutorial-adc-with-the-freedom-board/ "ADC tutorial "