ROMÂNIA

MINISTERUL APĂRĂRII NAȚIONALE

ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”

Facultatea de Sisteme Informatice și Securitate Cibernetică

Departamentul de Calculatoare și Securitate Cibernetică

UTILIZARE SENZOR TSI ȘI A SERVOMOTORULUI PLATFORMA DE DEZVOLTARE FRDM-KL25Z

Std. plt. Alexandra-Ioana BUZĂȚOIU
Std. sg. maj. Denisa-Mihaela OPREA
Std. sg. maj. Alexandra BOIANGIU

Grupa C114D

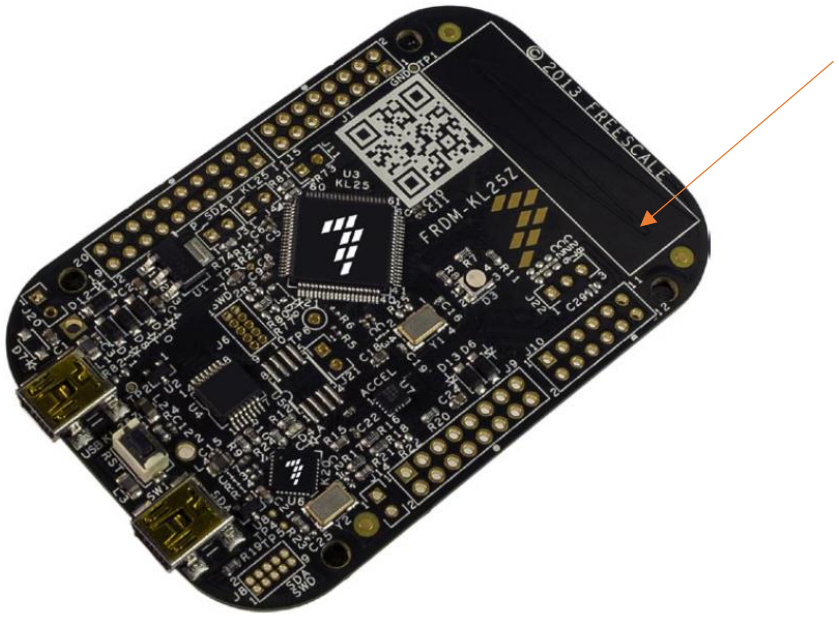
București

2022

Cuprins

1. Prezentarea senzorului integrat TSI

Modulul de intrare cu detecție capacitivă la atingere (TSI) oferă sensibilitate ridicată și robustețe sporită. Fiecare pin TSI implementează măsurarea capacitivă printr-o scanare a sursei de curent, încărcarea și descărcarea electrodului, o dată sau de mai multe ori. Un oscilator de referință bifează timpul de scanare și stochează rezultatul pe 16 biți înregistrând când scanarea se încheie. Între timp, o cerere de întrerupere este trimisă CPU pentru post-procesare dacă întreruperea TSI este activată și funcția DMA nu este selectată



Senzorul TSI integrat are următoarea schemă bloc:

42.1.3 Block diagram

The following figure is a block diagram of the TSI module.

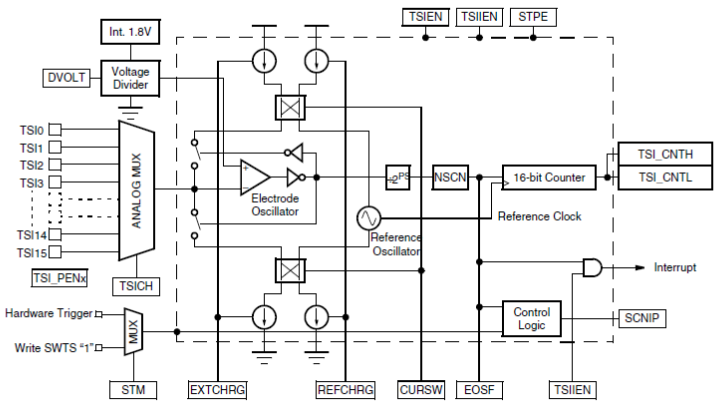


Figure 42-1. TSI module block diagram

2. Scop proiect

Scopul acestui proiect este acela de a modifica unghiul de rotație al servomotorului utilizând senzorul tactil integrat, printr-un semnal PWM generat și manipulat în funcție de plaja de valori a senzorului TSI.

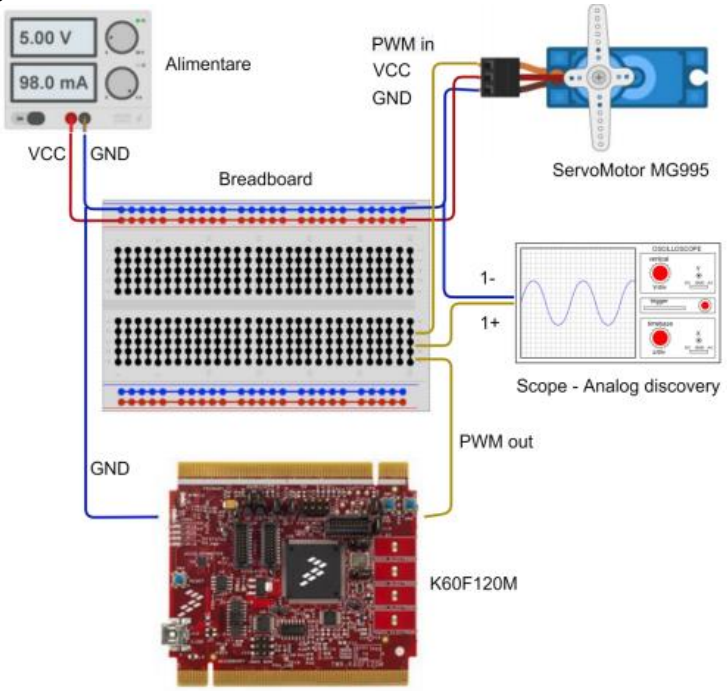
Se va dezvolta un program care, cu frecvența de 1kHz (o dată la 1 ms), va verifica ieșirea senzorului conectat. O dată la 150 ms, în funcție de câte valori de 0 (atingere detectată) a contorizat programul, se generează un semnal PWM cu factorul de umplere direct proporțional cu procentul de timp în care senzorul a detectat atingere. Acest semnal PWM va fi conectat către servomotor și va modifica astfel înclinația unghiului de rotație a acestuia în funcție de factorul de umplere.

De asemenea, se va transmite prin UART către PC , o dată la 10 ms, valoarea ieșirii senzorului TSI pentru a se putea realiza un grafic în timp real prin intermediul bibliotecii matplotlib din python în care să se observe modificarea unghiului de rotație în funcție de inputul primit prin senzorul TSI.

3. Conectare senzor – placă de dezvoltare

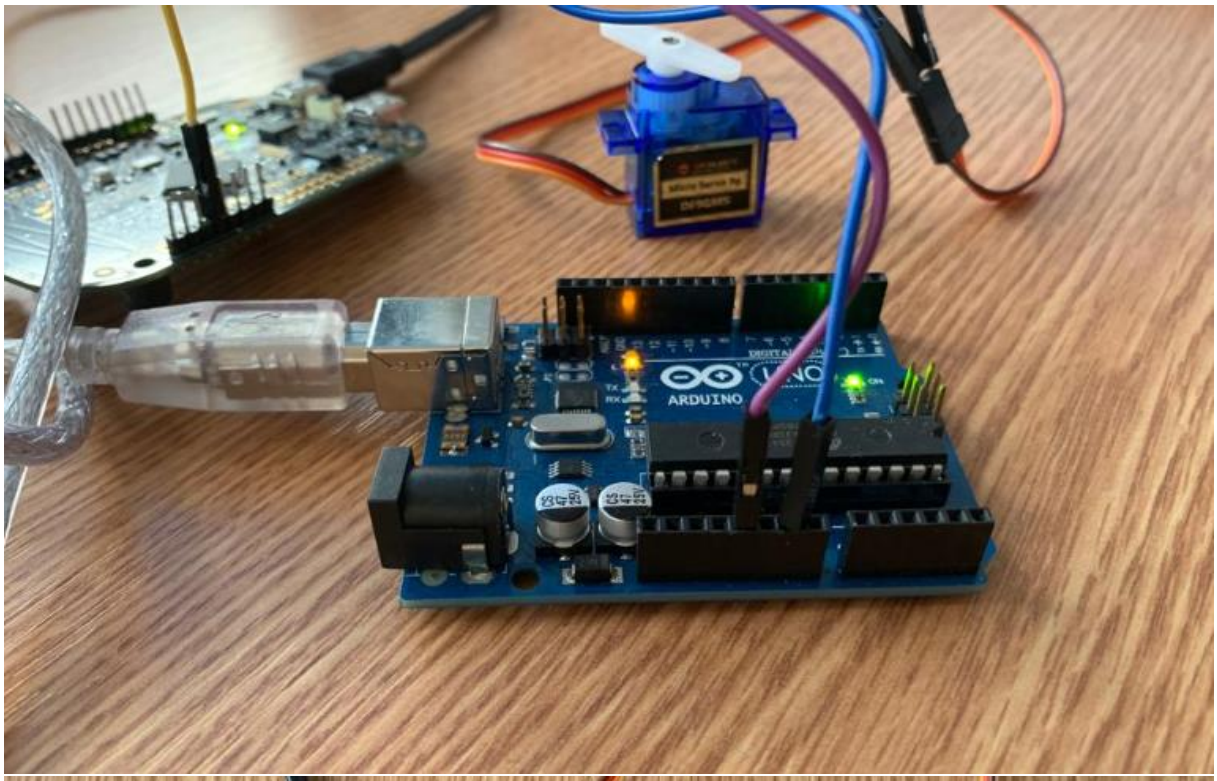
Vom conecta senzorul astfel:

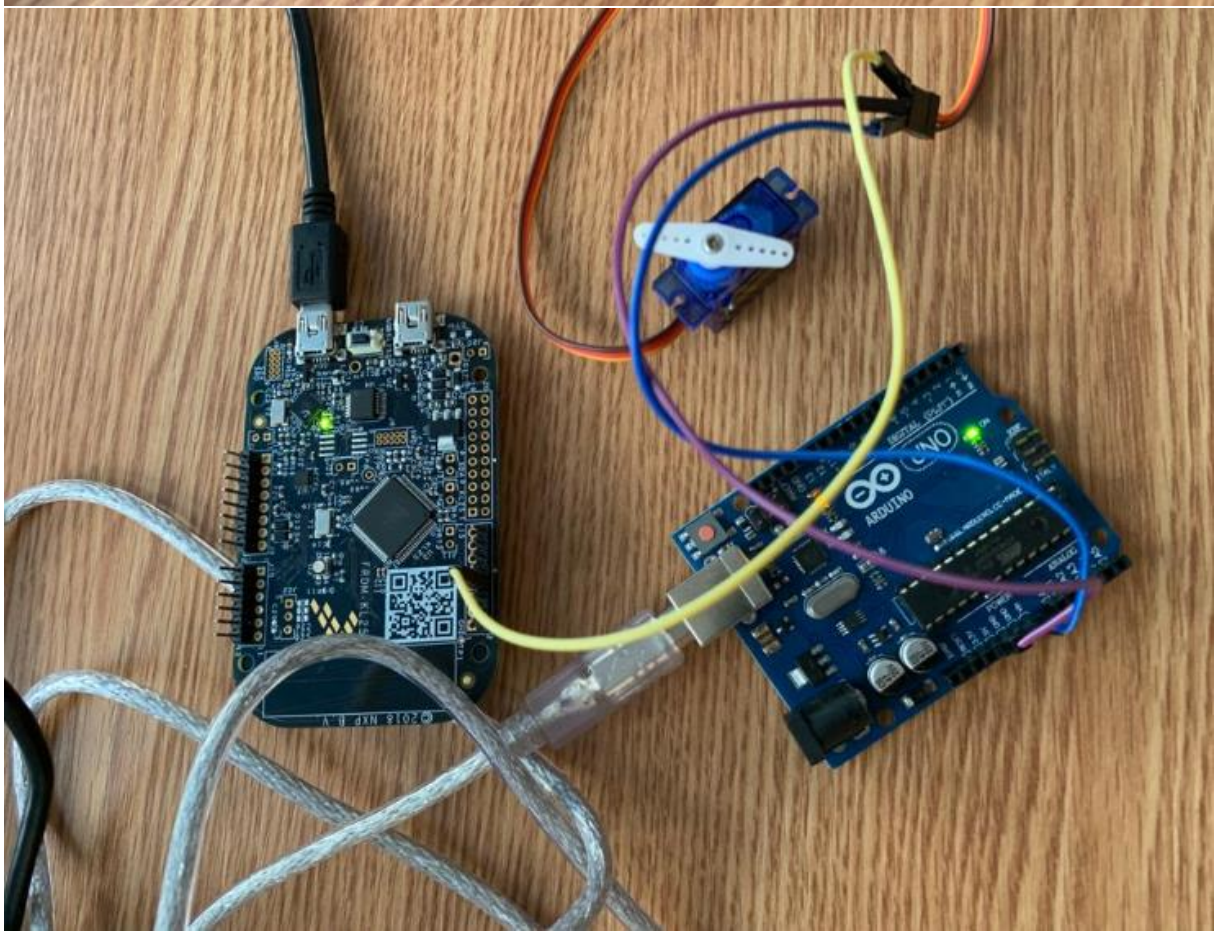
- Firul maroniu – galben conectează servomotorul la PTA5 (PWM)
- Firul roșu – mov conectează servomotorul la 5V
- Firul negru – albastru conectează servomotorul la GND

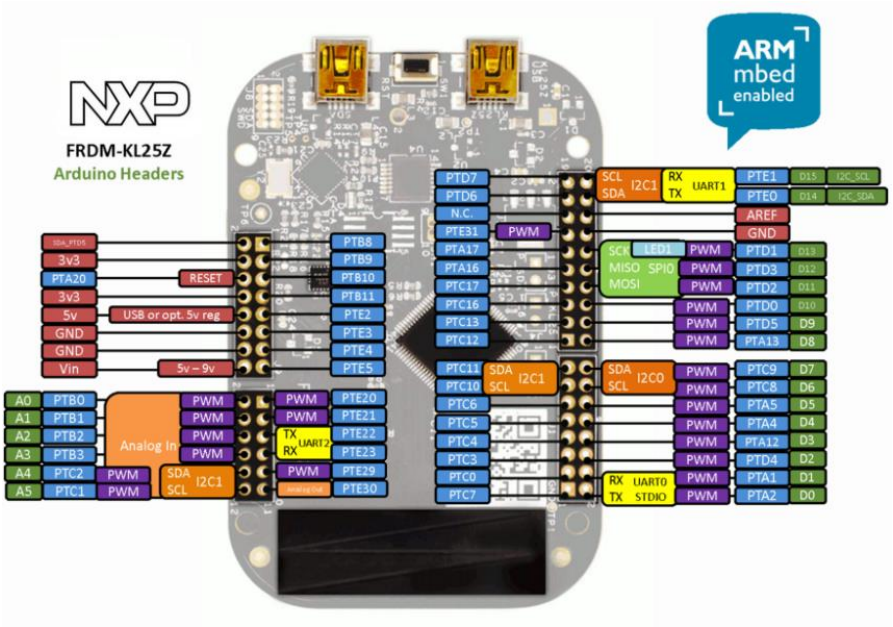


Este necesar să facem următoarea precizare:

Deoarece servomotorul funcționează la 5V (4.8V – 6V), iar plăcuța ***FRDM-KL25Z*** poate furniza o tensiune de 3.3V a fost necesară conectarea servomotorului la o plăcuță ARDUINO care putea furniza tensiunea necesară. Astfel, firul roșu și cel negru ale servomotorului au fost conectate la voltaj și ground pe arduino, iar firul pentru PWM a fost conectat pe plăcuța FRDM-KL25Z.


ARDUINO


ARDUINO



4. Descriere program

4.1. Funcția main

În fișierul main.c am inclus fișierele header în care sunt declarate funcții și variabile ce urmează a fi folosite: *ClockSettings.h* (funcțiile SystemClock_Configure, SystemClockTick_Configure și variabila flag_100ms), *Uart.h* (funcțiile UART0_Init și UTILS_PrintCounter), *Pwm.h* (funcțiile TPM0_Init și Signal_Control), *TouchCapacitiv.h* (funcțiile Touch_Init și Touch_Scan_LH) și fișierul header generat de către mediul de dezvoltare Keil, *MKL25Z4.h*, specific plăcii de dezvoltare.

```c
#include "TouchCapacitiv.h"
#include "Pwm.h"
#include "Uart.h"
#include "ClockSettings.h"
#include "MKL25Z4.h"

int main()
{

  SystemClock_Configure();
  SystemClockTick_Configure();
  UART0_Init(115200);
  Touch_Init();
  TPM0_Init();

  for(;;){
    if(flag_100ms){
      Signal_Control();
      flag_100ms = 0U;
    }
  }
  return 0;
}
```

Logica principală a programului este următoarea: întâi se apelează funcția de inițializare a timerelor și modulelor, apoi, într-un ciclu infinit, se verifică trecerea a 150 ms (când se va trimite valoarea de output a senzorului TSI prin UART, valoare în funcție de care se va modifica și unghiul de rotație al servomotorului).

4.2. Inițializarea modulelor

Prin apelul funcției init, sunt apelate funcții ce configurează anumite module necesare, ce vor fi utilizate. Vom folosi numeroase valori hardcodate, după cum urmează:

- frecvența de ceas a sistemului, folosită în calcule; valoarea acesteia este de 48 MHz în ClockSettings.c
- la care vom seta întreruperea de ceas, 1 ms în ClockSettings.c
- frecvența asociată intervalului de întrerupere (1/1ms = 1 * 1000 = KHz) în ClockSettings.c
- valoarea de resetare a timerului, raportul dintre cele două frecvențe – 1 (numărarea începe de la 0), în ClockSettings.c (48000000UL / 1000 - 1UL)

4.2.1. Configurarea ceasului

Modulul MCG (Multipurpose Clock Generator) oferă semnale de ceas pentru unitatea centrală. Acesta conține un FLL (frequency-locked loop), controlabil de către un ceas de referință intern.

```
10 void SystemClock_Configure(void) {
11
12
13   // MCGOUTCLOCK are ca si iesire, selectia FLL
14   MCG->C1 |= MCG_C1_CLKS(0);
15
16   // Selectarea ceasului intern ca si referinta pentru sursa FLL
17   MCG->C1 |= MCG_C1_IREFS_MASK;
18
19   // Setarea frecventei ceasului digital la 48 Mhz
20   // MCG->C4[DRST_DRS] = 1
21   // MCG->C4[DMX32] = 1
22   MCG->C4 |= MCG_C4_DRST_DRS(1);
23   MCG->C4 |= MCG_C4_DMX32(1);
24
25 }
26
```

În MCG_C1 (MCG Control 1 Register) am setat câmpul CLKS (Clock Source Select, biții 7-6) pe valoarea 0b00 pentru selectarea FLL/PLL ca output.

24.3.1 MCG Control 1 Register (MCG_C1)

Address: 4006_4000h base + 0h offset = 4006_4000h

Bit	7	6	5	4	3	2	1	0
Read / Write	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

MCG_C1 field descriptions

Field	Description
7–6 CLKS	Clock Source Select Selects the clock source for MCGOUTCLK . 00 Encoding 0 — Output of FLL or PLL is selected (depends on PLLS control bit). 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Reserved.

Cel folosit va fi selectat prin câmpul IREFS (Internal Reference Select, bitul 2) pe valoarea 1 – se selectează ceasul intern de referință, așadar se folosește FLL (PLL – phase-locked loop este configurabil doar extern).

2 IREFS	Internal Reference Select Selects the reference clock source for the FLL. 0 External reference clock is selected. 1 The slow internal reference clock is selected.

În MCG_C4 (MCG Control 4 Register), setăm câmpul DRST DRS (biții 6-5) pe valoarea 0b01 – se selectează frecvența DCO (Digitally-controlled oscillator) ca fiind una medie (40 – 50 MHz).

MCG_C4 field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>7
DMX32</td><td>DCO Maximum Frequency with 32.768 kHz Reference

The DMX32 bit controls whether the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference.

The following table identifies settings for the DCO frequency range.

NOTE: The system clocks derived from this source should not exceed their specified maximums.

<table>
<tr><th>DRST_DRS</th><th>DMX32</th><th>Reference Range</th><th>FLL Factor</th><th>DCO Range</th></tr>
<tr><td rowspan="2">00</td><td>0</td><td>31.25–39.0625 kHz</td><td>640</td><td>20–25 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>732</td><td>24 MHz</td></tr>
<tr><td rowspan="2">01</td><td>0</td><td>31.25–39.0625 kHz</td><td>1280</td><td>40–50 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>1464</td><td>48 MHz</td></tr>
<tr><td rowspan="2">10</td><td>0</td><td>31.25–39.0625 kHz</td><td>1920</td><td>60–75 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>2197</td><td>72 MHz</td></tr>
<tr><td rowspan="2">11</td><td>0</td><td>31.25–39.0625 kHz</td><td>2560</td><td>80–100 MHz</td></tr>
<tr><td>1</td><td>32.768 kHz</td><td>2929</td><td>96 MHz</td></tr>
</table>
0 DCO has a default range of 25%.
1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference.</td></tr>
<tr><td>6–5
DRST_DRS</td><td>DCO Range Select

The DRS bits select the frequency range for the FLL output, DCOOUT. When the LP bit is set, writes to the DRS bits are ignored. The DRST read field indicates the current frequency range for DCOOUT. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. See the DCO Frequency Range table for more details.

00 Encoding 0 — Low range (reset default).
01 Encoding 1 — Mid range.
10 Encoding 2 — Mid-high range.
11 Encoding 3 — High range.</td></tr>
</table>

În același registru setăm câmpul DMX32 (DCO Maximum Frequency, bitul 7) pe valoarea 1 – DCO va avea frecvența de 48 MHz. [2]

4.2.2. Configurarea întreruperii de ceas

```
27  void SystemClockTick_Configure(void){
28
29    // Deoarece imi doresc ca frecventa de generare a semnalului PWM sa tina cont de tick-uri de 1 ms,
30    // trebuie realizate urmatoarele configurari
31    // 48.000.000 pulsuri ..... 1 secunda
32    //      x      pulsuri ..... 10^(-3) secunde
33    //
34    SysTick->LOAD = (uint32_t)(48000000UL / 1000 - 1UL);
35    NVIC_SetPriority(SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL);
36    // Resetarea numaratorului digital
37    SysTick->VAL = 0UL;
38    SysTick->CTRL |= (SysTick_CTRL_CLKSOURCE_Msk | // Ceasul procesorului utilizat ca referinta (48Mhz)
39                      SysTick_CTRL_TICKINT_Msk | // Atunci cand numaratorul ajunge la 0, va porni handler-ul de intrerupere
40                      SysTick_CTRL_ENABLE_Msk); // Incarca valoarea pusa in registrul RELOAD si incepe numaratoarea
41  }
42
```

Setăm valoarea registrului de reload (STK_LOAD) cu valoarea de resetare a timerului, raportul dintre cele două frecvențe, care specifică intervalul de numărare al counterului.

4.5.2 SysTick reload value register (STK_LOAD)

Address offset: 0x04

Reset value: 0x0000 0000

Required privilege: Privileged

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RELOAD[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept cleared.

Bits 23:0 **RELOAD[23:0]:** RELOAD value

The LOAD register specifies the start value to load into the VAL register when the counter is enabled and when it reaches 0.

Calculating the RELOAD value

The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

The RELOAD value is calculated according to its use:

- To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.
- To deliver a single SysTick interrupt after a delay of N processor clock cycles, use a RELOAD of value N. For example, if a SysTick interrupt is required after 400 clock pulses, set RELOAD to 400.

Setăm prioritatea întreruperii SysTick pe valoarea maximă și valoarea inițială pe 0. În registrul STK_CTRL (registrul de control și stare SysTick) setăm biții 2 (CLKSOURCE – selectează ca sursă ceasul procesorului), 1 (TICKINT – activează întreruperea la valoarea 0) și 0 (ENABLE – pornește timerul).

4.5.1 SysTick control and status register (STK_CTRL)

Address offset: 0x00

Reset value: 0x0000 0000

Required privilege: Privileged

The SysTick CTRL register enables the SysTick features.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															COUNT FLAG
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CLKSO URCE	TICK INT	EN ABLE
													rw	rw	rw

Bits 31:17 Reserved, must be kept cleared.

Bit 16 COUNTFLAG:
Returns 1 if timer counted to 0 since last time this was read.

Bits 15:3 Reserved, must be kept cleared.

Bit 2 **CLKSOURCE:** Clock source selection
Selects the clock source.
0: AHB/8
1: Processor clock (AHB)

Bit 1 **TICKINT**: SysTick exception request enable
0: Counting down to zero does not assert the SysTick exception request
1: Counting down to zero to asserts the SysTick exception request.
Note: Software can use COUNTFLAG to determine if SysTick has ever counted to zero.

Bit 0 **ENABLE**: Counter enable
Enables the counter. When ENABLE is set to 1, the counter loads the RELOAD value from the LOAD register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.
0: Counter disabled
1: Counter enabled

4.2.3. Inițializarea modulului UART

Vom folosi modulul UART0 pentru comunicația serială cu PC prin cablul USB.

```
21  void UART0_Init(uint32_t baud_rate)
22  {
23
24    //Setarea sursei de ceas pentru modulul UART
25    SIM->SOPT2 |= SIM_SOPT2_UART0SRC(01);
26
27    //Activarea semnalului de ceas pentru modulul UART
28    SIM->SCGC4 |= SIM_SCGC4_UART0_MASK;
29
30    //Activarea semnalului de ceas pentru portul A
31    //intrucat dorim sa folosim pinii PTA1, respectiv PTA2 pentru comunicarea UART
32    SIM->SCGC5 |= SIM_SCGC5_PORTA_MASK;
33
34    //Fiecare pin pune la dispozitie mai multe functionalitati
35    //la care avem acces prin intermediul multiplexarii
36    PORTA->PCR[1] = ~PORT_PCR_MUX_MASK;
37    PORTA->PCR[1] = PORT_PCR_ISF_MASK | PORT_PCR_MUX(2); // Configurare RX pentru UART0
38    PORTA->PCR[2] = ~PORT_PCR_MUX_MASK;
39    PORTA->PCR[2] = PORT_PCR_ISF_MASK | PORT_PCR_MUX(2); // Configurare TX pentru UART0
40
41
42
43    UART0->C2 &= ~((UART0_C2_RE_MASK) | (UART0_C2_TE_MASK));
44
45    //Configurare Baud Rate
46    uint32_t osr = 15; // Over-Sampling Rate (numarul de esantioane luate per bit-time)
47
48    //SBR - vom retine valoarea baud rate-ului calculat pe baza frecventei ceasului de sistem
49    //   SBR  -   b16 b15 b14 [b13 b12 b11 b10 b09    b08 b07 b06 b05 b04 b03 b02 b01] &
50    // 0x1F00 -   0   0   0    1   1   1   1   1      0   0   0   0   0   0   0   0
51    //            0   0   0    b13 b12 b11 b10 b09    0   0   0   0   0   0   0   0 >> 8
52    //   BDH  -   0   0   0    b13 b12 b11 b10 b09
53    //   BDL  -   b08 b07 b06  b05 b04 b03 b02 b01
54
55
56    uint32_t sbr = 48000000UL / ((osr + 1)*baud_rate);
57    uint8_t temp = UART0->BDH & ~(UART0_BDH_SBR(0x1F));
58    UART0->BDH = temp | UART0_BDH_SBR(((sbr & 0x1F00)>> 8));
59    UART0->BDL = (uint8_t)(sbr & UART_BDL_SBR_MASK);
60    UART0->C4 |= UART0_C4_OSR(osr);
61
62    //Setare numarul de biti de date la 8 si fara bit de paritate
63    UART0->C1 = 0;
64
65    //Dezactivare intreruperi la transmisie
66    UART0->C2 |= UART0_C2_TIE(0);
67    UART0->C2 |= UART0_C2_TCIE(0);
68
69    //Activare intreruperi la receptie
70    UART0->C2 |= UART0_C2_RIE(1);
71
72    //Activare intreruperi REceptie TransmisiE
73    UART0->C2 |= ((UART_C2_RE_MASK) | (UART_C2_TE_MASK));
74
75    NVIC_EnableIRQ(UART0_IRQn);
76  }
```

În registrul SIM_SOPT2 (System Options Register 2) setăm pe 0b01 câmpul UART0SRC (biții 27-26) pentru selectarea ca sursă de ceas a modulului MCGFLLCLK anterior configurat,

27–26 UART0SRC	UART0 clock source select Selects the clock source for the UART0 transmit and receive clock. 00 Clock disabled 01 MCGFLLCLK clock or MCGPLLCLK/2 clock 10 OSCERCLK clock 11 MCGIRCLK clock

În registrul SIM_SCGC4 (System Clock Gating Control Register 4) setăm pe 1 câmpul UART0 (bitul 10) pentru activarea ceasului pentru acest modul, folosind masca

SIM_SCGC4_UART0_MASK. Masca are valoarea 0x400, adica 1024 în zecimal, adică are setat pe 1 al 10-lea bit (UART0).

```
#define SIM_SCGC4_UART0_MASK                    0x400u
```

10 UART0	UART0 Clock Gate Control This bit controls the clock gate to the UART0 module. 0 Clock disabled 1 Clock enabled

În registrul SIM_SCGC5 (System Clock Gating Control Register 5) setăm pe 1 câmpul PORTA (bitul 9) pentru activarea ceasului acestui port, folosind masca SIM_SCGC5_PORTA_MASK. Masca are valoarea 0x200, adica 512 în zecimal, adică are setat pe 1 al 9-lea bit (PORTA).

```
#define SIM_SCGC5_PORTA_MASK                    0x200u
```

9 PORTA	Port A Clock Gate Control This bit controls the clock gate to the Port A module. 0 Clock disabled 1 Clock enabled

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).

PTA1	27	TSI0_CH2	PTA1	UART0_RX
PTA2	28	TSI0_CH3	PTA2	UART0_TX

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read Write	LBKDIE	RXEDGIE	SBNS	SBR				
Reset	0	0	0	0	0	0	0	0

UARTx_BDH field descriptions

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	SBR							
Reset	0	0	0	0	0	1	0	0

UARTx_BDL field descriptions

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

4–0 SBR	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the baud rate generator. When BR is 1 - 8191, the baud rate equals baud clock / ((OSR+1) × BR).

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

4–0 OSR	Over Sampling Ratio This field configures the oversampling ratio for the receiver between 4x (00011) and 32x (11111). Writing an invalid oversampling ratio will default to an oversampling ratio of 16 (01111). This field should only be changed when the transmitter and receiver are both disabled.

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

1 PE	Parity Enable Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

4 M	9-Bit or 8-Bit Mode Select 0 Receiver and transmitter use 8-bit data characters. 1 Receiver and transmitter use 9-bit data characters.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

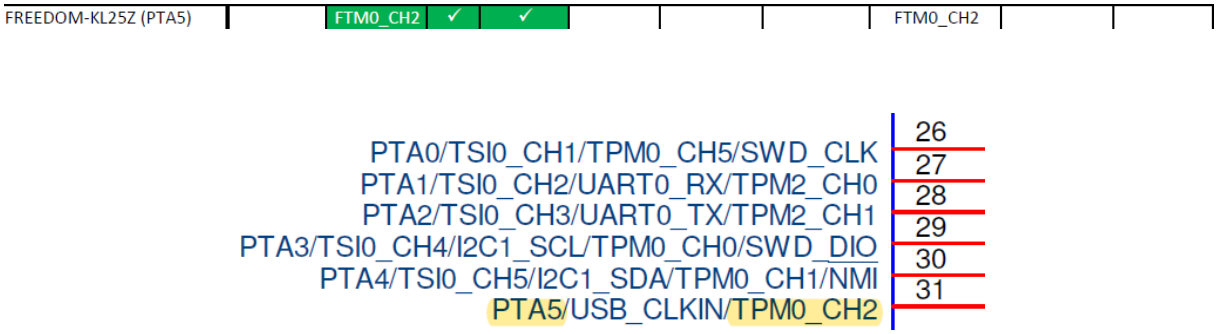
Activăm transmiterea prin UART prin setarea câmpului TE (bitul 3, Transmitter Enable) din UART0_C2 pe valoarea 1 cu ajutorul măștii UART_C2_TE_MASK ce are valoarea 0x8u, și activăm, de asemenea, recepția prin UART prin setarea câmpului RE (bitul 2, Receiver Enable) din UART0_C2 pe valoarea 1 cu ajutorul măștii UART_C2_RE_MASK ce are valoarea 0x4u.

3 TE	Transmitter Enable TE must be 1 to use the UART transmitter. When TE is set, the UART forces the UART _TX pin to act as an output for the UART system. When the UART is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single UART communication line (UART _TX pin). TE can also queue an idle character by clearing TE then setting TE while a transmission is in progress. When TE is written to 0, the transmitter keeps control of the port UART _TX pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to tristate. 0 Transmitter disabled. 1 Transmitter enabled.
2 RE	Receiver Enable When the UART receiver is off or LOOPS is set, the UART _RX pin is not used by the UART . When RE is written to 0, the receiver finishes receiving the current character (if any). 0 Receiver disabled. 1 Receiver enabled.

4.2.4. Inițializarea modulului TPM

Vom folosi modulul TPM (Timer/PWM Module) pentru generarea semnalului PWM.

Deoarece modulul UART a fost configurat pe pinii PTA0 și PTA1 va trebui să alegem un altul pentru servomotor. Am ales PTA5. Cum TPM2 poate fi asociat doar cu PTA1 și PTA2 dintre pinii disponibili fizic pe plăcuța noastră, am ales să folosim TPM0, care se poate folosi împreună cu PTA5.



```c
void TPM0_Init(){


  // Activarea semnalului de ceas pentru utilizarea LED-ului de culoare rosie
  SIM->SCGC5 |= SIM_SCGC5_PORTA_MASK;

  // Utilizarea alternativei de functionare pentru perifericul TMP
  // TMP2_CH0
  PORTA->PCR[OSCILLOSCOPE_PIN] |= PORT_PCR_MUX(3);

  // Selects the clock source for the TPM counter clock (MCGFLLCLK) - PG. 196
  // MCGFLLCLK Freq. - 48 MHz
  SIM->SOPT2 |= SIM_SOPT2_TPMSRC(1);

  // Activarea semnalului de ceas pentru modulul TPM
  SIM->SCGC6 |= SIM_SCGC6_TPM0(1);

  // Divizor de frecventa pentru ceasul de intrare
  // PWM CLK -> 48MHz / 128 = 48.000.000 / 128 [Hz] = 375.000 [Hz] = 375 kHz
  TPM0->SC |= TPM_SC_PS(7);

    // LPTPM counter operates in up counting mode. - PG. 553
  // Selects edge aligned PWM
  TPM0->SC |= TPM_SC_CPWMS(0);

  // LPTPM counter increments on every LPTPM counter clock
  TPM0->SC |= TPM_SC_CMOD(1);
```

```c


  // LPTPM counter operates in up-down counting mode. - PG. 553
  // Selects center aligned PWM
  //TPM2->SC |= TPM_SC_CPWMS(1);


  // Edge-Aligned Pulse Width Modulation
  // TPM->SC[CPWMS] = 0

  // ===== Mode selection ====
  // Configured for EPWM
  // TPM->CnSC[MnSB] = 1
  // TPM->CnSC[MnSB] = 0

  // ==== Edge selection ====
  // Set output on counter overflow, clear output on match
  // Counter overflow = LPTPM counter reaches the value configured in the MOD register, and then resets
  // Match = LPTPM counter reaches the value configured in the CnV register

  // TPM->CnSC[ELSnB] = 1
  // TPM->CnSC[ELSnA] = 0

  // Regiter associated to the control of channel 0
  // Why channel 0?
  TPM0->CONTROLS[2].CnSC |= (TPM_CnSC_MSB_MASK | TPM_CnSC_ELSB_MASK);
}
```

Vom porni ceasul portului A, unde se află servomotorul conectat prin setarea bitului al 9-lea al registrului SIM_SCGC5 pe valoarea 1 cu ajut0rul măștii SIM_SCGC5_PORTA_MASK ce are valoarea 0x200.

9 PORTA	Port A Clock Gate Control This bit controls the clock gate to the Port A module. 0 Clock disabled 1 Clock enabled

Setăm multiplexarea pinului 5 din portul A pe canalul 2 al modulului 0 TPM, prin setarea câmpului MUX pe valoarea 0b011.

PTA5	31		PTA5	USB_CLKIN	FTM0_CH2

Setăm FLL ca sursă de ceas prin setarea câmpului TPMSRC (biții 25-24, TPM clock source select) din registrul System Options Register 2 (SIM_SOPT0) pe valoarea 0b01.

25–24 TPMSRC	TPM clock source select Selects the clock source for the TPM counter clock 00 Clock disabled 01 MCGFLLCLK clock or MCGPLLCLK/2 10 OSCERCLK clock 11 MCGIRCLK clock

Activăm ceasul modului 2 TPM prin setarea câmpului TPM0 (bitul 24) din SIM_SCGC6 pe 1.

24 TPM0	TPM0 Clock Gate Control This bit controls the clock gate to the TPM0 module. 0 Clock disabled 1 Clock enabled

Setăm semnalul PWM generat ca fiind edge-aligned, setând counterul în mod up counting prin valoarea 0 în câmpul CPWMS (bitul 5, Center-aligned PWM Select) și valoarea 0b01 în câmpul CMOD (biții 4-3, Clock Mode Selection) pentru incrementarea numărătorului LPTPM.

5 CPWMS	Center-aligned PWM Select Selects CPWM mode. This mode configures the LPTPM to operate in up-down counting mode. This field is write protected. It can be written only when the counter is disabled. 0 LPTPM counter operates in up counting mode. 1 LPTPM counter operates in up-down counting mode.
4–3 CMOD	Clock Mode Selection Selects the LPTPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the LPTPM clock domain. 00 LPTPM counter is disabled 01 LPTPM counter increments on every LPTPM counter clock 10 LPTPM counter increments on rising edge of LPTPM_EXTCLK synchronized to the LPTPM counter clock 11 Reserved

Prin setarea câmpului PS (biții 2-0, Prescale Factor Selection) pe valoarea 0b111, ceasul sistemului va fi divizat cu 128, deci ceasul PWM va avea frecvența de 375 KHz.

2–0 PS	Prescale Factor Selection Selects one of 8 division factors for the clock mode selected by CMOD. This field is write protected. It can be written only when the counter is disabled. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

4.2.5. Inițializarea modulului pentru senzorul TSI

```
5  void Touch_Init()
6  {
7    // Enable clock for TSI PortB 16 and 17
8    SIM->SCGC5 |= SIM_SCGC5_TSI_MASK;
9
10
11   TSI0->GENCS = TSI_GENCS_OUTRGF_MASK |  // Out of range flag, set to 1 to clear
12                 //TSI_GENCS_ESOR_MASK |  // This is disabled to give an interrupt when out of range.
13                 //Enable to give an interrupt when end of scan
14                 TSI_GENCS_MODE(0u) |  // Set at 0 for capacitive sensing.
15                 //Other settings are 4 and 8 for threshold detection, and 12 for noise detection
16                 TSI_GENCS_REFCHRG(0u) | // 0-7 for Reference charge
17                 TSI_GENCS_DVOLT(0u) | // 0-3 sets the Voltage range
18                 TSI_GENCS_EXTCHRG(0u) | //0-7 for External charge
19                 TSI_GENCS_PS(0u) | // 0-7 for electrode prescaler
20                 TSI_GENCS_NSCN(31u) | // 0-31 + 1 for number of scans per electrode
21                 TSI_GENCS_TSIEN_MASK | // TSI enable bit
22                 //TSI_GENCS_TSIIEN_MASK | //TSI interrupt is disables
23                 TSI_GENCS_STPE_MASK | // Enables TSI in low power mode
24                 //TSI_GENCS_STM_MASK | // 0 for software trigger, 1 for hardware trigger
25                 //TSI_GENCS_SCNIP_MASK | // scan in progress flag
26                 TSI_GENCS_EOSF_MASK ; // End of scan flag, set to 1 to clear
27                 //TSI_GENCS_CURSW_MASK; // Do not swap current sources
28
29 }
30
```

Cu ajutorul măștii SIM_SCGC5_TSI_MASK, vom seta pe 1 bitul al 5-lea al SIM_SCGC5 pentru a activa accesul la modulul TSI.

5 TSI	TSI Access Control This bit controls software access to the TSI module. 0 Access disabled 1 Access enabled

Cu ajutorul măștilor corespunzătoare, setăm toți biții necesari bunei funcționări a modulului TSI – out of range flag, status bit, sarcină de referință, intervalul de tensiune, sarcina externă, electrode prescaler, numărul de scanări per electrode, TSI enable bit, TSI low power mode enabler. End of scan flag.

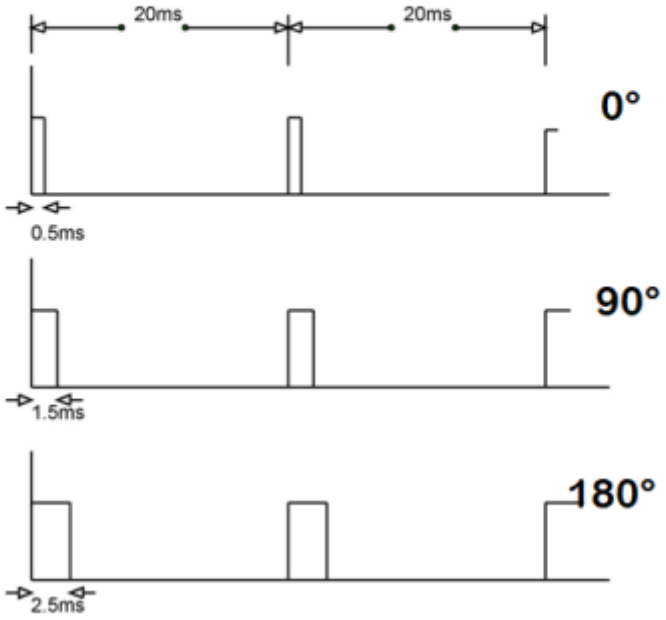
4.3. Generare PWM

După cum am menționat, în funcția main, atunci când flagul ce marchează trecerea a 150 ms devine 1 după incrementarea acestuia în funcția ce se ocupă de întreruperea de ceas, SysTick_Handler, este apelată funcția Signal_Control. În interiorul acestei funcții se preia inputul de la senzorul TSI, o valoare între 0 și 1757, se transmite valoarea prin modulul UART,

```
61  void Signal_Control(void){
62
63
64    uint8_t from_tsi = Touch_Scan_LH();
65    UTILS_PrintCounter(from_tsi);
66    UART0_Transmit(0x0A);
67    UART0_Transmit(0x0D);
68
69
70    // Resetarea valorii numaratorului asociat LPTPM Counter
71    TPM0->CNT = 0x0000;
72
73    // Setarea perioadei semnalului PWM generat
74    TPM0->MOD = 375 * 20;
75
76    // Setarea duty cycle-ului asociat semnalului PWM generat
77    TPM0->CONTROLS[2].CnV = (375 * 20 * from_tsi) / (1757) + 37*5;
78
79
80
81  }
```

La fiecare 150 ms, în funcție de raportul calculat, modificăm regiștrii: TPM2_CNT (resetăm numărătorul), TPM2_MOD (valoarea până la care trebuie să numere pentru a termina o perioadă – frecvența ceasului * perioada semnalului / 1000 (pentru conversia din ms) - 375 * 20). Pentru a determina factorul de umplere, este necesar să cunoaștem timpii asociați unghiurilor de rotație pentru servomotor. Astfel, pentru 0 grade, nu se pleacă de la 0 ms, ci de la 0.5 ms. Din cauza acestui termen liber, va trebui sa reducem intervalul [0.5 ms – 2.5 ms] la [0 ms – 2 ms] pentru a putea calcula valorile proporțional. La final, vom adăuga prin însumare trunchierea.

- Servomotorul TowerPro MG995
  - Frecvența semnalului PWM (50Hz)
  - Factorul de umplere (determină poziția axei servo)



Valoarea factorului de umplere va fi 375 (valoarea frecvenței), înmulțită cu 20 (corespunzător pentru 2 ms), înmulțit cu valoarea primită de la senzorul TSI pe care dorim să o aducem în limitele acceptate de servomotor, totul împărțit la 1757 (valoarea maximă a senzorului TSI), rezultat la care vom adăuga valoarea scăzută inițial pentru a lucra proporțional, și anume 375 * 5 / 10, adică 37 * 5.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 COUNT	Counter value

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																MOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

TPMx_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 MOD	Modulo value When writing this field, all bytes must be written at the same time.

Address: Base address + 10h offset + (8d × i), where i=0d to 5d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																VAL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_C*n*V field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 VAL	Channel Value Captured LPTPM counter value of the input modes or the match value for the output modes. When writing this field, all bytes must be written at the same time.

4.4. Transmitere date prin UART

O dată la 150 ms, când flagul corespunzător îndeplinește condiția din if, este apelată funcția UTILS_PrintCounter ce primește ca parametru valoarea primită de la senzorul TSI.

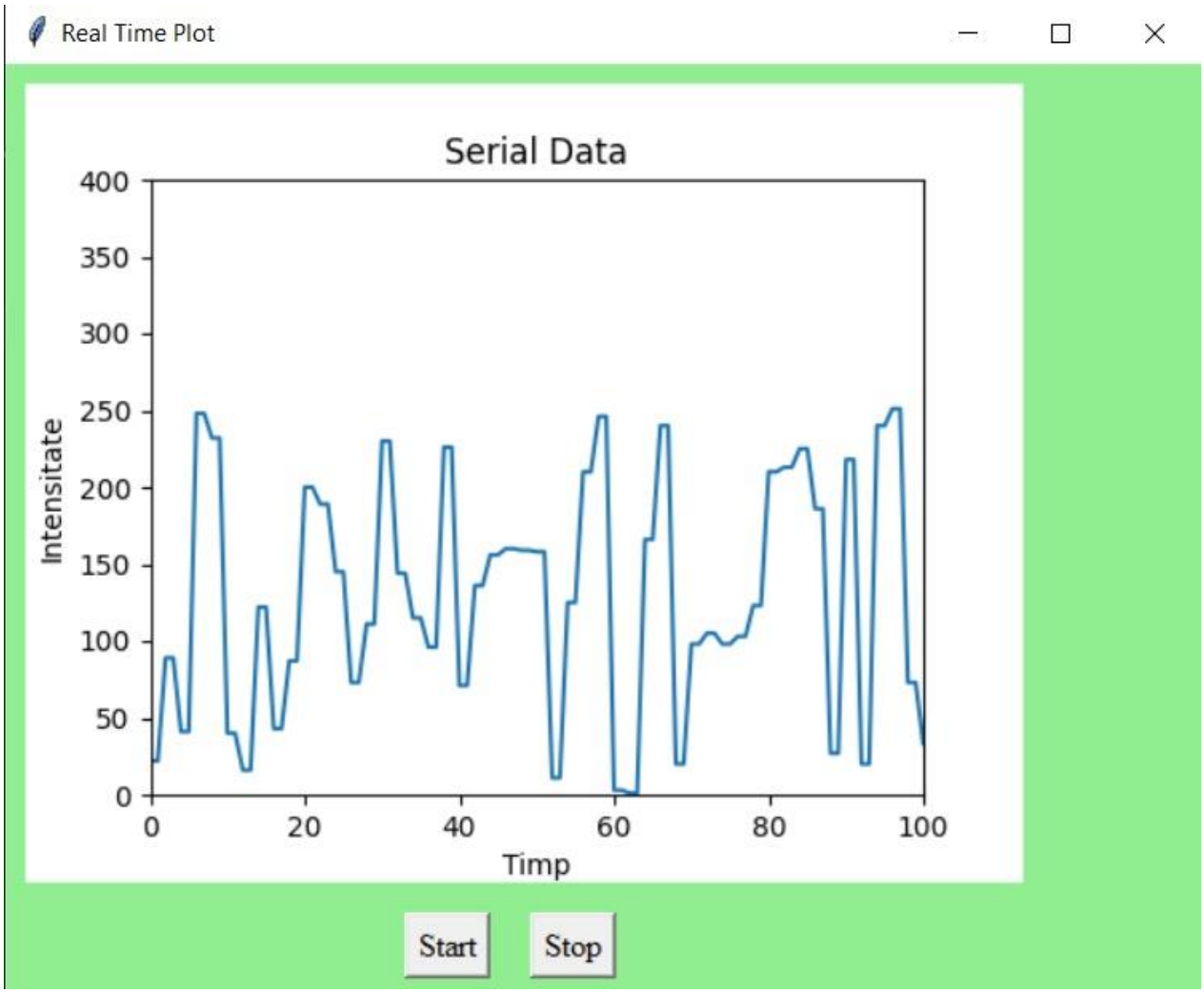
```
61  void Signal_Control(void){
62
63
64    uint8_t from_tsi = Touch_Scan_LH();
65    UTILS_PrintCounter(from_tsi);
66    UART0_Transmit(0x0A);
67    UART0_Transmit(0x0D);
68
69
70    // Resetarea valorii numaratorului asociat LPTPM Counter
71    TPM0->CNT = 0x0000;
72
73    // Setarea perioadei semnalului PWM generat
74    TPM0->MOD = 375 * 20;
75
76    // Setarea duty cycle-ului asociat semnalului PWM generat
77    TPM0->CONTROLS[2].CnV = (375 * 20 * from_tsi) / (1757) + 37*5; //
78
79
80
81  }
```

```
77  void UART0_IRQHandler(void) {
78
79      if(UART0->S1 & UART0_S1_RDRF_MASK) {
80        c = UART0->D;
81      }
82
83      if(c >= 'a' && c <= 'z') {
84        UART0_Transmit(c - 'a' + 'A');
85      } else if ( c >= 'A' && c <= 'Z') {
86        UART0_Transmit(c - 'A' + 'a');
87      } else if(c == 0X0D) {
88        UART0_Transmit(0X0D);
89        UART0_Transmit(0X0A);
90      }
91  }
92
93  void UTILS_PrintCounter(int count){
94    uint8_t Digit1=count / 10000;
95    uint8_t Digit2=count %10000 /1000;
96    uint8_t Digit3=count %1000 /100;
97    uint8_t Digit4=count %100/10;
98    uint8_t Digit5=count %10;
99    UART0_Transmit(Digit1+48);
100   UART0_Transmit(Digit2+48);
101   UART0_Transmit(Digit3+48);
102   UART0_Transmit(Digit4+48);
103   UART0_Transmit(Digit5+48);
104 }
```

Valoarea octetului dat ca parametru este pus în registrul UART0_D (UART Data Register). Nu se părăsește funcția până când câmpul TDRE (Transmit Data Register Empty Flag) din registrul UART0_S1 (UART Status Register 1) este 1, adică bufferul de trimitere s-a golit.

7 TDRE	Transmit Data Register Empty Flag TDRE is set out of reset and whenever there is room to write data to the transmit data buffer. To clear TDRE, write to the UART data register (UART _D). 0 Transmit data buffer full. 1 Transmit data buffer empty.

5. Rezultate matplotlib



6. Dificultăți întâmpinate

În timpul realizării acestui proiect, am întâmpinat unele dificultăți, precum:

- Folosirea inițială a pinului PTA1 (din considerente de a minimiza porturile folosite și de a grupa cablurile) pentru utilizarea PWM-ului în scopul rotirii servomotorului, însa pinul era ocupat de modulul UART, la fel ca si PTA2. Prin urmare, a fost nevoie să lucrăm cu TPM0 în loc de TPM2, deorece TPM2 comunica doar cu PTA1 și PTA2, iar TPM0 putea comunica și cu alți pini (PTA5).
- Stabilirea baud-rate-ului pentru modulul UART

7. Bibliografie

[1] https://www.dfrobot.com/product-255.html

[2] Freescale Semiconductor, Inc., KL25 Sub-Family Reference Manual, 2012.

[3] STM, STM32F10xxx/20xxx/21xxx/L1xxxx Cortex®-M3 programming manual, 2017.