Ceas deșteptător

Microcontrolere
Microcontrolere – arhitecturi și programare

Studenți:
Bulgariu Elena-Iuliana
Mihăilă Bogdan-Iulian

Suceava
2024

Cuprins

Cuprins

1. Noțiuni teoretice

In acest capitol veti descrie pe scurt (approx 1-2 pagini) arhitectura microcontrollerului (core si periferice) insistand pe acele periferice care vor fi folosite in proiect.

Arhitectura Microcontrollerului STM32F429:

Microcontrollerul STM32F429 face parte din familia STM32 dezvoltată de STMicroelectronics și se bazează pe arhitectura ARM Cortex-M4. Acesta oferă o combinație de performanță, eficiență energetică și versatilitate a perifericelor, fiind ideal pentru aplicații embedded.

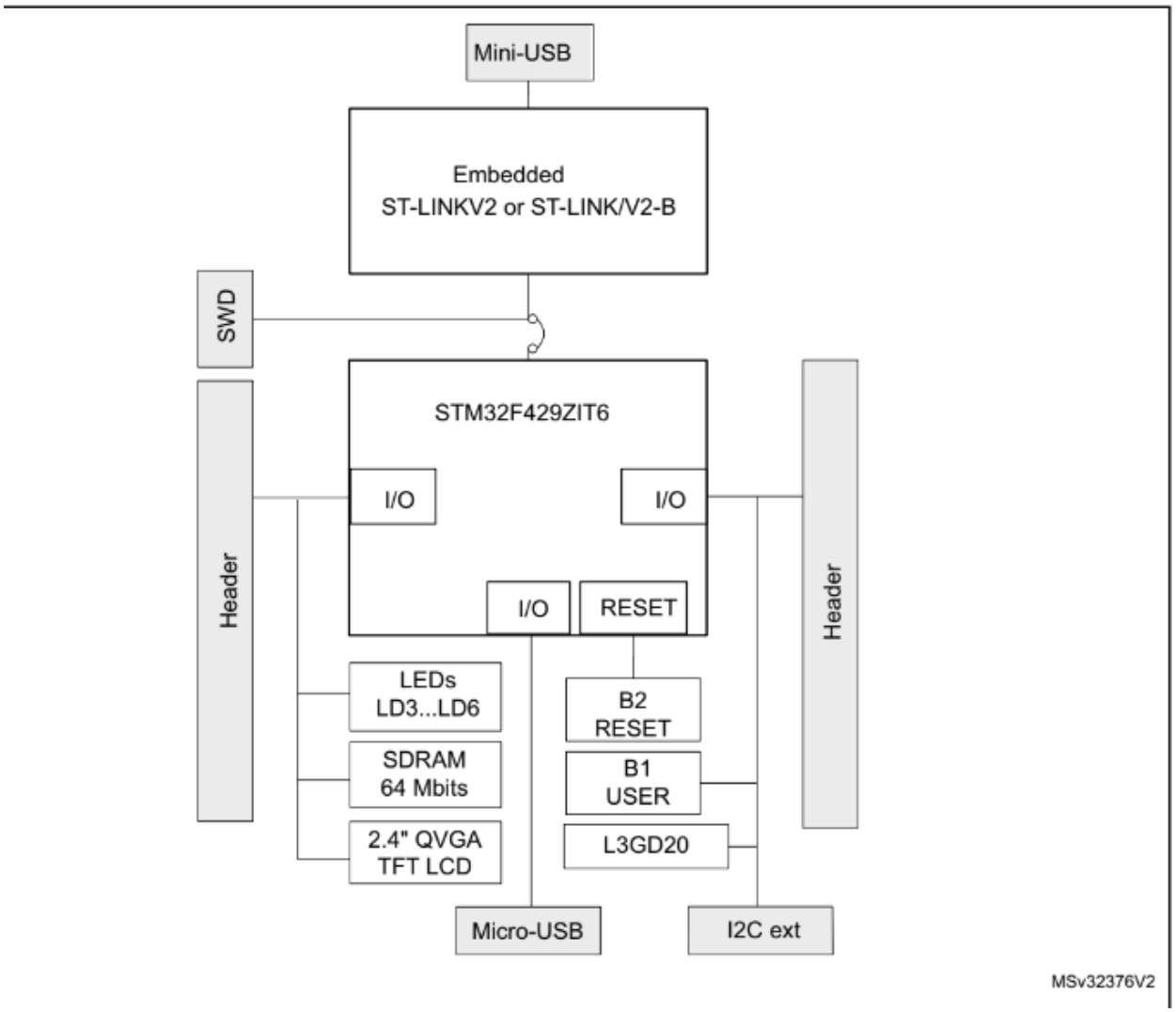


Figura 1. Hardware block diagram

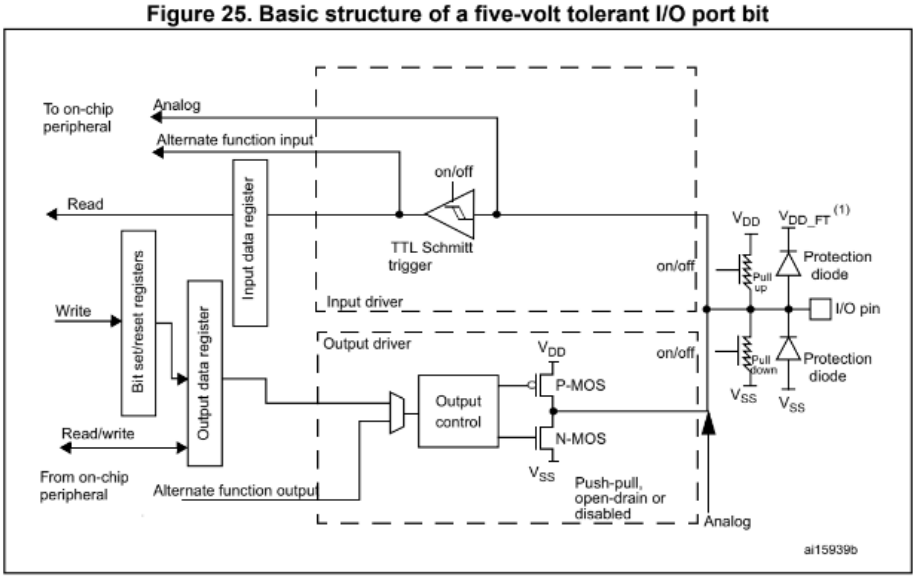
GPIO (General-Purpose I/O):

Porturile GPIO permit interacțiunea cu mediul exterior prin conectarea la diverse dispozitive și senzori.

Fiecare port de intrare/ieșire generală (GPIO) are patru registre de configurare de 32 de biți (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR și GPIOx_PUPDR), două registre de date de 32 de biți (GPIOx_IDR și GPIOx_ODR), un registru de setare/resetare de 32 de biți (GPIOx_BSRR), un registru de blocare de 32 de biți (GPIOx_LCKR) și două registre de selecție a funcției alternative de 32 de biți (GPIOx_AFRH și GPIOx_AFRL).

Cu condiția caracteristicilor hardware specifice ale fiecărui port I/O listate în fișa de date, fiecare bit de port al porturilor de intrare/ieșire generală (GPIO) poate fi configurat individual de către software în mai multe moduri

Figure 25. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 35. Port bit configuration table(1)

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 35. Port bit configuration table[1] (continued)

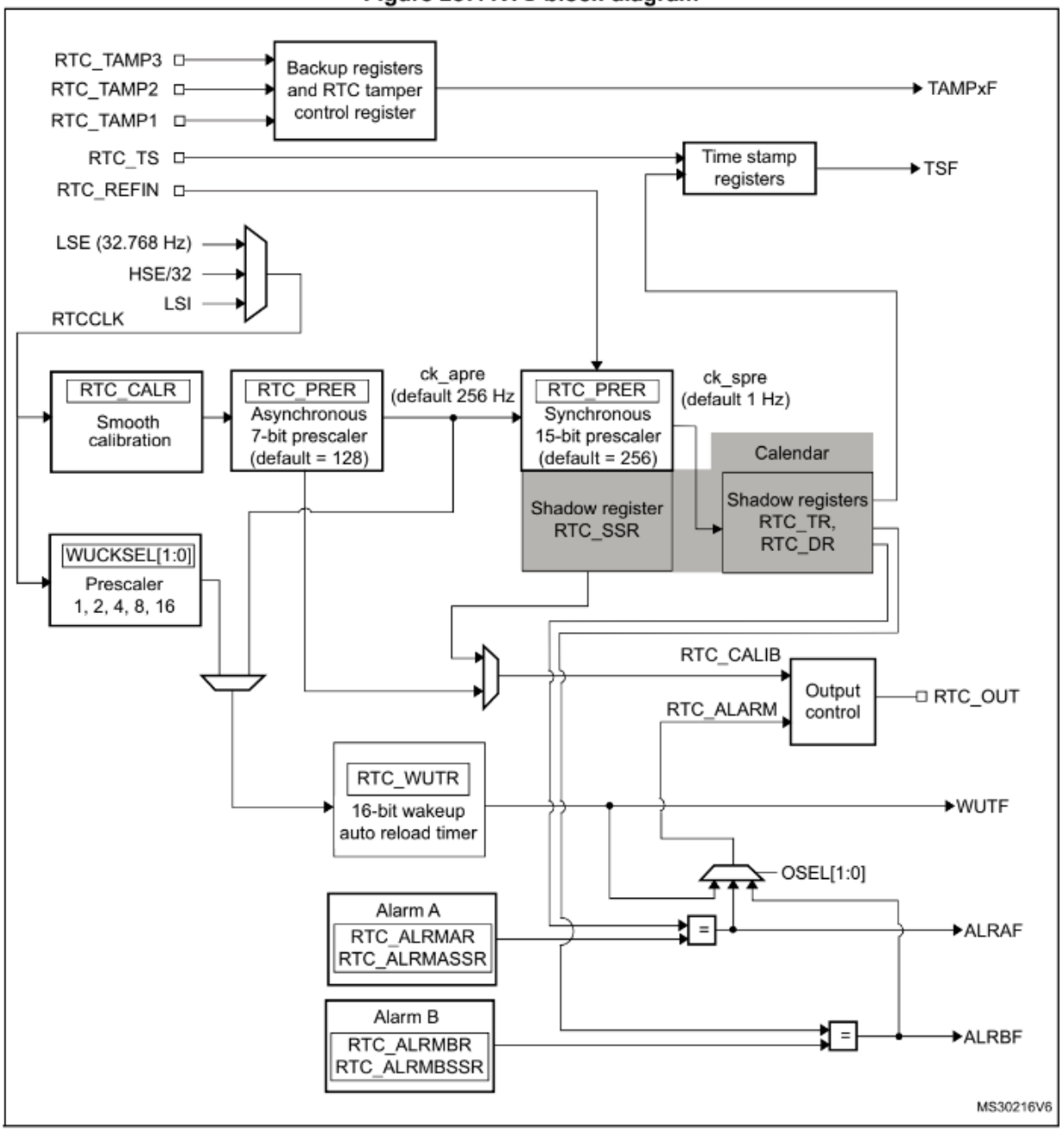
MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

RTC (Real-Time Clock):

Modulul RTC furnizează funcționalități de ceas în timp real, esențial în aplicații care necesită gestionarea precisă a timpului și a evenimentelor temporale.

Figure 237. RTC block diagram



1. On STM32F4xx devices, the RTC_AF1 and RTC_AF2 alternate functions are connected to PC13 and PI8, respectively.

26.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 804* and *Reading the calendar on page 805*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

26.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 804* and *Reading the calendar on page 805*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

26.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

26.6.21 RTC register map

Table 121. RTC register map and reset values

Offset	Register	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
0x00	**RTC_TR**	Reserved									PM	HT [1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Reserved								YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Reserved	DT [1:0]		DU[3:0]				
	Reset value																	0	0	1	0	0	0	0	1		0	0	0	0	0	0	1
0x08	**RTC_CR**	Reserved								COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WCKSEL [2:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ISR	Reserved																	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0x10	**RTC_PRER**	Reserved									PREDIV_A[6:0]							Reserved	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Reserved																WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CALIBR**	Reserved																								DCS	Reserved	DC[4:0]					
	Reset value																									0		0	0	0	0	0	

TIM (Timer):

Modulul Timer oferă funcționalități temporale și de generare a impulsurilor, esențiale pentru gestionarea evenimentelor periodice sau pentru controlul motoarelor.

Compatibilitate și Flexibilitate:

Microcontrollerul STM32F4xx face parte dintr-o familie mai largă de microcontrolere STM32, beneficiind de suport extins pentru dezvoltare și o comunitate activă. Folosind mediul de dezvoltare și instrumentele asociate, programatorii pot beneficia de flexibilitate în proiectarea și programarea aplicațiilor.

2. Descrierea temei alese

Proiectul se concentrează pe implementarea și utilizarea funcționalităților avansate ale modulului RTC disponibil pe microcontrolerele din seria STM32F429. Aceste funcționalități pot fi utile în aplicații care necesită gestionarea precisă a timpului, alarme programabile și capacitatea de a funcționa eficient în modurile de consum redus de energie. Implementarea corectă a acestor funcționalități depinde de configurarea adecvată a registrelor RTC și gestionarea corespunzătoare a evenimentelor și întreruperilor asociate.

Scopul temei propuse este de a dezvolta o aplicație pentru o placă STM32F429I, având ca principală funcționalitate implementarea unui ceas deșteptător. Aplicația permite programarea ceasului de timp real (RTC) în formatul HH:MM:SS printr-o interfață LCD+touch screen. Utilizatorul poate seta o alarmă pentru un moment viitor, iar la activarea acesteia, un buzzer emite un semnal sonor, semnalizând astfel alarma. Proiectul implică utilizarea perifericelor RTC, GPIO, și dezvoltarea unei interfețe utilizator intuitive pentru setarea timpului și a alarmelor.

3. Proiectarea aplicației

3.1 Specificarea cerințelor:

A. Cerințe Funcționale:

a. **Modulul RTC** va furniza timpul real pe ecranul principal al UI.

b. **Proiectarea Detaliată a UI:**

- **ECRAN PRINCIPAL:**

Afișarea a două butoane principale: "SET ALARM" și "SET TIME" si a orei.

- **SET ALARM:**

Apăsarea acestui buton va deschide o nouă interfață cu 3 butoane:

- "Incrementare minute"
- "Decrementare minute"
- "Salvare"

Permite utilizatorului să seteze ora alarmei.

Posibilitatea de a revizualiza setările alarmei.

- **SET TIME:**

Similar cu SET ALARM, permite setarea inițială a orei și, apoi, a minutelor după apăsarea butonului "SALVARE".

c. Proiectarea Modulului de Control al Alarmei:

Setarea Alarmei:

- După apăsarea butonului "Salvare" în interfața SET ALARM, timpul ales va fi transmis funcției principale de gestionare a alarmei.
- Un flag va deveni TRUE pentru a indica sistemului că o alarmă a fost setată.

Declanșarea Alarmei:

- După trecerea timpului setat de alarmă, buzzer-ul va fi activat, iar un semnal sonor va fi emis timp de 15 secunde.

Anularea Alarmei:

- Posibilitatea de a anula sau dezactiva alarma înainte ca aceasta să se declanșeze prin intermediul butonului SET ALARM.

B. Cerințe Non-funcționale:

1. **Performanță:**
   - Asigurarea unei performanțe adecvate în răspunsul la acțiunile utilizatorului și în manipularea datelor: Interfața utilizatorului trebuie să fie receptivă, cu un timp de răspuns sub un anumit prag (mai mic de 1 ms).
2. **Ușurința în Utilizare:**
   - Crearea unei interfețe cu utilizatorul intuitivă și ușor de înțeles: Utilizatorul trebuie să poată seta ora și alarmele fără a consulta manualul de utilizare.
3. **Fiabilitate și Disponibilitate:**
   - Asigurarea unei funcționări fiabile și disponibile a aplicației: Aplicația trebuie să fie disponibilă pentru utilizare 24/7, iar alarmele să funcționeze corect chiar și în condiții de instabilitate temporară a alimentării.

3.2 Analiza sistemului:

În cadrul analizei sistemului, ne vom concentra pe fiecare componentă a proiectului și vom detalia funcționalitățile, interacțiunile și necesitățile fiecărui modul.

1. **Modul de Gestionare a Timpului Real (RTC):**
    - **Funcționalități:**
        - Urmărirea continuă a timpului real.
        - Actualizarea constantă a orei curente.
    - **Interacțiuni:**
        - Comunicare cu modulul de setare a timpului pentru actualizări.
        - Furnizarea orei curente la cererea modulului de interfață cu utilizatorul.
2. **Modul de Interfață cu Utilizatorul (UI):**
    - **Funcționalități:**
        - Afișarea interfeței grafice pentru setarea orei și alarmelor.
        - Actualizarea modulului RTC cu noile valori introduse de utilizator.
        - Recepționarea și procesarea evenimentelor generate de utilizator prin intermediul ecranului tactil.
    - **Interacțiuni:**
        - Comunicare cu modulele de setare a timpului și de control al alarmei.
        - Afișarea informațiilor relevante către utilizator.
3. **Modul de Control al Alarmei:**
    - **Funcționalități:**
        - Permite utilizatorului să programeze și să activeze alarmele.
        - Declanșarea semnalului de alarmă (buzzer) la momentul setat.
    - **Interacțiuni:**
        - Comunicare cu modulul RTC pentru a verifica momentul activării alarmei.
        - Activarea buzzer-ului la momentul setat de utilizator.

4. Implementarea si testarea aplicației

- Descrierea hardware:

Pin	Configurație	Element de legătura	Descriere
PA10	GPIO	Pin buzzer	Pinul este folosit pentru a transmite un semnal electric către buzzer, care va emite un semnal sonor.

Buzzer-ul se va conectat la PA10 și la GND-ul de pe placă.

Descrierea software:

- module software:

modul de GESTIONARE AL TIMPULUI:

-
```
void Model::tick()
{
     if (tickstart == 0) {
            tickstart = HAL_GetTick();
            tickend = tickstart + 970;
        }

        if (HAL_GetTick() >= tickend) {
            if (s == 59) {
                s = 0;
                m++;
                if(alarm_minutes > 0)
                    alarm_minutes --;

            } else {
                s++;
            }

            if (m == 59) {
                m = 0;
                h++;
            }

            tickstart = 0;
            tickend = 0;
        }

        if (is_alarm_set == true && alarm_minutes == 0)
            {
                alarm();
            }
}
```

Modulul ALARMEI:

```
158 void Model::alarm()
159 {
160     if (tickalarmstart == 0) {
161             tickalarmstart = HAL_GetTick();
162                 tickalarmend = tickalarmstart + 350;
163             }
164
165             if (HAL_GetTick() >= tickalarmend) {
166                 HAL_GPIO_WritePin(BUZZER_GPIO_PORT, BUZZER_PIN, GPIO_PIN_RESET);
167
168                 if(alarm_s == 40){
169                     alarm_s = 0;
170                     is_alarm_set = false;
171                 }
172                 else{
173                     alarm_s++;
174                 }
175                 tickalarmstart = 0;
176                 tickalarmend = 0;
177             }
178             else
179                 HAL_GPIO_WritePin(BUZZER_GPIO_PORT, BUZZER_PIN, GPIO_PIN_SET);
180
181 }
182
```

modulul de ARMARE AL ALARMEI:

```
145 void Model::arm_alarm()
146 {
147     is_alarm_set = true;
148
149     GPIO_InitTypeDef gpioInit;
150     gpioInit.Pin = BUZZER_PIN;
151     gpioInit.Mode = GPIO_MODE_OUTPUT_PP;
152     gpioInit.Pull = GPIO_NOPULL;
153     gpioInit.Speed = GPIO_SPEED_FREQ_LOW;
154     gpioInit.Alternate = 0;
155     HAL_GPIO_Init(BUZZER_GPIO_PORT, &gpioInit);
156 }
```

module de OBȚINERE A INFORMAȚIILOR:

```
int getAlarmMinutes();
int getHours();
int getMinutes();
int getSeconds();
```

Module de MODIFICARE A INFORMAȚIILOR:

```
void incHours();
void decHours();
void incMinutes();
void decMinutes();
void incSeconds();
void decSeconds();
void incAlarm();
void decAlarm();
```

- interacțiunea dintre acestea:

Modulul principal de gestionare al timpului și al alarmei actualizează în timp real ceasul din interfața UI, și, de asemenea, verifică dacă o alarmă a fost setată în vederea declanșării acesteia. Acesta din urmă, comunică cu modulul alarmei, care se ocupă cu emiterea semnalului sonor de către buzzer, timp de 15s. Comunicația acestora are loc doar în momentul în care un flag este setat pe TRUE, iar timpul setat a expirat.

Modul de armare setează un flag pe TRUE, astfel, anunțând pe celelalte module că o alarma a fost setată. Totodată, inițializează și pinul pentru buzzer. Acesta comunică cu modulul de incrementare al minutelor alarmei, astfel, atunci când user-ul va seta mai mult de un minut pentru o alarmă, mai exact, atunci când se va apăsa pe butonul +, de incrementare.

Modulul principal mai comunică și cu modulele de obținere și modificare al informațiilor.

- Variabile utilizate:

```
int Model::h = 0;
int Model::m = 0;
int Model::s = 0;
int Model::alarm_minutes = 0;
bool Model::is_alarm_set = false;
bool Model::start_alarm = false;

int alarm_s;

uint32_t tickstart = 0;
uint32_t tickend = 0;

uint32_t tickalarmstart = 0;
uint32_t tickalarmend = 0;
```

- imagini relevante din procesul de testare a aplicației (ex. print screen-uri cu caracterele trimise pe magistrala de date, capturi de osciloscop, etc).

  Ecranul principal:

Ecranul de setare al orei:

Ecranul de setare al minutelor:

Ecranul de setare al alarmei:

5. Interpretarea rezultatelor și concluzii

În concluzie, în cadrul acestui proiect, am implementat un ceas deșteptător prin intermediul plăcii de dezvoltare STM32F429I și al unei buzzer. Acest lucru a fost posibil datorită bibliotecii HAL, dar și al modulelor create de noi, dar si utilizarea unui modul RTC adevărat :).

Îmbunătățiri potențiale ar putea viza optimizarea performanței și adăugarea de caracteristici suplimentare în viitor.

Bibliografie

1.STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs

2. 32b Arm® Cortex®-M4 MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 com. interfaces, camera & LCD-TFT

Anexe