Projekt

Sterowniki robotów

Dokumentacja

Logger danych. Akwizycja danych o temperaturze wraz z przeglądaniem danych archiwalnych na wyświetlaczu LCD

Logger temperatury

Skład grupy (2):
Ada Weiss, 218641
Beata Berajter, 218629

Termin: srTP11

Prowadzący:
mgr inż. Wojciech Domski

13 czerwca 2017

Spis treści

1 Opis projektu

Zadanie polega na tym, aby odczytywać dane z wewnętrznego termometru w określonych odstępach czasowych jednoczenie odczytując czas z RTC
Następnie dane te mają być przesyłane przez Quad SPI do pamięci zewnętrznej Flash. Zapamiętywanych jest 100 ostatnich wyników. Pamięć obsługiwana ma być w taki sposób, aby:

- zapisywać ostatnie miejsce w pamięci, w które został wpisany pomiar,
- w przypadku przekroczenia dostępnego miejsca - wyniki kolejno nadpisywały się w pamięci.

Ostatnią częścią projektu jest obsługa wyświetlacza LCD umożliwiająca za pomocą joysticka przejrzenie wszystkich zapisanych w pamięci pomiarów.

2 Zrealizowane prace

W programie STMCubeMx skonfigurowano mikrokontroler, zegar oraz peryferia. Po zapoznaniu się z dokumentacją udostępnianych przez ST przykładów dla płytki deweloperskiej STM32L476G [5] zdecydowałyśmy się wykorzystać biblioteki udostępnione w przykładach (do obsługi LCD oraz QSPI), co opisane jest w 2.3. Projekt został zrealizowany w całości zgodnie z założeniami.

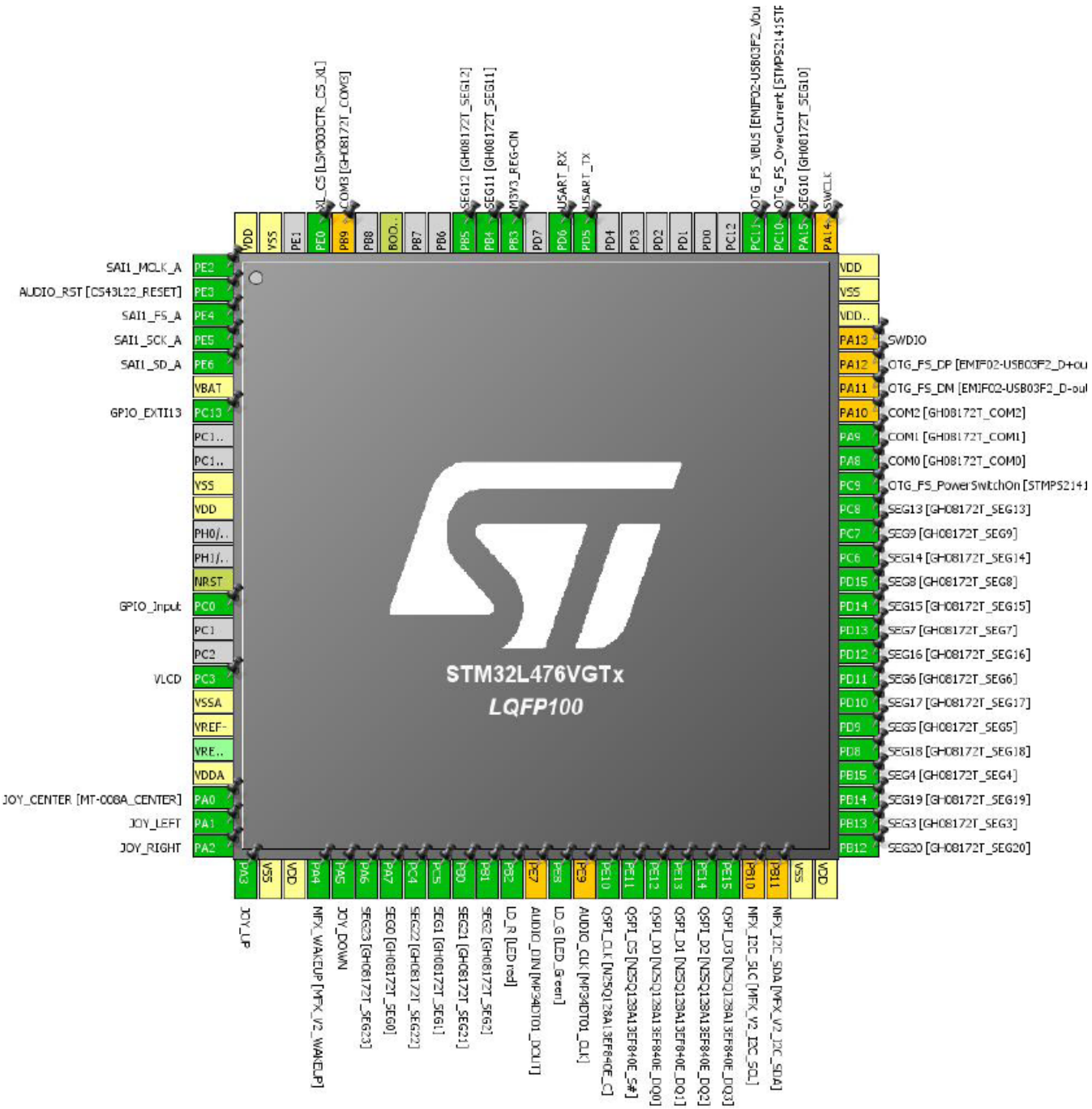
W tworzeniu projektu wykorzystywane były również materiały [4], [2] oraz [1].

2.1 Konfiguracja mikrokontrolera

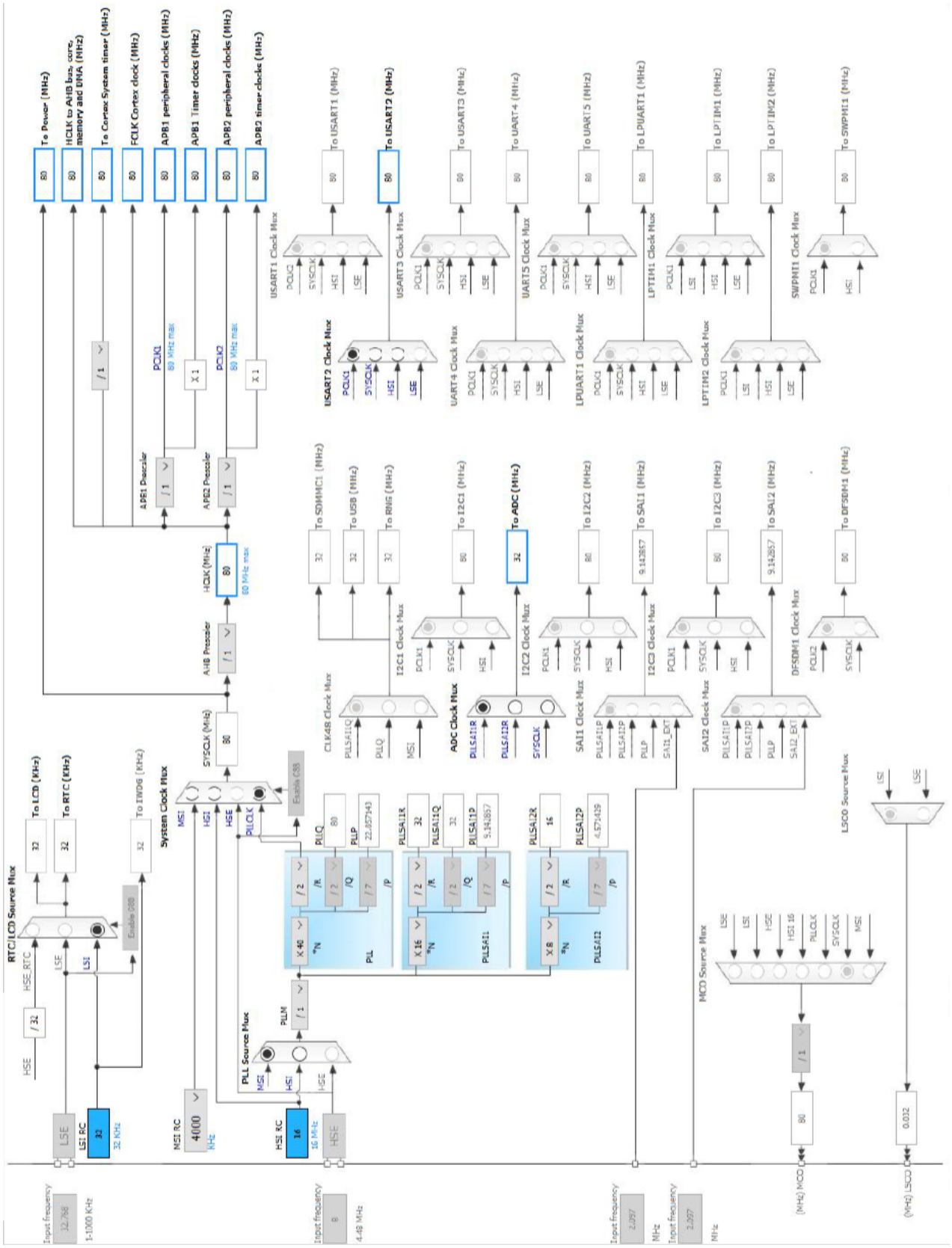
Konfiguracja pinów mikrokontrolera została przedstawiona w tabeli 1 i na rys. 1. Piny połączone z joystickiem zostały skonfigurowane jako GPIO_EXTI0. Skonfigurowane przez nas piny dotyczą przycisków górnego, dolnego oraz środkowego, które mają generować przerwania, do sterowania wyświetlaczem. Włączenie ADC wymagało zmiany konfiguracji zegara (32MHz dla ADC) co przedstawia rys. 2.

Tabela 1: Konfiguracja pinów mikrokontrolera wygenerowana w STM32CubeMx

Pin nr	**PINs**	**FUNCTIONs**	**LABELs**
1	PE2*	SAI1_MCLK_A	SAI1_MCK [CS43L22_MCLK]
2	PE3	GPIO_Output	AUDIO_RST [CS43L22_RESET]
3	PE4 *	SAI1_FS_A	SAI1_FS [CS43L22_LRCK]
4	PE5*	SAI1_SCK_A	SAI1_SCK [CS43L22_SCLK]
5	PE6*	SAI1_SD_A	SAI1_SD [CS43L22_SDIN]
7	PC13	GPIO_EXTI13	
8	PC14/	OSC32_IN*	RCC_OSC32_IN
9	PC15/	OSC32_OUT*	RCC_OSC32_OUT
12	PH0/	OSC_IN*	RCC_OSC_IN
13	PH1/	OSC_OUT*	RCC_OSC_OUT
15	PC0	GPIO_Input	
18	PC3	LCD_VLCD	VLCD
23	PA0	GPIO_EXTI0	JOY_CENTER [MT-008A_CENTER]
24	PA1	GPIO_Input	JOY_LEFT
25	PA2	GPIO_Input	JOY_RIGHT
26	PA3	GPIO_EXTI3	JOY_UP [MT-008A_UP]
29	PA4	GPIO_EXTI4	MFX_WAKEUP [MFX_V2_WAKEUP]
30	PA5	GPIO_EXTI5	JOY_DOWN [MT-008A_DOWN]
37	PB2	GPIO_Output	LD_R [LED red]
38	PE7*	SAI1_SD_B	AUDIO_DIN [MP34DT01_DOUT]
39	PE8	GPIO_Output	LD_G [LED_Green]
41	PE10	QUADSPI_CLK	QSPI_CLK [N25Q128A13EF840E_C]
42	PE11	QUADSPI_NCS	QSPI_CS [N25Q128A13EF840E_S#]
43	PE12	QUADSPI_BK1_IO0	QSPI_D0 [N25Q128A13EF840E_DQ0]
44	PE13	QUADSPI_BK1_IO1	QSPI_D1 [N25Q128A13EF840E_DQ1]
45	PE14	QUADSPI_BK1_IO2	QSPI_D2 [N25Q128A13EF840E_DQ2]
46	PE15	QUADSPI_BK1_IO3	QSPI_D3 [N25Q128A13EF840E_DQ3]
66	PC9	GPIO_Output	OTG_FS_PowerSwitchOn [STMPS2141STR_EN]
70	PA11*	USB_OTG_FS_DM	OTG_FS_DM [EMIF02-USB03F2_D-out]
71	PA12*	USB_OTG_FS_DP	OTG_FS_DP [EMIF02-USB03F2_D+out]
72	PA13*	SYS_JTMS-SWDIO	SWDIO
76	PA14*	SYS_JTCK-SWCLK	SWCLK
78	PC10	GPIO_EXTI10	OTG_FS_OverCurrent [STMPS2141STR_FAULT]
79	PC11	GPIO_Output	OTG_FS_VBUS [EMIF02-USB03F2_Vbus]
82	PD1*	SPI2_SCK	MEMS_SCK [L3GD20_SCL/SPC]
86	PD5	USART2_TX	USART_TX
87	PD6	USART2_RX	USART_RX
89	PB3	GPIO_Output	M3V3_REG-ON
97	PE0	GPIO_Output	XL_CS [LSM303CTR_CS_XL]



Rysunek 1: Konfiguracja pinów mikrokontrolera w programie STM32CubeMx



Rysunek 2: Konfiguracja zegara w programie STM32CubeMx

2.2 Konfiguracja peryferów mikrokontrolera

Po zapoznaniu się z dokumentacją mikrokontrolera [6] przeprowadzono konfigurację ADC w STM32CubeMx. Czujnik temperatury, który wykorzystujemy jest dostępny przez ADC1 i ADC3 na kanale 17. Konfiguracja ADC1 oraz innych peryferiów została przedstawiona w tab.2.

Tabela 2: Konfiguracja peryferiów mikrokontrolera

PER	**MODES**	**FUNCTIONS**	**PINS**
ADC1	Temperature Sensor Channel	ADC1_TempSens_Input	VP_ADC1_TempSens_Input
LCD	1/2 Duty Cycle	LCD_VLCD	PC3
LCD	1/2 Duty Cycle	LCD_COM0	PA8
LCD	1/2 Duty Cycle	LCD_COM1	PA9
LCD	Multiplex	LCD_LCD_mux	VP_LCD_LCD_mux
QUADSPI	Quad SPI Line	QUADSPI_BK1_IO0	PE12
QUADSPI	Quad SPI Line	QUADSPI_BK1_IO1	PE13
QUADSPI	Quad SPI Line	QUADSPI_BK1_IO2	PE14
QUADSPI	Quad SPI Line	QUADSPI_BK1_IO3	PE15
QUADSPI	Quad SPI Line	QUADSPI_NCS	PE11
QUADSPI	Quad SPI Line	QUADSPI_CLK	PE10
RTC	Activate RTC Clock Source	RTC_VS_RTC_Activate	VP_RTC_VS_RTC_Activate
RTC	RTC Enabled	RTC_VS_RTC_Calendar	VP_RTC_VS_RTC_Calendar
RTC	Internal Alarm A	RTC_VS_RTC_Alarm_A_Intern	VP_RTC_VS_RTC_Alarm_A_Intern
SYS	SysTick	SYS_VS_Systick	VP_SYS_VS_Systick
USART2	Asynchronous	USART2_RX	PD6
USART2	Asynchronous	USART2_TX	PD5

Konfigurację parametrów ADC przedstawia tabela 3.

Tabela 3: Konfiguracja ADC. (tylko wartości inne niż domyślne w STM32CubeMx)

Parameter	Settings
Clock Prescaler	Asynchronous clock mode divided by 8
Sampling Time	640.5 Cycles

Aktywowany został zegar RTC oraz skonfigurowano dla niego alarm, tak aby włączał się co minutę. Konfigurację alarmu na podstawie [7] przedstawia tabela 4.

Tabela 4: Konfiguracja RTC. (tylko wartości inne niż domyślne w STM32CubeMx)

Parameter	Settings
Alarm Mask Date Week day	Enable
Alarm Mask Hours	Enable
Alarm Mask Minutes	Enable
Alarm Mask Seconds	Disable

Konfiguracja joysticka została przedstawiona w tabeli 5

Tabela 5: Konfiguracja joysticka.

GPIO Pin	GPIO mode	GPIO Pull-up/Pull-down
JOY_CENTER	EXTI with falling edge detection	no pullup/pull-down
JOY_UP	EXTI with falling edge detection	pull-down
JOY_DOWN	EXTI with falling edge detection	pull-down

2.3 Opis wykorzystanych bibliotek i funkcji

2.3.1 ADC

Do odczytu danych z czujnika temperatury wykorzystane zostały przerwania. Obsługa przerwań od ADC jest przedstawiona na listingu 1..

```
1
2 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
3 {
4   if (hadc==&hadc1)
5   {
6     adc_value=HAL_ADC_GetValue(&hadc1);
7     adc_flag=1;
8   }
9 }
```

Listing 1: Obsługa przerwania od ADC

Zamiana danych na stopnie Celsiusza odbywa się zgodnie ze wzorem znajdującym się w dokumentacji [6], wykorzystywane są tam stałe TS_CAL1 i TS_CAL2, którym odpowiadają adresy w pamięci przedstawione w [3].

```
1     unsigned short int TS_CAL1=*((volatile unsigned short int*)0x1FFF75A8);
2     unsigned short int TS_CAL2=*((volatile unsigned short int*)0x1FFF75CA);
```

Dane przetwarzane są zgodnie ze wzorem z dokumentacji [6]

```
1     temperature=(110-30)*(adc_value-TS_CAL1)/(TS_CAL2-TS_CAL1)+30;
```

2.3.2 RTC

Zegar czasu rzeczywistego jest wykorzystywany do generowania przerwania co minutę w celu odczytania wartości temperatury oraz do odczytu daty pomiaru. Obsługa przerwania:

```
1 void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc){
2
3   HAL_ADC_Start_IT(&hadc1);
4 }
```

W main() konieczne jest uruchomienie przerwań dla alarmu RTC.

```
1 __HAL_RTC_ALARM_ENABLE_IT(&hrtc, RTC_IT_ALRA);
```

Odczytywanie daty z RTC wykonywane jest za pomocą funkcji:

```
1     HAL_RTC_GetTime(&hrtc,&sTime,RTC_FORMAT_BIN);
2     HAL_RTC_GetDate(&hrtc,&sDate,RTC_FORMAT_BIN);
```

Konieczne jest wcześniejsze utworzenie zmiennych

```
1       RTC_TimeTypeDef sTime;
2       RTC_DateTypeDef sDate;
```

2.3.3 LCD

Do obsługi wyświetlacza oraz QSPI_Flash zostały wykorzystane biblioteki dołączone do STM32CubeMx znajdujące się w */STM32Cube/Repository/STM32Cube_FW_L4_V1.7.0/Drivers/BSP/STM32L476G-Discovery*:

- stm32l476g_discovery_glass_lcd.h
- stm32l476g_discovery_glas_lcd.c
- stm32l476g_discovery_qspi.h
- stm32l476g_discovery_qspi.c

Do inicjalizacji peryferium wykorzystywana jest funkcja BSP_LCD_GLASS_Init() zamiast MSX_LCD_Init().

Natomiast do wyświetlenia wykorzystujemy funkcje.

```
1    void BSP_LCD_GLASS_DisplayString(uint8_t);
2    void BSP_LCD_GLASS_ScrollSentence(uint*ptr, uint16_t nScroll,uint16_t ScrollSpeed)
```

2.3.4 FLASH

Do zainicjalizowania pamięci flash wykorzystujemy funkcję

```
1    void QSPI_Init();
```

która, która wykorzystuje BSP_QSPI_Init() oraz sprawdza poprawność inicjalizacji peryferium.

W celu zapisania danych do pamięci Flash, konieczne jest wcześniejsze usunięcie wybranego sektora, dlatego napisana została funkcja przedstawiona na listingu 2. W funkcji tej dane z pamięci flash kopiowane są do tablicy, następnie są modyfikowane i pamięć jest czyszczona (usuwany jest QSPI_BLOCK wielkości 0x1000 bajtów za pomocą BSP_QSPI_Erase_Block()). Na koniec cała zmodyfikowana tablica jest zapisywana do pamięci Flash.

```
1  int QSPI_Write(int temp, RTC_DateTypeDef sDate, RTC_TimeTypeDef sTime, int ind){
2
3    int i=0;
4     for(i=0;i<101;i++){
5       BSP_QSPI_Read((uint8_t *)tmp1_from_flash[i], WRITE_READ_ADDR+(uint32_t)(BUFFER_SIZE
            *(i)), BUFFER_SIZE);
6
7       }
8
9    ind=(int)tmp1_from_flash[0][0];
10   sprintf(tmp1_from_flash[ind], "%02d%04d%02d%02d%02d%02d",(int)temp, sDate.Year, sDate.
         Month, sDate.WeekDay, sTime.Hours, sTime.Minutes);    if(BSP_QSPI_Erase_Block(
         WRITE_READ_ADDR) != QSPI_OK)
11              {
12                BSP_LCD_GLASS_ScrollSentence((uint8_t *)"       QSPI ERASE : FAILED.", 1,
                      SCROLL_SPEED_HIGH);
13                return 0;
14              }
15              else
16              {
17
18
19                for(i=1;i<101;i++){
20                  if(BSP_QSPI_Write((uint8_t *)tmp1_from_flash[i], WRITE_READ_ADDR+(
                        uint32_t)(BUFFER_SIZE*(i)),BUFFER_SIZE)!=QSPI_OK) return 0;
21                }
22                char index[1];
23                ind=ind+1;
24                  ind=ind%100;
25                  index[0]=ind;
```

```
26                  BSP_QSPI_Write((uint8_t *)index, WRITE_READ_ADDR, (uint32_t)(sizeof(index
                        )));
27              return 1;
28            }
29 }
```

Listing 2: Funkcja zapisu do QSPI Flash

Do odczytywania z pamięci Flash wykorzystywana jest funkcja z biblioteki BSP w przedstawiony poniżej sposób:

```
1  if(BSP_QSPI_Read(qspi_aRxBuffer, WRITE_READ_ADDR+(uint32_t)BUFFER_SIZE*(current_ind),
       BUFFER_SIZE) != QSPI_OK)
2          BSP_LCD_GLASS_ScrollSentence((uint8_t *)"        QSPI READ : FAILED.", 1,
               SCROLL_SPEED_HIGH);
```

2.3.5 JOYSTICK

Joystick wykorzystywany jest do poruszania się po "menu". Obsługa przerwania od odpowiednich przycisków joysticka została przedstawiona na listingu 3.

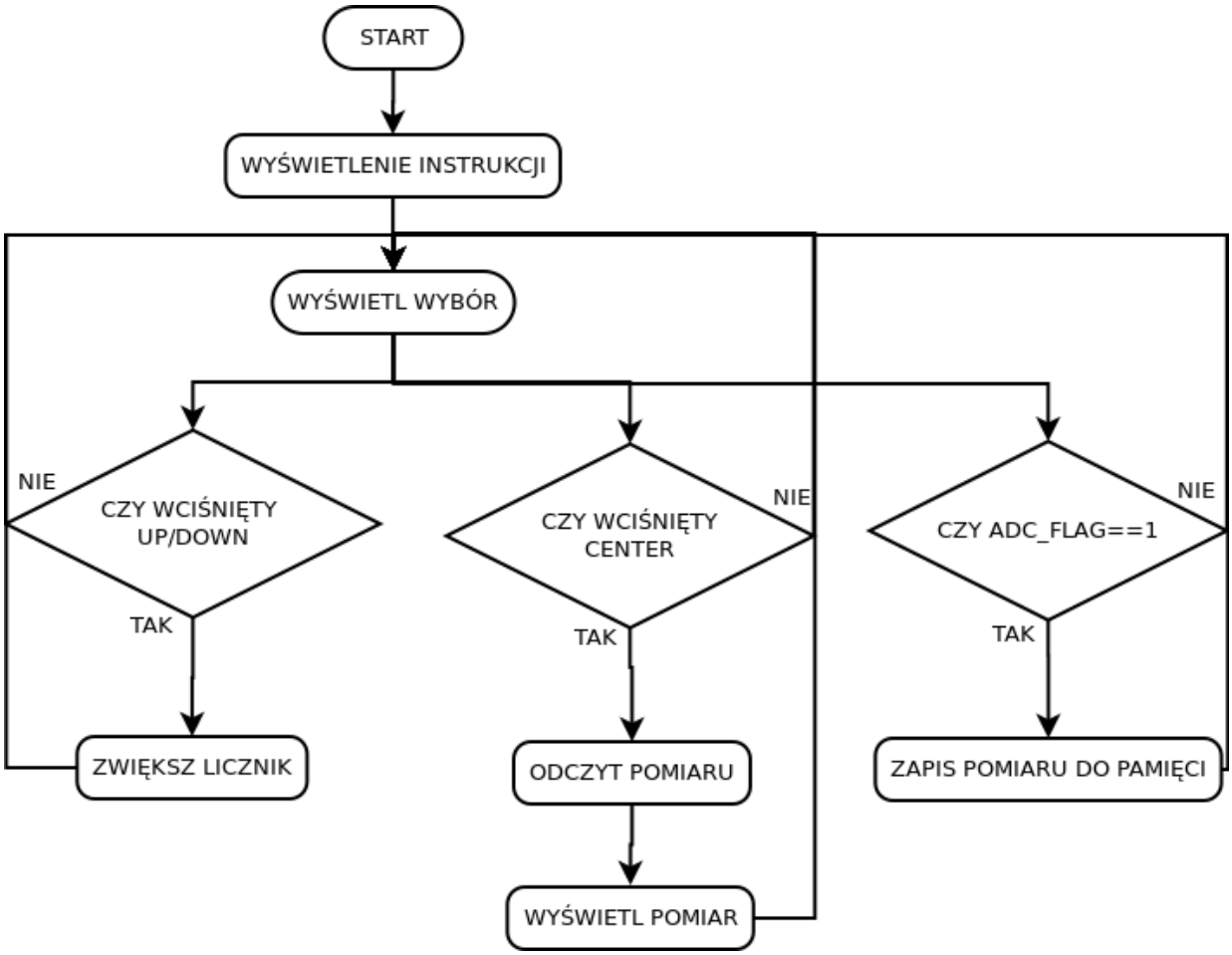
```
1  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
2     if(GPIO_Pin == JOY_UP_Pin){
3     HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
4     if(current_ind==100) current_ind = 0;
5     ++current_ind;
6     current_ind=current_ind%101;
7
8
9     lcd_flag=1;
10    }
11    if(GPIO_Pin == JOY_DOWN_Pin){
12      HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
13      --current_ind;
14       if(current_ind==0)current_ind=100;
15       lcd_flag=1;
16       }
17    if(GPIO_Pin==JOY_CENTER_Pin){
18      lcd_pomiar=1;    }
19
20   }
```

Listing 3: Funkcja obsługująca przerwanie zewnętrzne od Joysticka

2.4 Opis działania programu

Opis dziania programu został przedstawiony na schemacie blokowym 3

Po starcie następuje inicjacja ADC, RTC, Flash, ekranu LCD oraz konfiguracja GPIO. Na ekranie zostaje wyświetlona instrukcja dotycząca wyboru pomiaru : przycisk UP/DOWN - przejście do następnego/poprzedniego pomiaru, przycisk CENTER - wyświetlenie wybranego pomiaru. W pętli następuje sprawdzenie wyboru użytkownika oraz odpowiednia reakcja, po czym program czeka na kolejną akcję użytkownika. Jednocześnie w tle prowadzone są pomiary. Wykonywane są one co jedną minutę.



Rysunek 3: Schemat działania programu

Literatura

[1] mgr inż. Wojciech Domski, Materiały do zajęć. edu.domski.pl.

[2] Kurs STM32 F4 #5 i #4. http://forbot.pl/blog/artykuly/programowanie/kurs-stm32-f4-5-pomiar-napiecia-adc-dma-stmstudio-id13099, May 2015.

[3] STM32L476xx. MCU Datasheet. http://www.st.com/content/ccc/resource/technical/document/datasheet/c5/ed/2f/60/aa/79/42/0b/DM00108832.pdf/files/DM00108832.pdf/jcr:content/translations/en.DM00108832.pdf, December 2015.

[4] STM32L476G-DISCOVERY. User manual. http://www.st.com/content/ccc/resource/technical/document/user_manual/d1/84/86/4b/08/82/47/91/DM00172179.pdf/files/DM00172179.pdf/jcr:content/translations/en.DM00172179.pdf, March 2016.

[5] STM32Cube firmware examples for STM32L4 Series. Application note. http://www.st.com/content/ccc/resource/technical/document/application_note/group0/09/77/32/19/96/59/4c/1e/DM00209748/files/DM00209748.pdf/jcr:content/translations/en.DM00209748.pdf, February 2017.

[6] STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs. Reference manual. http://www.st.com/content/ccc/resource/technical/document/reference_

manual/02/35/09/0c/4f/f7/40/03/DM00083560.pdf/files/DM00083560.pdf/jcr:content/translations/en.DM00083560.pdf, February 2017.

[7] Using the hardware real-time clock (RTC) in low-power modes with STM32 microcontrollers. http://www.st.com/content/ccc/resource/technical/document/application_note/group0/71/b8/5f/6a/8e/d5/45/0a/DM00226326/files/DM00226326.pdf/jcr:content/translations/en.DM00226326.pdf, May 2017.