Національний технічний університет України

“ Київський політехнічний інститут ”



Лабораторна робота №4

По курсу: ” МІКРОПРОЦЕСОРНІ СИСТЕМИ НА БАЗІ ARM ПРОЦЕСОРІВ*”*

Тема: “ Передача даних інтерфейсом I2C ”

Варіант 6

Виконав:Студент І курсу

Групи ДС-31мп

Рєзнік Олександр Сергійович

Київ 2024 р.

**Мета роботи:** Ознайомитися з інтерфейсом I2C. Навчитися налаштовувати та використовувати I2C для передачі даних між мікроконтролером STM32 та сенсором BMP180. Вивести отримані дані на LCD дисплей.

**Хід роботи:**

**Налаштування мікроконтролера STM32:**

* Ініціалізація системного годинника.
* Ініціалізація GPIO.
* Ініціалізація I2C.

**Ініціалізація периферійних пристроїв:**

* Ініціалізація LCD дисплея.
* Ініціалізація сенсора BMP180.

**Основний цикл програми:**

* Зчитування даних температури та тиску з сенсора BMP180.
* Виведення отриманих даних на LCD дисплей.
* Оновлення даних кожну секунду.

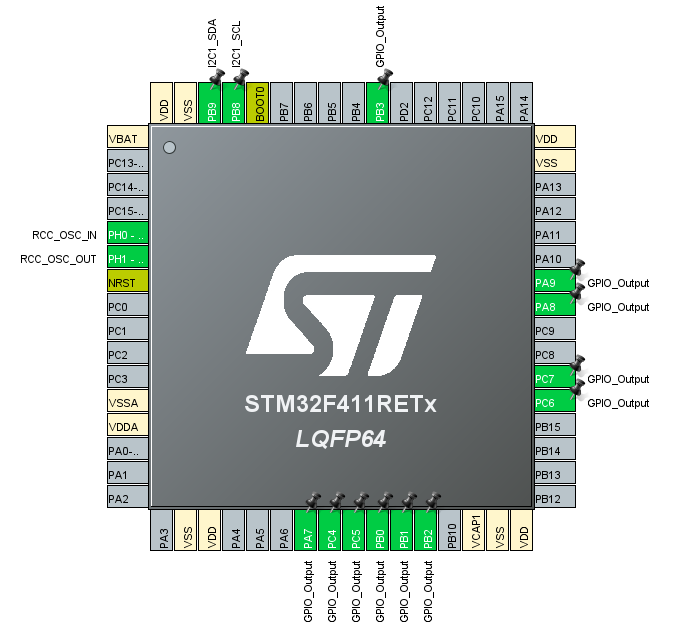


Рис. 1 Конфігурація пінів

Лістинг коду:

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

**#include** "stm32f4xx\_hal.h"

**#include** "bmp180\_for\_stm32\_hal.h"

**#include** "Lcd.h"

**#include** "stdio.h"

**#include** "stdlib.h"

**#include** "string.h"

/\* USER CODE END Includes \*/

/\* Private variables ---------------------------------------------------------\*/

I2C\_HandleTypeDef hi2c1;

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_I2C1\_Init**(**void**);

**int** **main**(**void**)

{

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_I2C1\_Init();

/\* USER CODE BEGIN 2 \*/

quickInitLCD(); // Ініціалізація дисплея

/\* Initializes BMP180 sensor and oversampling settings. \*/

BMP180\_Init(&hi2c1);

BMP180\_SetOversampling(*BMP180\_ULTRA*);

/\* Update calibration data. Must be called once before entering main loop. \*/

BMP180\_UpdateCalibrationData();

**char** buffer1[16];

**char** buffer2[16];

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

clearDisplay(); // Очистка дисплея

/\* Reads temperature. \*/

int32\_t temperature = BMP180\_GetRawTemperature();

/\* Reads pressure. \*/

int32\_t pressure = BMP180\_GetPressure();

**sprintf**(buffer1, " Temp: %d.%d C", (**int**) temperature / 10, (**int**) temperature % 10); //заповнення змінної відповідними значеннями

printstr(buffer1); // Виведення числа на дисплей

setCursor(0, 0); //повернення курсору на початок

HAL\_Delay(1000);

**sprintf**(buffer2, " Pres: %d Pa", (**int**) pressure);

printstr(buffer2); // Виведення числа на дисплей

setCursor(0, 0); //повернення курсору на початок

HAL\_Delay(1000);

}

/\* USER CODE END 3 \*/

}

…\bmp180\_for\_stm32\_hal.h

**const** uint8\_t BMP180\_EEPROM\_ADDR\_MSB[11] = { 0xaa, 0xac, 0xae, 0xb0, 0xb2, 0xb4, 0xb6, 0xb8, 0xba, 0xbc, 0xbe };

**const** uint8\_t BMP180\_EEPROM\_ADDR\_LSB[11] = { 0xab, 0xad, 0xaf, 0xb1, 0xb3, 0xb5, 0xb7, 0xb9, 0xbb, 0xbd, 0xbf };

**const** uint8\_t BMP180\_CMD\_TEMP = 0x2e;

**const** uint8\_t BMP180\_DELAY\_TEMP = 5;

**const** uint8\_t BMP180\_CMD\_PRES[4] = { 0x34, 0x74, 0xb4, 0xf4 };

**const** uint8\_t BMP180\_DELAY\_PRES[4] = { 5, 8, 14, 26 };

**void** **BMP180\_Init**(I2C\_HandleTypeDef \*hi2c) {

\_bmp180\_ui2c = hi2c;

}

**void** **BMP180\_SetOversampling**(BMP180\_OSS oss) {

\_bmp180\_oss = oss;

}

/\*\*

\* @brief Updates calibration data.

\* @note Must be called once before main loop.

\*/

**void** **BMP180\_UpdateCalibrationData**(**void**) {

\_bmp180\_eeprom.BMP180\_AC1 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC1*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC1*]);

\_bmp180\_eeprom.BMP180\_AC2 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC2*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC2*]);

\_bmp180\_eeprom.BMP180\_AC3 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC3*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC3*]);

\_bmp180\_eeprom.BMP180\_AC4 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC4*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC4*]);

\_bmp180\_eeprom.BMP180\_AC5 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC5*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC5*]);

\_bmp180\_eeprom.BMP180\_AC6 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_AC6*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_AC6*]);

\_bmp180\_eeprom.BMP180\_B1 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_B1*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_B1*]);

\_bmp180\_eeprom.BMP180\_B2 = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_B2*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_B2*]);

\_bmp180\_eeprom.BMP180\_MB = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_MB*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_MB*]);

\_bmp180\_eeprom.BMP180\_MC = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_MC*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_MC*]);

\_bmp180\_eeprom.BMP180\_MD = (BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_MSB[*BMP180\_INDEX\_MD*]) << 8) | BMP180\_ReadReg(BMP180\_EEPROM\_ADDR\_LSB[*BMP180\_INDEX\_MD*]);

}

**void** **BMP180\_WriteReg**(uint8\_t reg, uint8\_t cmd) {

uint8\_t arr[2] = { reg, cmd };

HAL\_I2C\_Master\_Transmit(\_bmp180\_ui2c, BMP180\_I2C\_ADDR << 1, arr, 2, BMP180\_I2C\_TIMEOUT);

}

/\*\*

\* @brief Reads from a specific register.

\* @param reg Address of register to read from.

\* @return Byte read.

\*/

uint8\_t **BMP180\_ReadReg**(uint8\_t reg) {

HAL\_I2C\_Master\_Transmit(\_bmp180\_ui2c, BMP180\_I2C\_ADDR << 1, &reg, 1, BMP180\_I2C\_TIMEOUT);

uint8\_t result;

HAL\_I2C\_Master\_Receive(\_bmp180\_ui2c, BMP180\_I2C\_ADDR << 1, &result, 1, BMP180\_I2C\_TIMEOUT);

**return** result;

}

/\*\*

\* @brief Measures and calculates temperature.

\* @return Temperature in 0.1 (1/10) degrees Celsius.

\*/

int32\_t **BMP180\_GetRawTemperature**(**void**) {

BMP180\_WriteReg(BMP180\_CONTROL\_REG, BMP180\_CMD\_TEMP);

HAL\_Delay(BMP180\_DELAY\_TEMP);

int32\_t ut = (BMP180\_ReadReg(BMP180\_MSB\_REG) << 8) | BMP180\_ReadReg(BMP180\_LSB\_REG);

int32\_t x1 = (ut - \_bmp180\_eeprom.BMP180\_AC6) \* \_bmp180\_eeprom.BMP180\_AC5 / (1 << 15);

int32\_t x2 = (\_bmp180\_eeprom.BMP180\_MC \* (1 << 11)) / (x1 + \_bmp180\_eeprom.BMP180\_MD);

int32\_t b5 = x1 + x2;

**return** (b5 + 8) / (1 << 4);

}

/\*\*

\* @brief Measures and calculates temperature.

\* @return Temperature in degrees Celsius.

\*/

**float** **BMP180\_GetTemperature**(**void**) {

int32\_t temp = BMP180\_GetRawTemperature();

**return** temp / 10.0;

}

/\*\*

\* @brief Measures and calculates pressure.

\* @return Pressure in Pascal(Pa).

\*/

int32\_t **BMP180\_GetPressure**(**void**) {

BMP180\_WriteReg(BMP180\_CONTROL\_REG, BMP180\_CMD\_TEMP);

HAL\_Delay(BMP180\_DELAY\_TEMP);

int32\_t ut = BMP180\_GetUT();

BMP180\_WriteReg(BMP180\_CONTROL\_REG, BMP180\_CMD\_PRES[\_bmp180\_oss]);

HAL\_Delay(BMP180\_DELAY\_PRES[\_bmp180\_oss]);

int32\_t up = BMP180\_GetUP();

int32\_t x1 = (ut - \_bmp180\_eeprom.BMP180\_AC6) \* \_bmp180\_eeprom.BMP180\_AC5 / (1 << 15);

int32\_t x2 = (\_bmp180\_eeprom.BMP180\_MC \* (1 << 11)) / (x1 + \_bmp180\_eeprom.BMP180\_MD);

int32\_t b5 = x1 + x2;

int32\_t b6 = b5 - 4000;

x1 = (\_bmp180\_eeprom.BMP180\_B2 \* (b6 \* b6 / (1 << 12))) / (1 << 11);

x2 = \_bmp180\_eeprom.BMP180\_AC2 \* b6 / (1 << 11);

int32\_t x3 = x1 + x2;

int32\_t b3 = (((\_bmp180\_eeprom.BMP180\_AC1 \* 4 + x3) << \_bmp180\_oss) + 2) / 4;

x1 = \_bmp180\_eeprom.BMP180\_AC3 \* b6 / (1 << 13);

x2 = (\_bmp180\_eeprom.BMP180\_B1 \* (b6 \* b6 / (1 << 12))) / (1 << 16);

x3 = ((x1 + x2) + 2) / 4;

uint32\_t b4 = \_bmp180\_eeprom.BMP180\_AC4 \* (uint32\_t) (x3 + 32768) / (1 << 15);

uint32\_t b7 = ((uint32\_t) up - b3) \* (50000 >> \_bmp180\_oss);

int32\_t p;

**if** (b7 < 0x80000000)

p = (b7 \* 2) / b4;

**else**

p = (b7 / b4) \* 2;

x1 = (p / (1 << 8)) \* (p / (1 << 8));

x1 = (x1 \* 3038) / (1 << 16);

x2 = (-7357 \* p) / (1 << 16);

p = p + (x1 + x2 + 3791) / (1 << 4);

**return** p;

}

…\Lcd.h

**void** **quickInitLCD**() {

HAL\_Delay(200);//power on init time

//setEnableSig(GPIO\_PIN\_SET);//set enable signal

setInputType(*GPIO\_PIN\_RESET*);//set to instruction mode

setReadWriteMode(*GPIO\_PIN\_RESET*);//set to write

sendInstruction( 0x30 );

HAL\_Delay(4);

sendInstruction( 0x30 );

HAL\_Delay(4);

sendInstruction( 0x30 );

HAL\_Delay(4);

//now in 4bit mode

sendInstruction( 0x20 );

HAL\_Delay(4);

//disp settings

sendInstruction( 0x28 );

HAL\_Delay(4);

//disp on

sendInstruction( 0x08 );

HAL\_Delay(4);

//clear display

sendInstruction( 0x01 );

HAL\_Delay(4);

//entry mode

sendInstruction( 0x06 );

HAL\_Delay(4);

//Setting LCD to 4-bit mode

//sendInstruction(0x28);

//Display parameters of LCD

sendInstruction(0x0F);

**return**;

}

**void** **clearDisplay**() {

sendInstruction(0X01);//instruction to clear display

HAL\_Delay(50);

**return**;

}

**void** **printstr**(**char**\* string) {

**int** i = 0;

**while**(string[i] !='\0') {

sendData( (**int**)string[i] );

HAL\_Delay(20);

i++;

}

**return**;

}

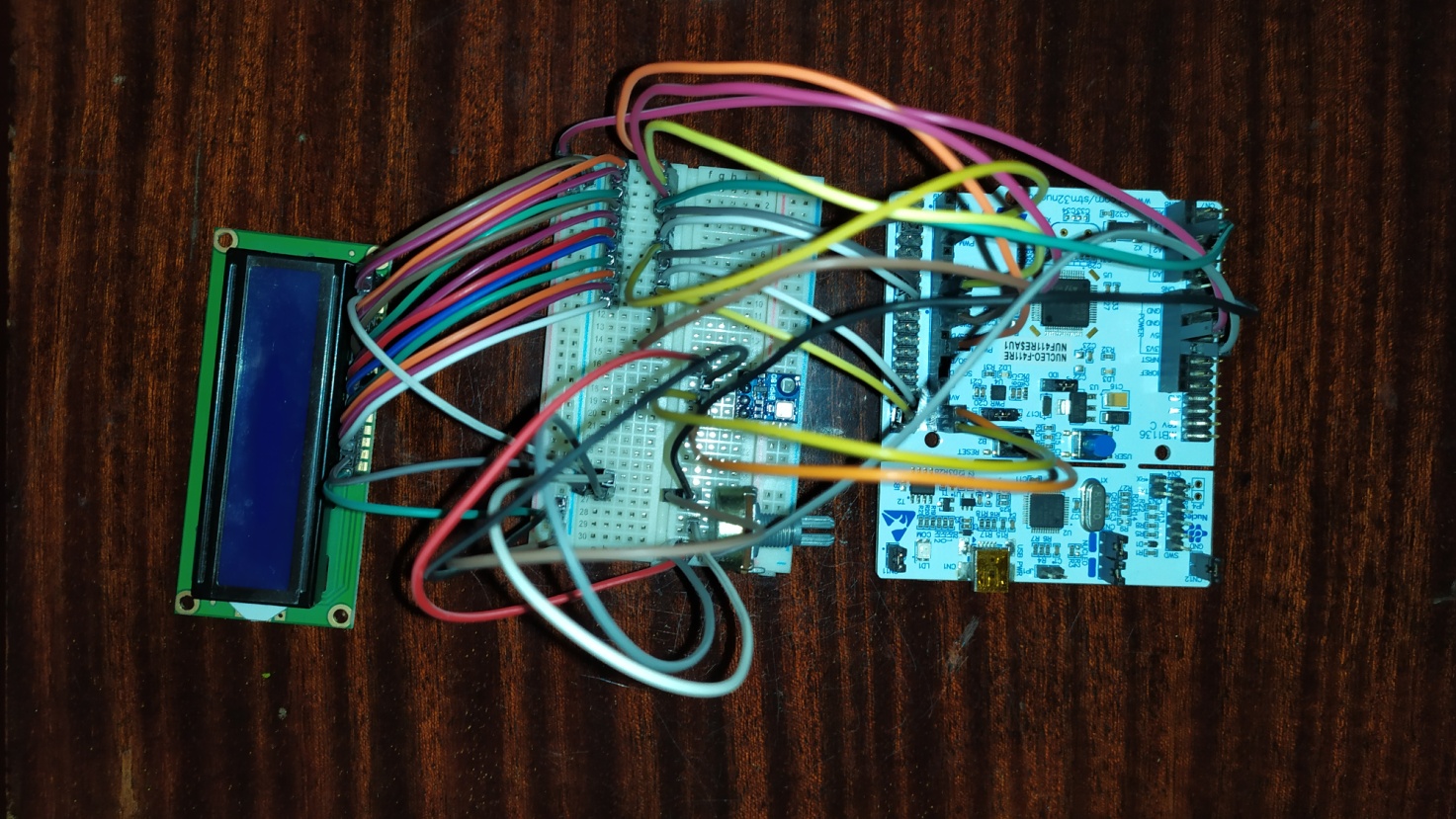


Рис. 2 Зібраний макет для виконання завдання

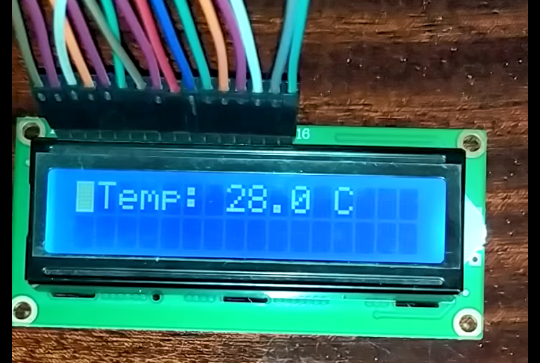


Рис. 3Виведення на дисплей значення температури



Рис. 4 Виведення на дисплей значень тиску

Висновок:

В результаті виконання лабораторної роботи було реалізовано передачу даних між мікроконтролером STM32 та сенсором BMP180 за допомогою інтерфейсу I2C. Дані температури та тиску успішно зчитувалися з сенсора та виводилися на LCD дисплей. Це дозволило ознайомитися з принципами роботи інтерфейсу I2C та його налаштуванням на практиці, а також поглибити знання з мікропроцесорних систем на базі ARM процесорів.