МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

з дисципліни «Проектування мікроконтролерних пристроїв в радіотехнічних комп’ютеризованих системах»

на тему «Реалізація годинника»

Виконав:

студент РТФ

групи РС-п21

Плясов Д. Ю.

Київ – 2024

Зміст

Вступ С.3

1. Опис використаних елементів бази С.4

2. Структурна схема пристрою С.5

3. Схема електрична-принципова С.6

4. Блок-схеми С.7

5. Інструкція по керуванню пристроєм для користувача С.9

6. Програмний код прошивки С.10

Висновки С.22

Вступ

Завданням до розрахункової роботи, було створення годинника з додатковими функціями на базі мікропроцесора ATmega16A. За варіантом №7, вивід інформації повинен здійснюватися на 7-сегментний дисплей, з додатковою функцією показу температури. Налаштування годинника повинно відбуватись за допомогою UART.

Під час виконання розрахунково-графічної роботи був розроблений годинник який налаштовується по UART(Universal asynchronous receiver/transmitter) через ПК завдяки конвертеру «USB to TTL» CH-597. Також пристрій має додаткову функцію виводу температури завдяки датчику температури та вологості SHT3X передавання інформації якої зроблено через TWI (Two Wired Interface).

Розробка прошивки виконувалась у середовищі розробки AVRStudio на мові програмування С.

1. Опис використаних елементів бази

Під час виконання роботи були використані такі елементи:

ATmega16A – основний мікроконтролер на якому виконувалась робота.

AVRISP mkII – програматор для мікроконтролеру, завдяки якому виконувалась прошивка плати.

HW-597 – конвертер USB to TTL, USB пристрій який емулює COM-порт завдяки якому мікроконтролер може спілкуватися з ПК.

STH3X – датчик температури та вологості, який підключений до контролеру шиною TWI.

1. Структурна схема пристрою

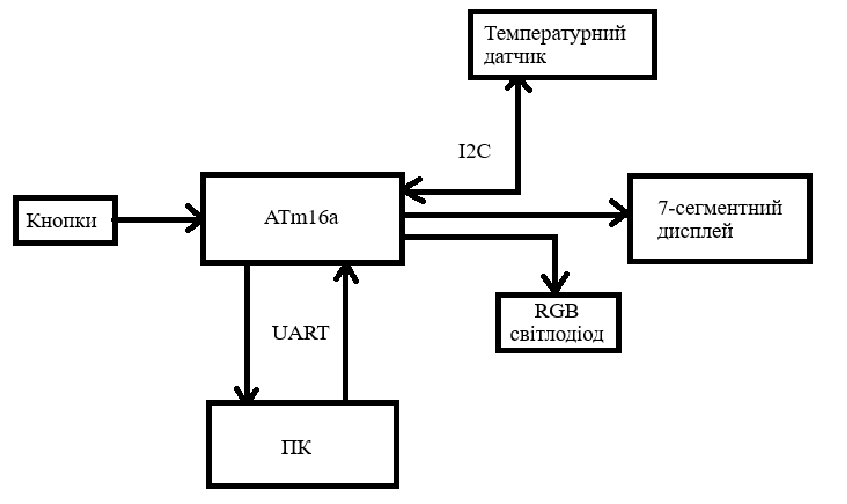


Рисунок 1 – Структурна схема пристрою

Принцип роботи девайсу наступний, контролер встановлює значення часу по замовчуванню, після чого виводить на 7-сегментний дисплей поточний час (у 24 часовому форматі). При досягнені нової години, RGB-світлодіод почне мигати з затримкою у 1 секунду, стільки разів, скільки зараз годин (у 12 часовому форматі).

При натисканні кнопок, змінюється режим роботи. При натисканні на кнопку 1 – пристрій перейде у режим показу температури з датчику. Мікроконтролер відправляє через I2C команду запуску розрахунку на датчик, той в свою чергу відправляє сирий результат до процесора, після перерахунку, інформація виводиться на дисплей, оновлення відбувається кожну секунду.

При натисканні на кнопку 0 – режим роботи змінюється на зміну часу через UART, від ПК приймається години, хвилини і секунди, після чого, поченається новий відлік часу.

1. Схема електрична принципова

Схема електрична-принципова була виконана у САПР Proteus 8.

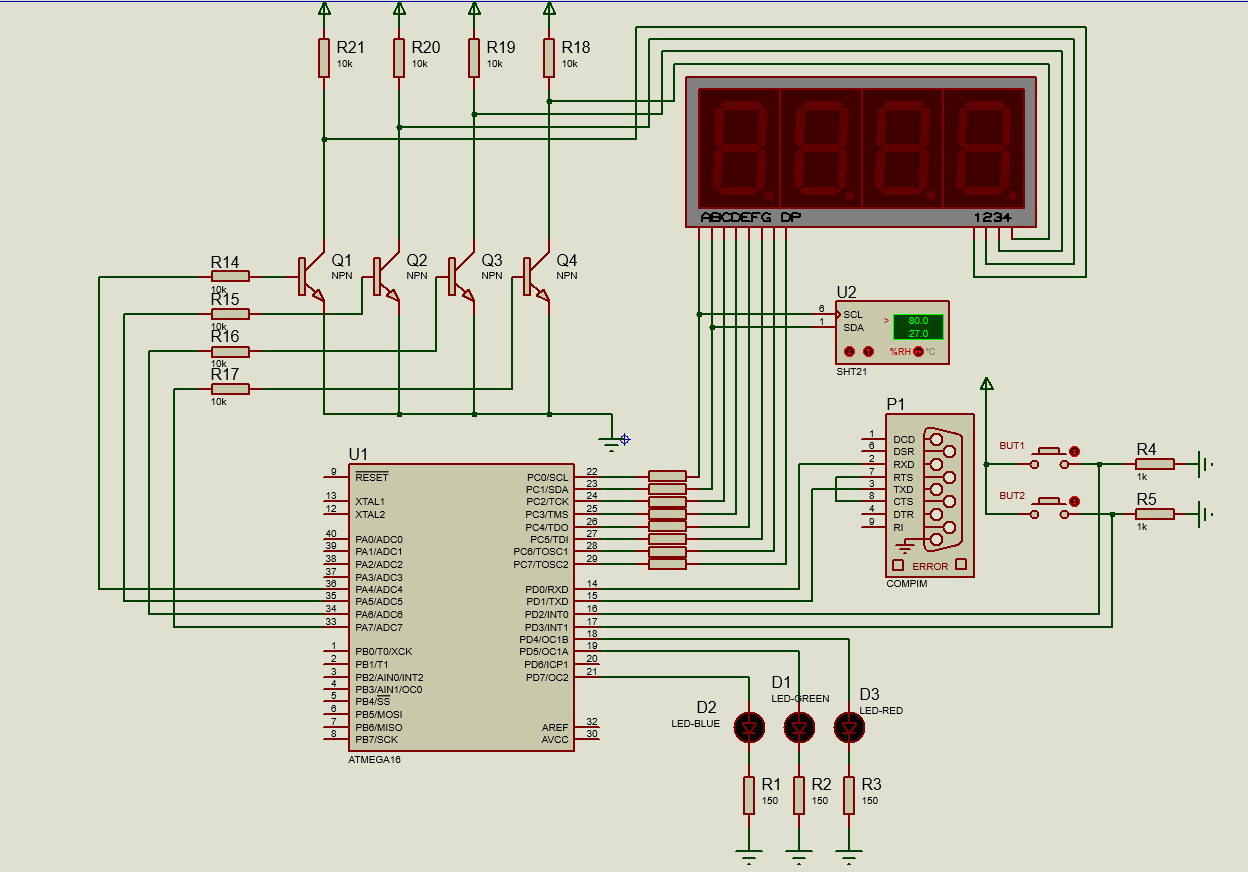


Рисунок 2 – схема електрична-принципова

На схемі показано електрична схема плати ATmega16A та модулів які використовуються у даній роботі, де U1 – сам мікроконтролер, D1:3 – світлодіоди, BTN1:2 – кнопки керування, 7-сегментний дисплей з 4 блоками, U2 – датчик температури, P1 – вивід COM-порту.

1. Блок-схеми основних алгоритмів роботи програми

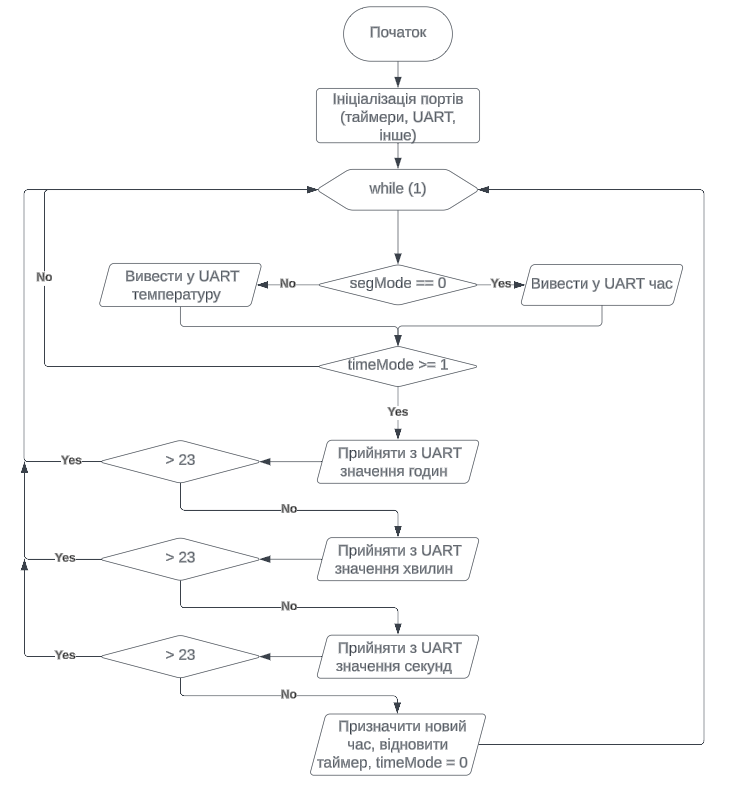


Рисунок 3 – Блок схема основної функції main(), де segMode – режим відображення (1 – температура, 0 – час), timeMode – режим таймеру оновлення (1 – таймер вимкнений, режим налагодження, 0 – таймер ввімкнений)

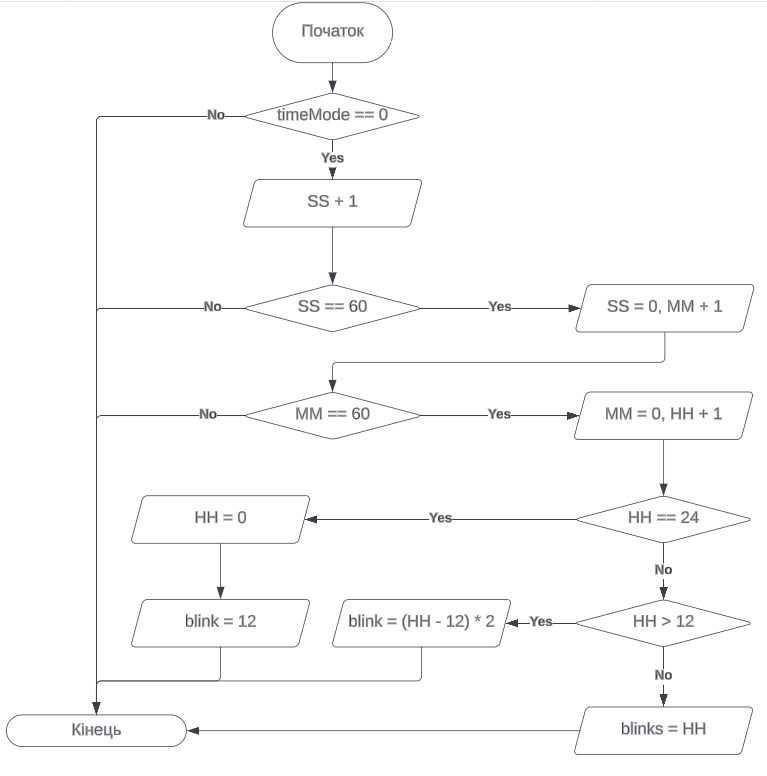


Рисунок 4 – Блок схеми секундного таймера TIMER1, де timeMode – режим таймеру оновлення (1 – таймер вимкнений, режим налагодження, 0 – таймер ввімкнений), SS – секунд, MM – хвилини, HH – години, blink – кількість бліків світлодіода

1. Інструкція по керуванню пристроєм для користувача

Для того щоб запустити пристрій, спочатку його треба підключити до живлення 5В, для цього можна підключити до ПК або блоку живлення за допомогою проводу USB-A – miniUSB до самої плати (при наявності порту miniUSB), або до програматора, який підключений 6-піновим шлейфом до плати.

Після підключення живлення, пристрій запуститься, на 7-сегментному дисплеї з’явиться час по замовчуванню 11:59:50 (у форматі HH:MM). Як тільки час досягне години, світлодіод на платі почне періодично світитися (1 блік в 2 секунди), кількість бліків залежить від години (5 година – 5 бліків, 15 година – 3 бліки).

Для того щоб переналаштувати час, треба підключити «конвертер USB to TTL» до ПК, запустити програму-термінал, вибрати номер потрібного COM-порта, та встановити частоту Baud Rate на 2400, після чого натиснути «Connect». На вікні виводу інформації повинна з’явитись стрічка з поточним часом (у форматі HH:MM:SS). Далі, на пристрої треба натиснути кнопку 0 (сама крайня) після чого на екрані виводу інформації в терміналі повинна з’явитись напис «Enter hours(HH):», який означає, девайс перейшов у режим керування часом. Тепер треба ввести у поле відправки число годин (від 00 до 23, при введені більшого числа, на екрані з’явиться текст «Wrong hours»), і натиснути «Send» (Зауважте, що галочка на «+CR» повинна бути вимкнута), після чого на екрані з’явиться текст «Enter minutes(MM)», так само вводимо число (00 - 59), так само з’явиться текст «Enter seconds(SS)», повторіть минулі кроки. Після чого час буде змінено.

Для того щоб побачити температуру датчика треба натиснути кнопку 1, після чого на термінал буде виводитись температура у градусах Цельсія, а також на 7-сегментному дисплеї з точністю 1 символ після коми.

1. Програмний код прошивки

Файли заголовків:

main.h

//main.h

#ifndef MAIN\_H\_

#define MAIN\_H\_

#include <avr/io.h>

#include <avr/delay.h>

#include <avr/interrupt.h>

#endif /\* MAIN\_H\_ \*/

uartlib.h

//uartlib.h

#ifndef UARTLIB\_H\_

#define UARTLIB\_H\_

void UARTInit();

void UARTSend(char Value);

char UARTReceive();

void SendString(char\* str);

#endif /\* UARTLIB\_H\_ \*/

TWI.h

//TWI.h

#ifndef TWI\_H\_

#define TWI\_H\_

typedef enum

{

standard,

fast

} TwiMode;

void twi\_init(TwiMode mode);

void twi\_shut(void);

int twi\_master\_tx(uint8\_t addr, uint8\_t \*data, int len);

int twi\_master\_rx(uint8\_t addr, uint8\_t \*data, int len);

int twi\_master\_tx\_rx(uint8\_t addr, uint8\_t \*tx\_data, int tx\_len,

uint8\_t \*rx\_data, int rx\_len);

#endif /\* TWI\_H\_ \*/

Файли С

main.c

//main.c

// Libraries

#include "main.h"

#include "uartlib.h"

#include "TWI.h"

// Consts

#define RED 0x80

#define GREEN 0x20

#define BLUE 0x10

#define BTN1 0x04

#define BTN2 0x08

#define DIGITS 4

// Interrupts prototypes

ISR(TIMER0\_COMP\_vect);

ISR(TIMER1\_COMPA\_vect);

ISR(TIMER2\_COMP\_vect);

ISR(INT0\_vect);

ISR(INT1\_vect);

// Functions prototypes

void TimerInit();

void CalculateTemp();

void SendTime();

void SendTemperature();

void TempTo7Seg();

void NumToArr(int numbr);

void TempToArr(int temp);

unsigned char DecToDigit(unsigned char Dec);

// All global variable

mode = fast;

int j = 0;

int n = 0;

int timeArr[DIGITS];

int tempArr[DIGITS];

int HH = 11;

int MM = 59;

int SS = 50;

int time = 0;

long int temp = 270;

double cTemp;

char data[6] = {0};

char sendData[2] = {0x2C, 0x06};

char string[128];

char tmpStr[128];

int timeMode = 0;

int flag = 0;

char tmp[3];

int blink = 0;

int segMode = 0;

int main(void)

{

DDRA = 0xF0;

DDRC = 0xFF;

DDRD |= RED | GREEN | BLUE | BTN1 | BTN2;

MCUCR = 0x0F;

GICR = 0xC0;

\_delay\_ms(1);

TimerInit();

UARTInit();

sei();

while(1)

{

if (segMode == 0)

{

SendTime();

}

else

{

//TempToArr(temp);

ltoa((long int)(cTemp), tmpStr, 10);

TempToArr((long int)(cTemp));

SendTemperature();

//TempTo7Seg();

}

if (timeMode >= 1)

{

while(1)

{

HH = 0;

MM = 0;

SS = 0;

SendString("Enter hours (HH)");

UARTSend('\r');

UARTSend('\n');

tmp[0] = UARTReceive();

tmp[1] = UARTReceive();

HH = atoi(tmp);

if (HH > 23)

{

SendString("Wrong hours");

UARTSend('\r');

UARTSend('\n');

break;

}

\_delay\_ms(100);

SendString("Enter minutes (MM)");

UARTSend('\r');

UARTSend('\n');

tmp[0] = UARTReceive();

tmp[1] = UARTReceive();

MM = atoi(tmp);

if (MM > 59)

{

SendString("Wrong minutes");

UARTSend('\r');

UARTSend('\n');

break;

}

\_delay\_ms(100);

SendString("Enter seconds (SS)");

UARTSend('\r');

UARTSend('\n');

tmp[0] = UARTReceive();

tmp[1] = UARTReceive();

SS = atoi(tmp);

if (SS > 59)

{

SendString("Wrong seconds");

UARTSend('\r');

UARTSend('\n');

break;

}

\_delay\_ms(100);

timeMode = 0;

break;

}

}

}

}

ISR(TIMER1\_COMPA\_vect)

{

if (timeMode == 0)

{

SS++;

if (SS == 60)

{

MM++;

SS = 0;

if (MM == 60)

{

HH++;

MM = 0;

if (HH == 24)

{

HH = 0;

}

if (HH > 12)

{

blink = (HH - 12) \* 2;

}

else if (HH == 0)

{

blink = 24;

}

else

{

blink = HH \* 2;

}

}

}

time = (HH \* 100) + MM;

NumToArr(time);

// PORTC = 0x00;

// PORTA = 0x00;

// CalculateTemp();

if (segMode == 1)

{

PORTC = 0x00;

PORTA = 0x00;

CalculateTemp();

}

}

if (blink != 0)

{

PORTD ^= RED | GREEN | BLUE;

blink--;

}

}

ISR(TIMER2\_COMP\_vect)

{

if (segMode == 0)

{

PORTC = 0x00;

PORTA = 0x00;

PORTC = DecToDigit(timeArr[j]);

if ((j == 2) && ((SS % 2) == 0) && (segMode == 0))

{

PORTC ^= 0b10000000;

}

PORTA = (1 << (7 - j));

j++;

j %= 4;

}

else

{

PORTC = 0x00;

PORTA = 0x00;

PORTC = DecToDigit(tempArr[j]);

if (j == 3)

{

PORTC |= 0b10000000;

}

PORTA = (1 << (7 - j));

j++;

j %= 4;

}

}

ISR(INT0\_vect)

{

timeMode++;

}

ISR(INT1\_vect)

{

segMode++;

if (segMode == 1)

{

PORTC = 0x00;

PORTA = 0x00;

CalculateTemp();

//TIMSK ^= (1 << OCIE2);

}

else

{

//TIMSK ^= (1 << OCIE2);

segMode = 0;

}

}

void TimerInit()

{

//TIMER1

TCNT1 |= 0;

OCR1A |= 15625;

TCCR1A |= (1 << FOC1A);

TCCR1B |= (1 << WGM12) | (1 << CS11) | (1 << CS10);

TIMSK |= (1 << OCIE1A);

//TIMER2

TCCR2 |= (1<<WGM21) | (1<<CS22) | (1<<CS21);

OCR2 = 2;

TIMSK |= (1 << OCIE2);

}

void CalculateTemp()

{

TIMSK ^= (1 << OCIE2);

TWBR = (mode == standard) ? 32 : 2;

TWSR &= ~(0b11 << TWPS0); // Clearing TWSP to 0

TWCR |= (1 << TWEN); // Enable TWI, generating the SCLK

\_delay\_ms(1);

twi\_master\_tx\_rx(0x44, sendData, 2, data, 6);

temp = (data[0] \* 256 + data[1]);

cTemp = ((double)(175 \* temp / 65535.0) - 45) \* 10;

TWBR = 0;

TWSR = 0;

TWCR = 0;

TIMSK ^= (1 << OCIE2);

}

void SendTime()

{

itoa(HH, string, 10);

SendString(string);

UARTSend(':');

itoa(MM, string, 10);

SendString(string);

UARTSend(':');

itoa(SS, string, 10);

SendString(string);

UARTSend('\r');

UARTSend('\n');

}

void SendTemperature()

{

itoa(tempArr[3], string, 10);

SendString(string);

itoa(tempArr[2], string, 10);

SendString(string);

SendString(".");

itoa(tempArr[1], string, 10);

SendString(string);

SendString(" C");

UARTSend('\r');

UARTSend('\n');

}

void TempTo7Seg()

{

PORTC = DecToDigit(tempArr[0]);

PORTA = (1 << 7);

PORTC = DecToDigit(tempArr[1]);

PORTA = (1 << 6);

PORTC = DecToDigit(tempArr[2]);

PORTA = (1 << 5);

PORTC = DecToDigit(tempArr[3]);

PORTA = (1 << 4);

}

void NumToArr(int numbr)

{

for (int k = 0; k < DIGITS; k++)

{

timeArr[k] = numbr % 10;

numbr /= 10;

}

}

void TempToArr(int tempr)

{

tempArr[0] = 12;

for (int k = 1; k < DIGITS; k++)

{

tempArr[k] = tempr % 10;

tempr /= 10;

}

}

unsigned char DecToDigit(unsigned char Dec)

{

unsigned char Digit;

switch(Dec)

{

case 0:

Digit = 0b01011111;

break;

case 1:

Digit = 0b00000110;

break;

case 2:

Digit = 0b00111011;

break;

case 3:

Digit = 0b00101111;

break;

case 4:

Digit = 0b01100110;

break;

case 5:

Digit = 0b01101101;

break;

case 6:

Digit = 0b01111101;

break;

case 7:

Digit = 0b00000111;

break;

case 8:

Digit = 0b01111111;

break;

case 9:

Digit = 0b01101111;

break;

case 10:

Digit = 0b01110111;

break;

case 11:

Digit = 0b01111100;

break;

case 12:

Digit = 0b01011001;

break;

case 13:

Digit = 0b00111110;

break;

case 14:

Digit = 0b01111001;

break;

case 15:

Digit = 0b01110001;

break;

default:

Digit = 0b00000000;

break;

}

return Digit;

}

uartlib.c

/\*

\* uartlib.c

\*

\* Created: 04.06.2024 22:10:43

\* Author: Administrator

\*/

#include "main.h"

#include "uartlib.h"

void UARTInit()

{

UCSRA = 0x00;

UCSRB |= (1 << RXEN) | (1 << TXEN);

UCSRC |= (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0);

UBRRL = 25;

}

void UARTSend(char Value)

{

while (!(UCSRA & (1 << UDRE)))

{

}

UDR = Value;

}

char UARTReceive()

{

while (!(UCSRA & (1 << RXC)))

{

}

return UDR;

}

void SendString(char\* str)

{

while(\*str != '\0')

{

UARTSend(\*str);

str++;

}

// //UARTSend(\*str);

//

// UARTSend('\r');

// UARTSend('\n');

}

TWI.c

//TWI.c

#include "main.h"

#include "TWI.h"

#define QUIT\_TWI\_OP { send\_stop(); return -1; }

typedef enum

{

/\* TWI Master Status Codes \*/

st\_start = 0x08,

st\_restart = 0x10,

st\_sla\_w\_ack = 0x18,

st\_sla\_w\_noack = 0x20,

st\_data\_w\_ack = 0x28,

st\_data\_w\_noack = 0x30,

st\_arb\_lost = 0x38,

st\_sla\_r\_ack = 0x40,

st\_sla\_r\_noack = 0x48,

st\_data\_r\_ack = 0x50,

st\_data\_r\_noack = 0x58,

} TwiStatus;

typedef enum

{

dir\_write,

dir\_read

} TwiOperation;

void twi\_init(TwiMode mode)

{

// 1 = output, 0 = input

DDRC &= ~0b00000011; // PC0 = SCL; PC1 = SDA

PORTC |= 0b00000011; // Internal pull-up on both lines

TWBR = (mode == standard) ? 32 : 2;

TWSR &= ~(0b11 << TWPS0); // Clearing TWSP to 0

TWCR |= (1 << TWEN); // Enable TWI, generating the SCLK

}

void twi\_shut(void)

{

TWCR &= ~(1 << TWEN); // Disable TWI

TWBR = 0;

TWSR &= ~(0b11 << TWPS0);

// 1 = output, 0 = input

DDRC &= ~0b00000011; // PC0 = SCL; PC1 = SDA

PORTC &= ~0b00000011; // Clear pull-up on both lines

}

static uint8\_t get\_status(uint8\_t status)

{

uint8\_t st;

while (!(TWCR & (1 << TWINT)))

;

if ((st = (TWSR & 0xF8)) == status)

return 0;

else

return st;

}

static int send\_start(uint8\_t status)

{

TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);

return get\_status(status);

}

static void send\_stop(void)

{

TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);

}

static int send\_data(uint8\_t data, uint8\_t status)

{

TWDR = data;

TWCR = (1 << TWINT) | (1 << TWEN);

return get\_status(status);

}

static int recv\_data(uint8\_t \*data, uint8\_t status, uint8\_t ack)

{

TWCR = (1 << TWINT) | (ack << TWEA) | (1 << TWEN);

if (get\_status(status) == 0)

{

\*data = TWDR;

return 0;

}

else

{

return -1;

}

}

int twi\_master\_tx(uint8\_t addr, uint8\_t \*data, int len)

{

int i;

if (send\_start(st\_start)) QUIT\_TWI\_OP;

if (send\_data((addr << 1) | dir\_write, st\_sla\_w\_ack)) QUIT\_TWI\_OP;

for (i = 0; i < len; i++)

{

if (send\_data(data[i], st\_data\_w\_ack)) QUIT\_TWI\_OP;

}

send\_stop();

return 0;

}

int twi\_master\_rx(uint8\_t addr, uint8\_t \*data, int len)

{

int i;

if (send\_start(st\_start)) QUIT\_TWI\_OP;

if (send\_data((addr << 1) | dir\_read, st\_sla\_r\_ack)) QUIT\_TWI\_OP;

for (i = 0; i < len - 1; i++)

{

if (recv\_data(&data[i], st\_data\_r\_ack, 1)) QUIT\_TWI\_OP;

}

if (recv\_data(&data[i], st\_data\_r\_noack, 0)) QUIT\_TWI\_OP;

send\_stop();

return 0;

}

int twi\_master\_tx\_rx(uint8\_t addr, uint8\_t \*tx\_data, int tx\_len, uint8\_t \*rx\_data,

int rx\_len)

{

int i;

if (send\_start(st\_start)) QUIT\_TWI\_OP;

if (send\_data((addr << 1) | dir\_write, st\_sla\_w\_ack)) QUIT\_TWI\_OP;

for (i = 0; i < tx\_len; i++)

{

if (send\_data(tx\_data[i], st\_data\_w\_ack)) QUIT\_TWI\_OP;

}

if (send\_start(st\_restart)) QUIT\_TWI\_OP;

if (send\_data((addr << 1) | dir\_read, st\_sla\_r\_ack)) QUIT\_TWI\_OP;

for (i = 0; i < rx\_len - 1; i++)

{

if (recv\_data(&rx\_data[i], st\_data\_r\_ack, 1)) QUIT\_TWI\_OP;

}

if (recv\_data(&rx\_data[i], st\_data\_r\_noack, 0)) QUIT\_TWI\_OP;

send\_stop();

return 0;

}

Висновки:

Під час виконання розрахунково-графічної роботи, були використані всі знання які були набуті з курсу «Проектування мікроконтролерних пристроїв в радіотехнічних комп’ютеризованих системах», а саме робота з реєстрами, кнопками, світлодіодами, I2C шиною та UART.

При виконанні була використана користувальницька бібліотека для роботи з I2C. Головна функція була написана повністю мною.

У результаті роботи ми отримали повністю робочий годинник, який показує температуру, та керується завдяки кнопок та UART.