МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



ЗВІТ

про виконання лабораторної роботи № 5

«Робота з шинами I2C та 1-Wire»

з дисципліни

«Вбудовані системи»

Виконала:

студентка групи ІТ-31

Щербак Л. В. Прийняв викладач:

Думич С.С.

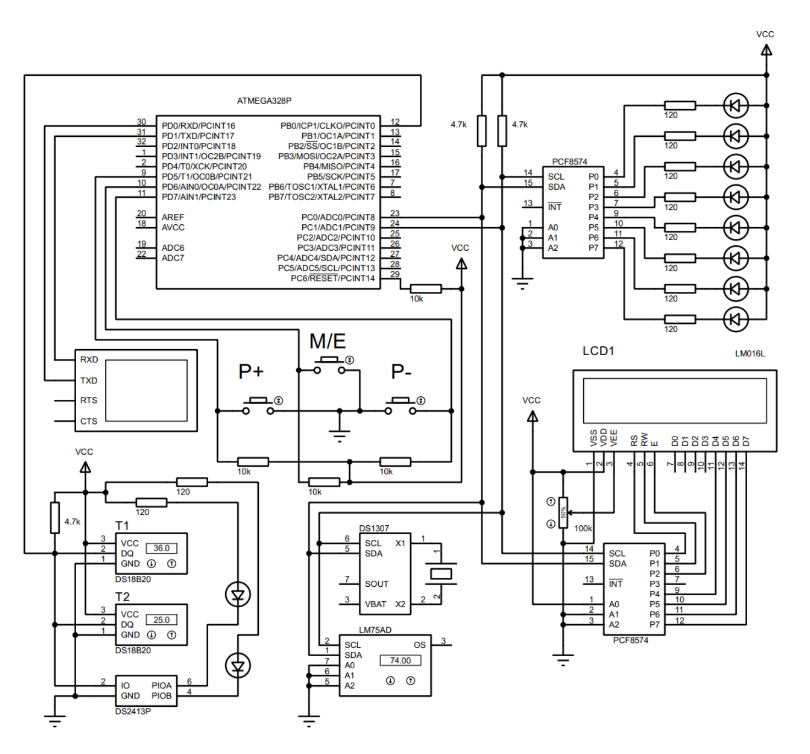
Львів 2022

Мета роботи: ознайомитися із принципами організації взаємодії мікроконтролера з різними пристроями, використовуючи інтерфейси I2C та 1-Wire.

Завдання до роботи:

1. Написати програму на мові Сі згідно з варіантом завдання.

2. Створити схему (Рис. 7) в програмі для моделювання (SimulIDE або Proteus 8). Провести моделювання написаної програми. Результати моделювання (скріншоти) добавити в звіт.



3. При наявності деталей зібрати відповідну до завдання частину схеми на макетній платі та запрограмувати мікроконтролер.

Варіанти завдань:

1.Написати програму яка виводить біжучий вогонь на світлодіодах підключених до I2C- експандера (PCF8574). Кнопки P+/P- збільшують/зменшують швидкість ефекту.

2. Модифікувати завдання 1. добавивши зміну ефекту кнопкою M/E та відображення на LCD індикаторі додаткової інформації (назва ефекту і його швидкість).

3. Використовуючи схему Рис 7. реалізувати термометр з виводом результату на LCD індикатор з періодом 5с. Для реалізації термометра використовувати мікросхему LM75AD.

4. Модифікувати завдання 3. замінивши мікросхему LM75AD на DS18B20.

5. Написати програму яка по черзі з періодом 1с засвічує і гасить світлодіоди підключені до мікросхеми DS2413P. (дана мікросхема підключена до мікроконтролера по шині 1-Wire)

6. Розробити 2-х канальний термометр з виводом даних на LCD індикатор використовуючи мікросхеми DS18B20. Оновлення даних відбувається кожні 2с.

7. (\*) Використовуючи мікросхему DS1307 (годинник реального часу) реалізувати «годинник» який відображає на LCD індикаторі поточну дату та час. Добавити можливість налаштування дати і часу з допомогою кнопок M/E, P+, P-.

8. (\*) Написати програму яка здійснює пошук пристроїв підключених до шини 1-Wire та результат їхніх-ROM ідентифікаторів виводить в консоль (UART).

Хід роботи:

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <util/delay.h>

#include <avr/io.h>

#define FISRT\_BUTTON PD5

#define SECOND\_BUTTON PD6

#define Third\_BUTTON PD7

#define ADDRESS\_FOR\_LED\_DISPLAY 0x42

#define ADDRESS\_FOR\_LED\_BULBS 0x40

#define SDA\_PIN 0

#define SCL\_PIN 1

#define I2C\_ACK 0

#define I2C\_NAK 1

#define I2C\_PORT PORTC

#define I2C\_PIN PINC

#define I2C\_DDR DDRC

#define LCD\_PIN\_E 4

#define LCD\_PIN\_RS 1

#define LCD\_PIN\_RW 2

#define LCD\_CMD 0

#define LCD\_DATA 1

#define LCD\_DISP\_CLEAR 0x01

#define LCD\_DISP\_OFF 0x08

#define LCD\_DISP\_ON 0x0C

#define LCD\_CURSOR\_ON 0x0E

#define LCD\_CURSOR\_BLINK 0x0F

#define LCD\_RETURN\_HOME 0x02

#define LCD\_ENTRY\_MODE 0x06

#define LCD\_4BIT\_MODE 0x20

#define LCD\_8BIT\_MODE 0x30

#define LCD\_2\_ROWS 0x08

#define LCD\_FONT\_5x8 0x00

#define LCD\_FONT\_5x10 0x04

#define LCD\_POSITION 0x80

#define SDA\_HIGH() I2C\_DDR &= ~(1<<SDA\_PIN)

#define SDA\_LOW() I2C\_DDR |= (1<<SDA\_PIN)

#define SCL\_HIGH() I2C\_DDR &= ~(1<<SCL\_PIN)

#define SCL\_LOW() I2C\_DDR |= (1<<SCL\_PIN)

#define NUMBER\_EFFECTS 3

#define NUMBER\_SPEED 3

#define SPEED\_LOW 310

#define SPEED\_MEDIUM 150

#define SPEED\_HIGH 90

#define VALUE\_RIGHT\_CENTER\_LED 16

#define VALUE\_LEFT\_CENTER\_LED 8

#define VALUE\_FIRST\_LED 1

#define VALUE\_LAST\_LED 128

enum Effects

{

FIRST\_EFFECT = 1,

SECOND\_EFFECT,

THIRD\_EFFECT

};

enum Speed

{

LOW = 1,

MEDIUM,

HIGH

};

typedef struct LedsConfiguration

{

uint32\_t speed;

uint8\_t effect;

} LedsConfiguration;

LedsConfiguration leds\_configuration = { 0, 0 };

void I2C\_Start(void)

{

I2C\_DDR &= ~((1<<SDA\_PIN)|(1<<SCL\_PIN));

\_delay\_us(5);

SDA\_LOW();

\_delay\_us(5);

SCL\_LOW();

}

void I2C\_Stop(void)

{

SDA\_LOW();

\_delay\_us(5);

SCL\_HIGH();

\_delay\_us(3);

SDA\_HIGH();

\_delay\_us(2);

}

uint8\_t I2C\_Write(uint8\_t data)

{

uint8\_t i, ack;

for (i = 0; i < 8; i++)

{

SCL\_LOW();

\_delay\_us(2);

if (data & 0x80)

{

SDA\_HIGH();

}

else

{

SDA\_LOW();

}

\_delay\_us(4);

SCL\_HIGH();

\_delay\_us(5);

data <<= 1;

}

SCL\_LOW();

\_delay\_us(3);

SDA\_HIGH();

\_delay\_us(5);

SCL\_HIGH();

\_delay\_us(3);

ack = I2C\_PIN & (1<<SDA\_PIN);

\_delay\_us(2);

SCL\_LOW();

\_delay\_us(5);

return ack;

}

uint8\_t I2C\_Read(uint8\_t ack)

{

uint8\_t i, data = 0;

for (i = 0; i < 8; i++)

{

data <<= 1;

SCL\_LOW();

\_delay\_us(5);

SCL\_HIGH();

\_delay\_us(5);

if (I2C\_PIN & (1<<SDA\_PIN))

{

data |= 1;

}

}

SCL\_LOW();

\_delay\_us(2);

if (!ack)

{

SDA\_LOW();

}

else

{

SDA\_HIGH();

}

\_delay\_us(3);

SCL\_HIGH();

\_delay\_us(5);

SCL\_LOW();

\_delay\_us(5);

return data;

}

void LCD\_E\_pulse(uint8\_t data)

{

I2C\_Write(data | LCD\_PIN\_E);

\_delay\_us(1);

I2C\_Write(data & ~LCD\_PIN\_E);

\_delay\_us(40);

}

void LCD\_Send(uint8\_t addr, uint8\_t d, uint8\_t type)

{

uint8\_t up\_nibble = (d & 0xF0);

uint8\_t low\_nibble = (d << 4);

if(type)

{

up\_nibble |= LCD\_PIN\_RS;

low\_nibble |= LCD\_PIN\_RS;

}

I2C\_Start();

I2C\_Write(addr);

I2C\_Write(up\_nibble);

LCD\_E\_pulse(up\_nibble);

I2C\_Write(low\_nibble);

LCD\_E\_pulse(low\_nibble);

I2C\_Stop();

\_delay\_ms(5);

}

uint8\_t InitializationOfLedDisplay(uint8\_t addr)

{

uint8\_t res = 0;

uint8\_t data = 0;

\_delay\_ms(16);

I2C\_Start();

res = I2C\_Write(addr);

if(res != I2C\_ACK)

{

return 1;

}

data = 0x30;

I2C\_Write (data);

LCD\_E\_pulse(data);

\_delay\_ms(5);

I2C\_Write (data);

LCD\_E\_pulse(data);

\_delay\_us(5);

I2C\_Write (data);

LCD\_E\_pulse(data);

\_delay\_us(5);

data = 0x20;

I2C\_Write (data);

LCD\_E\_pulse(data);

\_delay\_us(50);

I2C\_Stop();

LCD\_Send(addr, LCD\_4BIT\_MODE | LCD\_2\_ROWS | LCD\_FONT\_5x8, LCD\_CMD);

LCD\_Send(addr, LCD\_DISP\_OFF , LCD\_CMD);

LCD\_Send(addr, LCD\_DISP\_CLEAR, LCD\_CMD);

LCD\_Send(addr, LCD\_ENTRY\_MODE, LCD\_CMD);

LCD\_Send(addr, LCD\_DISP\_ON , LCD\_CMD);

return 0;

}

void LCD\_SetXY(uint8\_t addr, uint8\_t x, uint8\_t y)

{

if (y != 0) {

x += 0x40;

}

LCD\_Send(addr, (LCD\_POSITION | x), LCD\_CMD);

}

void LCD\_Clear(uint8\_t addr)

{

LCD\_Send(addr, LCD\_DISP\_CLEAR, LCD\_CMD);

\_delay\_ms(10);

}

void LCD\_Print(uint8\_t x, uint8\_t y, char \*str)

{

LCD\_SetXY(ADDRESS\_FOR\_LED\_DISPLAY, x, y);

while(\*str)

{

LCD\_Send(ADDRESS\_FOR\_LED\_DISPLAY, \*str++, LCD\_DATA);

}

}

void DisplayingInformationAboutEffectOnLedDisplay(void)

{

const uint8\_t MAX\_NUMBER\_SYMBOLS = 20;

const uint8\_t position\_word\_effect = 0;

const uint8\_t position\_word\_speed = 1;

const uint8\_t variable\_position = 8;

static uint8\_t check\_for\_effect = NUMBER\_EFFECTS + 1;

if(check\_for\_effect != leds\_configuration.effect)

{

char effect[MAX\_NUMBER\_SYMBOLS];

sprintf(effect, "%d", leds\_configuration.effect);

LCD\_Print(variable\_position, position\_word\_effect, effect);

check\_for\_effect = leds\_configuration.effect;

}

static uint8\_t check\_for\_speed = NUMBER\_SPEED + 1;

if(check\_for\_speed != leds\_configuration.speed)

{

char speed[MAX\_NUMBER\_SYMBOLS];

sprintf(speed, "%ld", leds\_configuration.speed);

LCD\_Print(variable\_position, position\_word\_speed, speed);

check\_for\_speed = leds\_configuration.speed;

}

}

uint8\_t WriteDataIntoPCF8574(uint8\_t adr, uint8\_t data)

{

I2C\_Start();

if(I2C\_Write(adr) != I2C\_ACK)

{

I2C\_Stop();

return 1;

}

I2C\_Write(~data);

I2C\_Stop();

return 0;

}

void ProgramDelay(void)

{

switch(leds\_configuration.speed)

{

case LOW:

\_delay\_ms(SPEED\_LOW);

break;

case MEDIUM:

\_delay\_ms(SPEED\_MEDIUM);

break;

case HIGH:

\_delay\_ms(SPEED\_HIGH);

break;

default:

\_delay\_ms(0);

}

}

void StartFirstEffectForLeds(void)

{

static uint16\_t value\_led\_right = VALUE\_RIGHT\_CENTER\_LED;

static uint16\_t value\_led\_left = VALUE\_LEFT\_CENTER\_LED ;

WriteDataIntoPCF8574(ADDRESS\_FOR\_LED\_BULBS, value\_led\_right | value\_led\_left);

value\_led\_right = value\_led\_right << 1;

value\_led\_left = value\_led\_left >> 1;

if( (value\_led\_left < VALUE\_FIRST\_LED) &&

(value\_led\_right > VALUE\_LAST\_LED) )

{

value\_led\_right = VALUE\_RIGHT\_CENTER\_LED;

value\_led\_left = VALUE\_LEFT\_CENTER\_LED ;

}

ProgramDelay();

}

void StartSecondEffectForLeds(void)

{

static uint16\_t value\_previous\_led = VALUE\_FIRST\_LED;

static uint16\_t value\_next\_led = VALUE\_FIRST\_LED << 1;

WriteDataIntoPCF8574(ADDRESS\_FOR\_LED\_BULBS, value\_previous\_led | value\_next\_led);

static bool direction = true;

if(value\_next\_led == VALUE\_LAST\_LED)

{

direction = false;

}

else if(value\_previous\_led == VALUE\_FIRST\_LED)

{

direction = true;

}

if(direction)

{

value\_previous\_led = value\_previous\_led << 1;

value\_next\_led = value\_next\_led << 1;

}

else

{

value\_previous\_led = value\_previous\_led >> 1;

value\_next\_led = value\_next\_led >> 1;

}

ProgramDelay();

}

void StartThirdEffectForLeds(void)

{

static uint16\_t value\_previous\_led = VALUE\_FIRST\_LED;

static uint16\_t value\_next\_led = VALUE\_FIRST\_LED << 1;

static uint16\_t value\_last\_led = VALUE\_LAST\_LED;

static uint16\_t value\_penultimate\_led = VALUE\_LAST\_LED >> 1;

WriteDataIntoPCF8574(ADDRESS\_FOR\_LED\_BULBS, value\_previous\_led | value\_next\_led |

value\_last\_led | value\_penultimate\_led);

value\_previous\_led = value\_previous\_led << 1;

value\_next\_led = value\_next\_led << 1;

value\_last\_led = value\_last\_led >> 1;

value\_penultimate\_led = value\_penultimate\_led >> 1;

if( (value\_previous\_led > VALUE\_LAST\_LED ) &&

(value\_last\_led < VALUE\_FIRST\_LED) )

{

value\_previous\_led = VALUE\_FIRST\_LED;

value\_next\_led = value\_previous\_led << 1;

value\_last\_led = VALUE\_LAST\_LED;

value\_penultimate\_led = value\_last\_led >> 1;

}

ProgramDelay();

}

void HandlerFirstButton(void)

{

++leds\_configuration.speed;

if(leds\_configuration.speed > NUMBER\_SPEED)

{

leds\_configuration.speed = 1;

}

}

void HandlerSecondButton(void)

{

++leds\_configuration.effect;

if(leds\_configuration.effect > NUMBER\_EFFECTS)

{

leds\_configuration.effect = 1;

}

}

void HandlerThirdButton(void)

{

--leds\_configuration.speed;

if(leds\_configuration.speed == 0)

{

leds\_configuration.speed = 3;

}

}

void FirstButtonScan (void)

{

static uint8\_t shreg;

shreg <<= 1;

if( (PIND & (1 << FISRT\_BUTTON)) != 0 )

{

shreg |= 1;

}

if( (shreg & 0x07) == 0x04 )

{

HandlerFirstButton();

}

}

void SecondButtonScan(void)

{

static uint8\_t shreg;

shreg <<= 1;

if( (PIND & (1 << SECOND\_BUTTON)) != 0 )

{

shreg |= 1;

}

if( (shreg & 0x07) == 0x04 )

{

HandlerSecondButton();

}

}

void ThirdButtonScan (void)

{

static uint8\_t shreg;

shreg <<= 1;

if( (PIND & (1 << Third\_BUTTON)) != 0 )

{

shreg |= 1;

}

if( (shreg & 0x07) == 0x04 )

{

HandlerThirdButton();

}

}

void StartDataInitialization(void)

{

//Atmega328 Data Initialization

DDRB = 0;

DDRC = 0;

DDRD = 0;

//Initialization of I2c Bus Interface

I2C\_DDR &= ~( (1 << SDA\_PIN) | (1 << SCL\_PIN) );

I2C\_PORT &= ~( (1 << SDA\_PIN) | (1 << SCL\_PIN) );

//Initialization of LedDisplay

InitializationOfLedDisplay(ADDRESS\_FOR\_LED\_DISPLAY);

//Initialisation Text for LedDisplay

LCD\_Print(0, 0, "Effect: 0");

LCD\_Print(0, 1, "Speed : 0");

}

int main(void)

{

StartDataInitialization();

while(1) {

//Start Button Scan

FirstButtonScan();

SecondButtonScan();

ThirdButtonScan ();

//Start Effect for Leds

switch(leds\_configuration.effect)

{

case FIRST\_EFFECT:

StartFirstEffectForLeds();

break;

case SECOND\_EFFECT:

StartSecondEffectForLeds();

break;

case THIRD\_EFFECT:

StartThirdEffectForLeds();

break;

}

//Displaying Information about Effect On LedDisplay

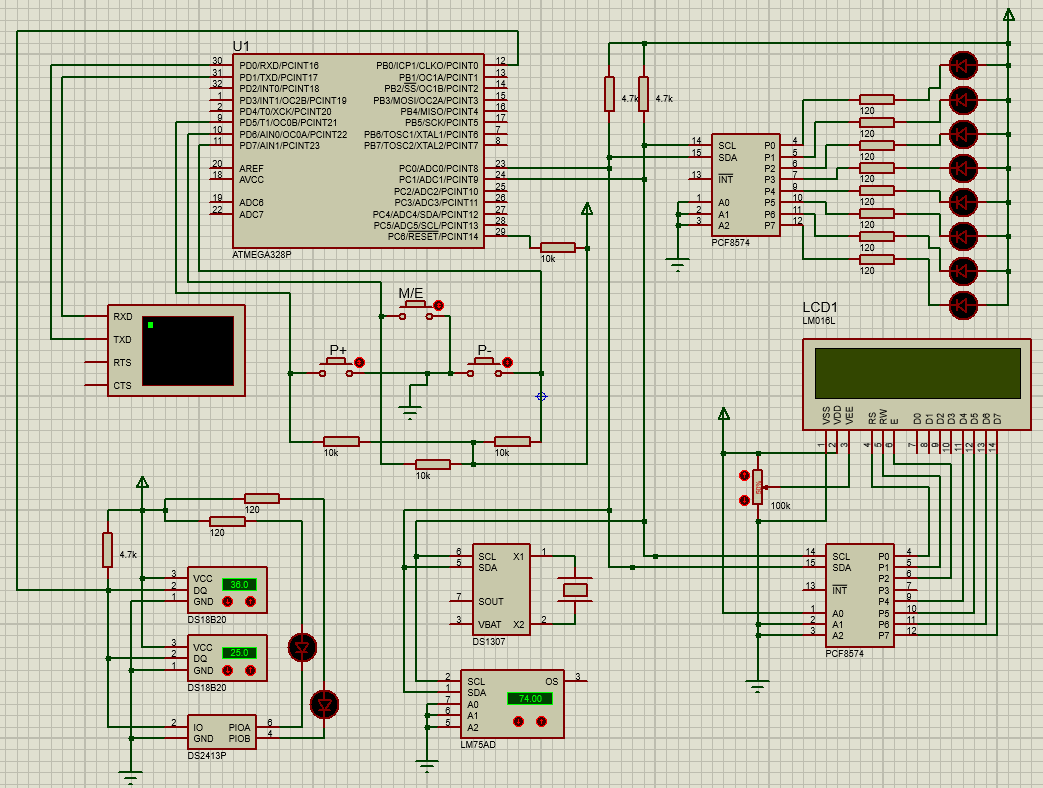
DisplayingInformationAboutEffectOnLedDisplay();

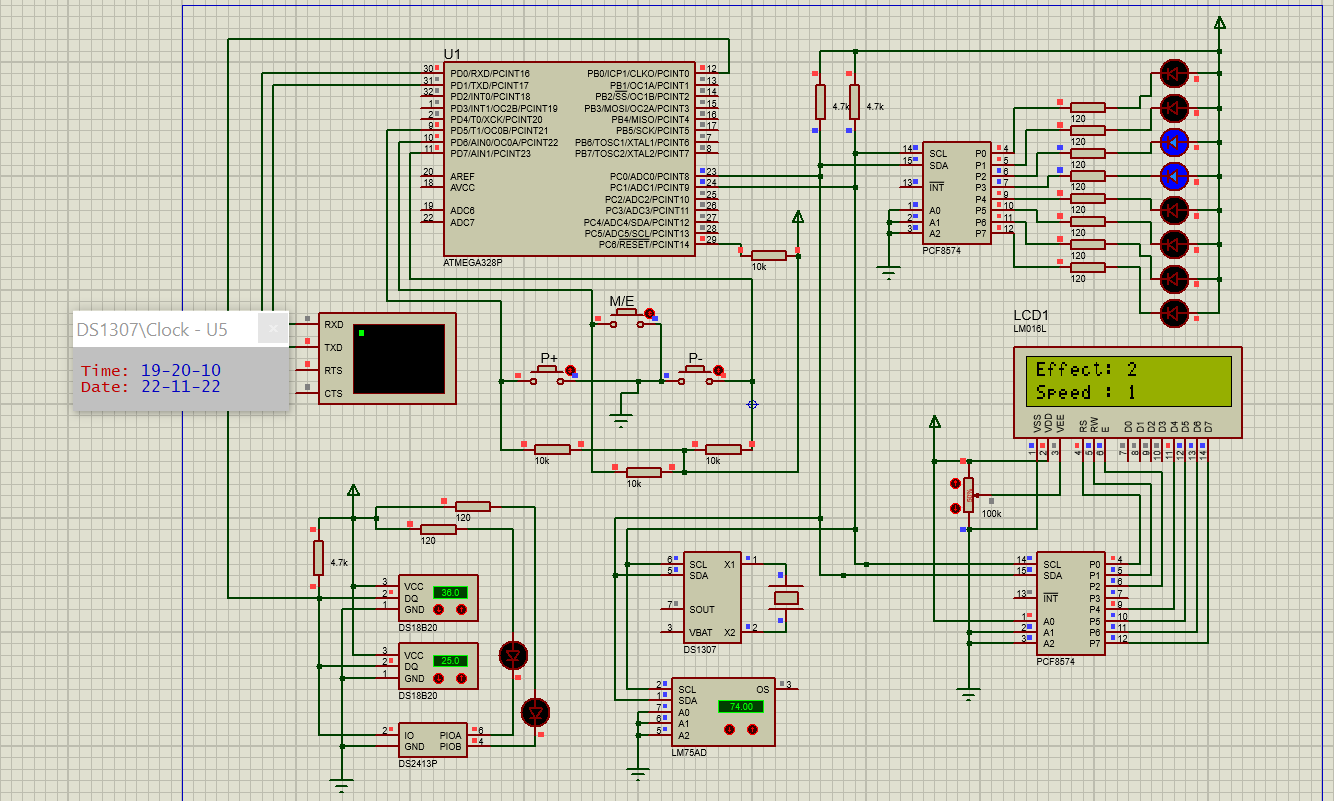
}

return 0;

}

Результат:





Висновок: ознайомилася із принципами організації взаємодії мікроконтролера з різними пристроями, використовуючи інтерфейси I2C та 1-Wire.