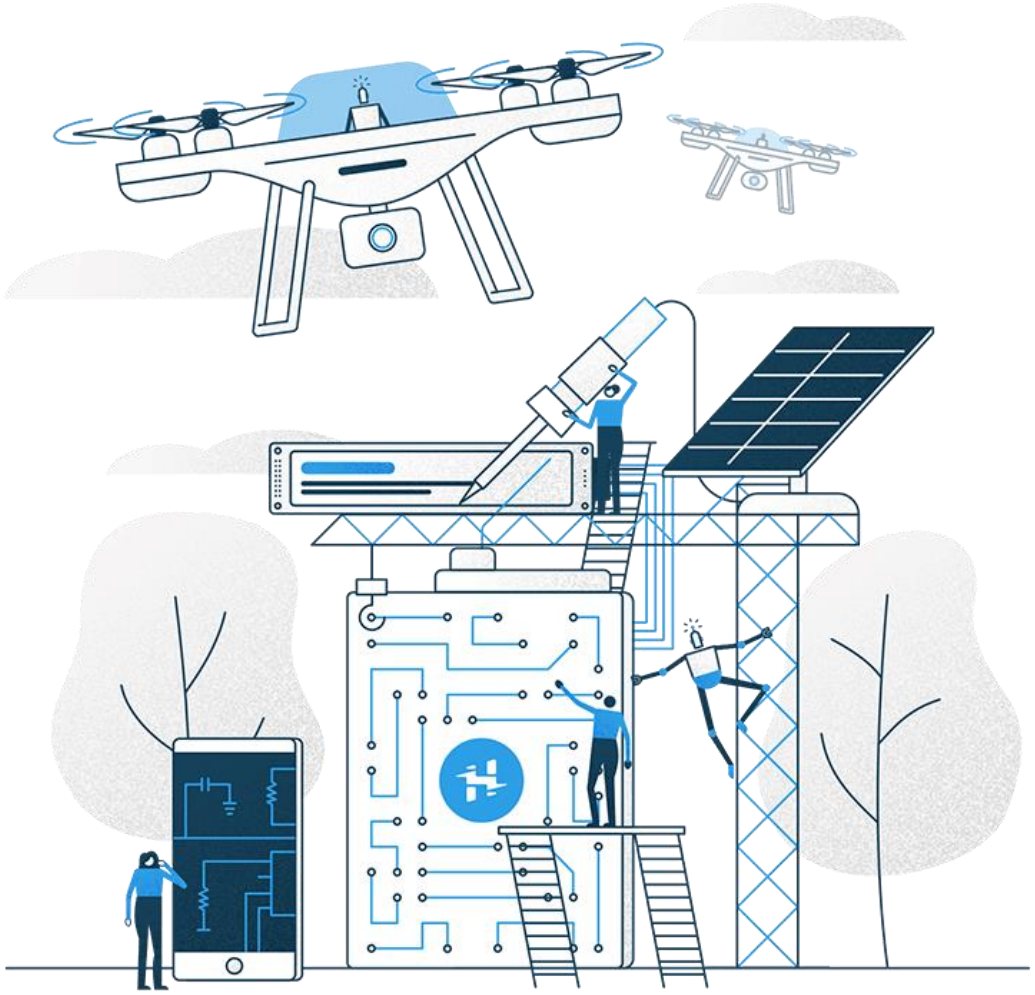
МІКРОПРОЦЕСОРНІ СИСТЕМИ УПРАВЛІННЯ



Lesson 15

Інтерфейс TWI (I2C)

Двопроводовий послідовний інтерфейс TWI вимагає тільки дві лінії зв'язку. Протокол TWI дозволяє проектувальнику системи зовні зв'язати до 128 різних пристроїв через одну двухпроводную двосторонню шину, де одна лінія – лінія синхронізації SCL і одна – лінія даних SDA. Всі пристрої, які підключені до шини, мають свої індивідуальні адреси, а механізм визначення вмісту шини підтримується протоколом.

Рисунок 1 – Система протоколу TWI

Як показано на рис. 1, обидві лінії шини підключені до шини живлення через навантажувальні (pull up) резистори. У всіх сумісних з TWI пристроях, як драйвер шини, використовуються транзистор або з відкритим стоком, або з відкритим колектором. Так реалізована функція, яка дуже важлива для двобічної роботи інтерфейсу. Низький логічний рівень на лінії шини TWI генерується, якщо один або більше з TWI-пристроїв виводить логічний 0. Високий рівень на лінії присутній, якщо всі TWI-пристрої перейшли у третій високоімпедансний стан, дозволяючи підтягуючим резисторам задати рівень логічної 1.

Інтерфейс TWI (I2C)

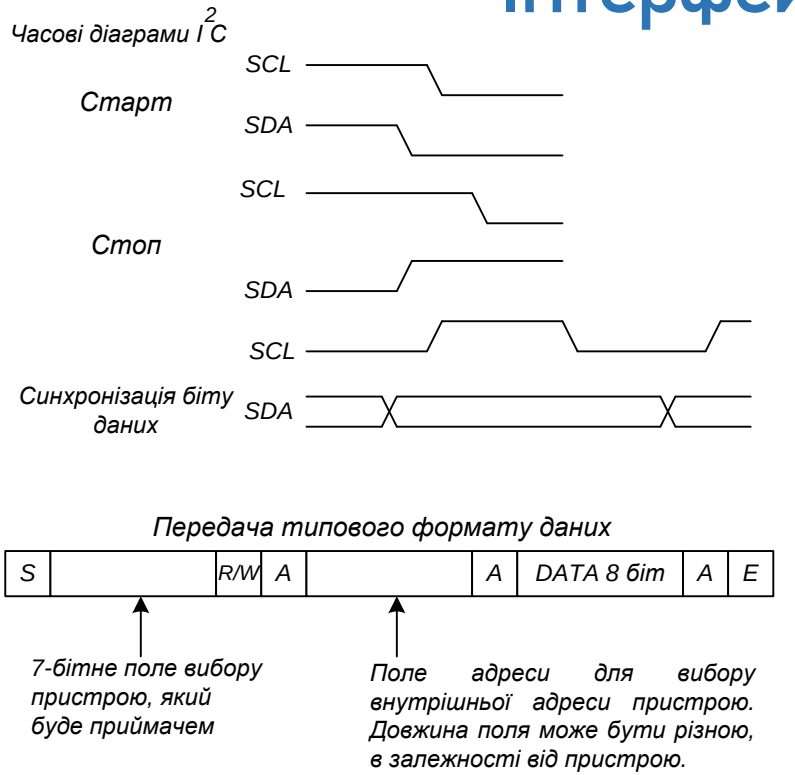


Рисунок 2 – Часові діаграми роботи шини I2C

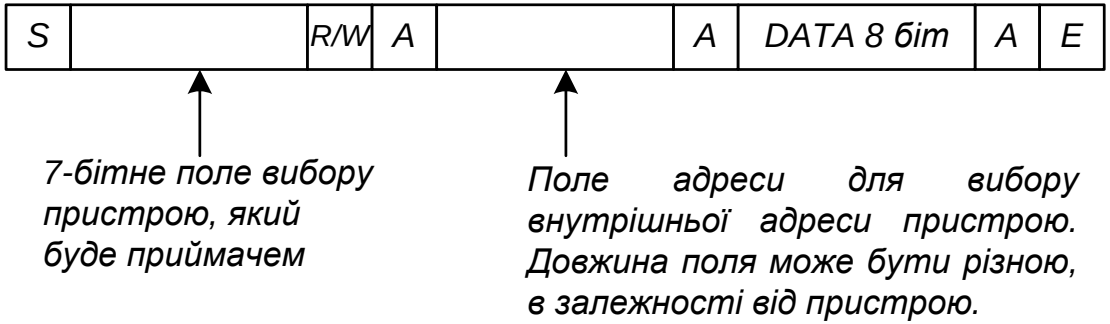
Кількість пристроїв, яка може бути підключено до однієї шини обмежується гранично допустимої ємністю шини (400 пФ) і 7-розрядним простором адрес. Підтримуються два різних набори технічних вимог, де один набір - для шин зі швидкістю передачі даних нижче 100 кГц, а інший дійсний для швидкостей понад 400 кГц.

Уся передача даних складається із стартової посилки, бітів і стопової посилки. Початок передачі визначається Start послідовністю - провал SDA при високому рівні SCL (рис. 2).

При передачі інформації від Master до Slave, Master генерує такти на SCL і видає біти на SDA. Які Slave зчитує коли SCL стає 1. При передачі інформації від Slave до Master, Master генерує такти на SCL і дивиться, що там Slave робить з лінією SDA – зчитує дані. А Slave, коли SCL йде в 0, виставляє на SDA біт, який Master зчитує коли підніме SCL назад. Закінчується обмін STOP послідовністю (при високому рівні на SCL лінія SDA переходить з низького на високий рівень. Перший пакет від Master до Slave – це фізична адреса пристрою і біт напрямку (рис. 2).

Інтерфейс TWI (I2C)

Передача типового формату даних



Сама адреса складається з семи біт (ось чому до 127 пристроїв на шині), а восьмий біт означає, що буде робити Slave на наступному байті – приймати або передавати дані. Дев'ятим бітом йде біт підтвердження ACK. Якщо Slave розпізнав свою адресу повністю, то на дев'ятому такті він переведе лінію SDA в 0, згенерувавши ACK – тобто зрозумів. Тоді Master продовжить передачу даних. Якщо Slave не відповів, тобто SDA на дев'ятому такті не буде переведено в 0 (не буде ACK), то майстер припинить свої спроби підєднатися.

Після адресного пакета йдуть пакети з даними в ту або іншу сторону, в залежності від біта R/W в заголовному пакеті.

На Arduino UNO, Nano, Pro Mini виводи I2C: контакт SDA – A4, контакт SCL – A5.

Бібліотека Wire.h дозволяє Arduino взаємодіяти з різними пристроями по інтерфейсу I2C / TWI. Всі функції бібліотеки Wire.h використовують 7-бітну адресацію, тим самим обмежуючи діапазон можливих адрес в межах 0 – 127.

Функції I2C / TWI на платах Arduino

Wire.begin(address) ініціалізує бібліотеку Wire і підключає Arduino до шини I2C в ролі ведучого (master) або веденого (slave) пристрою, де address: 7-бітова адреса Slave-пристрою (необов'язковий параметр); якщо адреса не вказана, то Arduino виступає в ролі Master-пристрою.

Wire.requestFrom (address, quantity) запрошує дані у Slave; використовується тільки Master. Після виклику requestFrom () запитувані дані повинні бути зчитані за допомогою функцій available () і read (), де address: 7-бітова адреса відомого пристрою, у якого запитуються дані; quantity: кількість запитуваних байт.

Wire.beginTransmission (address) починає процедуру передачі даних по I2C веденому пристрою з вказаною адресою. Для відправки даних, необхідно поставити їх в чергу функцією write () та здійснити передачу функцією endTransmission ().

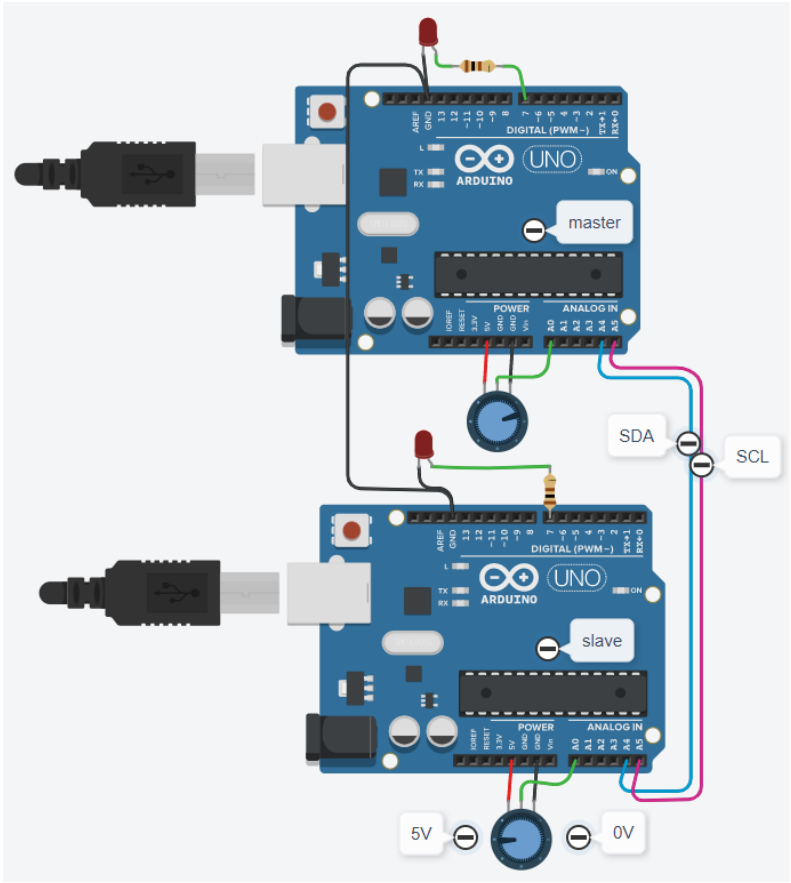
Wire.endTransmission () завершує процедуру передачі даних веденому пристрою, ініційовану функцією beginTransmission(). Функція відправляє байти, поставлені в чергу функцією write ().

Wire.write(value), Wire.write(string), Wire.write(data, length) повертає кількість записаних байт, де value: значення, яке необхідно відправити у вигляді 1 байта; string: рядок, який необхідно відправити у вигляді послідовності байт; data: масив даних, який необхідно відправити у вигляді декількох байт; length: кількість переданих байт.

Wire.available () повертає кількість байт, доступних для зчитування функцією read (). На Master пристрої, ця функція має викликатися після функції requestFrom (), а на Slave - всередині обробника onReceive ().

Wire.read () зчитує байт даних, отриманий Master від Slave (або навпаки) в результаті виконання функції requestFrom ().

Обмін даними по I2C

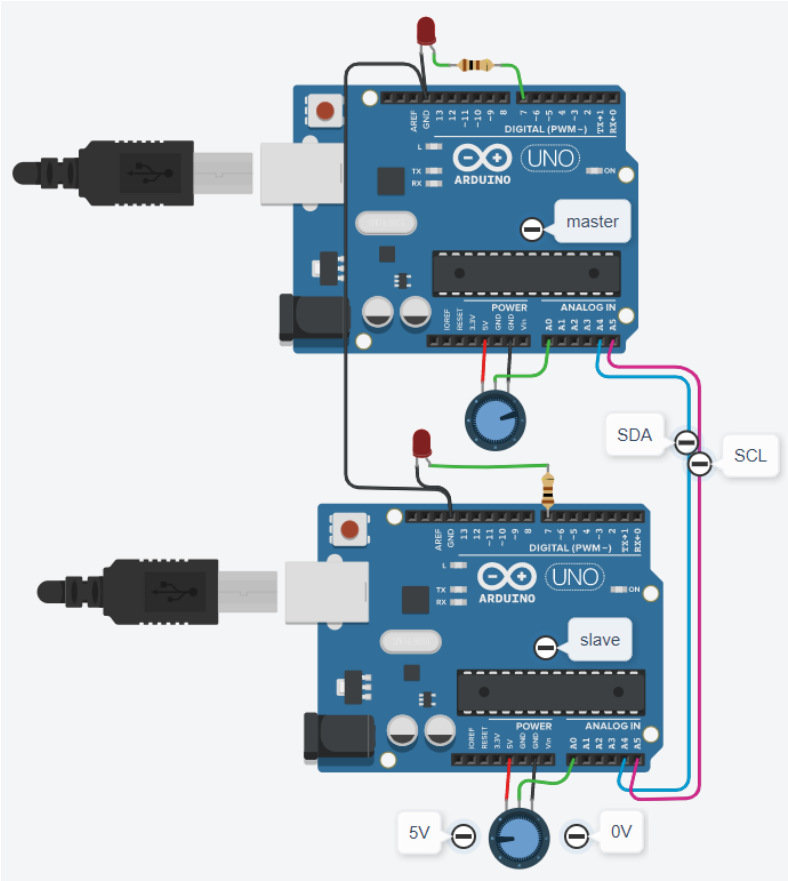



```
// MASTER
#include <Wire.h>

byte i2c_rcv;          // data received from I2C bus
int stat_LED;          // status of LED: 1 = ON, 0 = OFF
byte value_pot;        // potentiometer position

void setup() {
   Wire.begin(); // join I2C bus as Master
   // initialize global variables
   i2c_rcv = 255;
   stat_LED = 0;
   pinMode(7, OUTPUT);   // set pin 7 mode to output
}

void loop() {
    // read potentiometer position
    value_pot = analogRead(A0);   // read analog value at pin A0 (potentiometer voltage)
    // send potentiometer position to Slave device 0x08
   // informs the bus that we will be sending data to slave device 8 (0x08)
   Wire.beginTransmission(0x08);
   Wire.write(value_pot);       // send value_pot
   // informs the bus and the slave device that we have finished sending data
   Wire.endTransmission();
   Wire.requestFrom(0x08, 1);   // request potentiometer position from slave 0x08
   if(Wire.available()) {       // read response from slave 0x08
     i2c_rcv = Wire.read();
   }
   stat_LED = !stat_LED;
   delay (10*i2c_rcv);
   digitalWrite(7, stat_LED);
}
```

Відправлення даних по I2C включає три функції: Wire.beginTransmission(0x08) – інформує шину про те, що надсилатимемо дані та вказуємо адресу одержувача; Wire.write(value_pot) – передача даних; Wire.endTransmission() – звільняє мережу, щоб дозволити іншим пристроям спілкуватися через мережу.

Master пристрій також отримає положення потенціометра від Slave пристрою за допомогою команд Wire.requestFrom(0x08, 1) – адреса звідки та кількість байт, що будуть прийняті; Wire.available() – перевіряє чи є дані на шині; Wire.read() – читає дані з шини.

Обмін даними по I2C

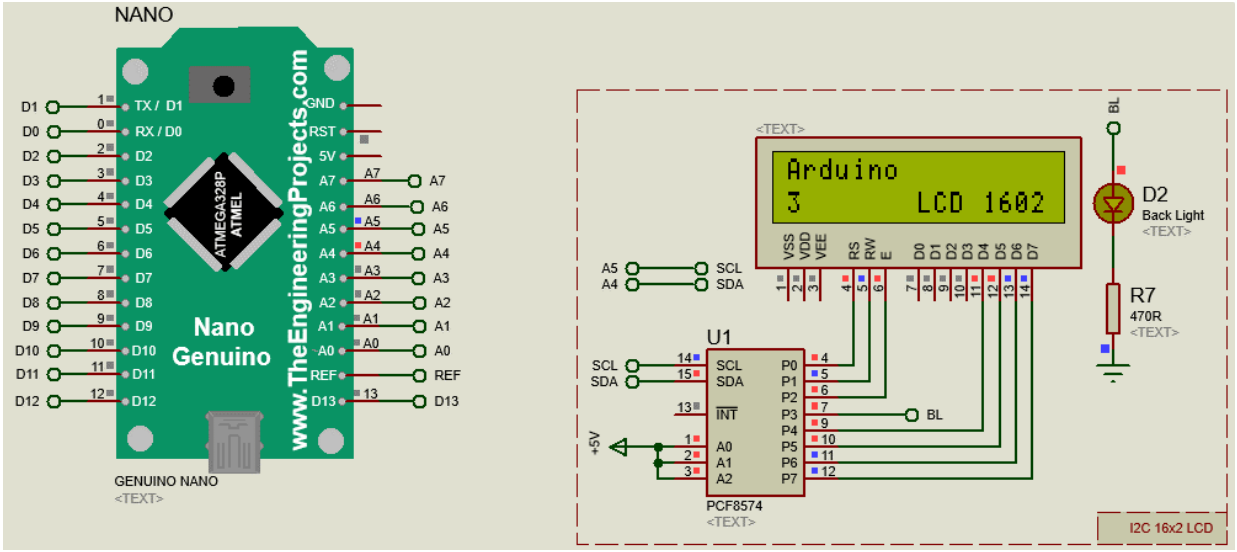



```
// SLAVE
#include <Wire.h>
byte i2c_rcv;            // data received from I2C bus
int stat_LED;            // status of LED: 1 = ON, 0 = OFF
byte value_pot;          // potentiometer position

void setup(){
  Wire.begin(0x08);        // join I2C bus as Slave with address 0x08
  // event handler initializations
  Wire.onReceive(dataRcv);   // register an event handler for received data
  Wire.onRequest(dataRqst);  // register an event handler for data requests
  // initialize global variables
  i2c_rcv = 255;
  stat_LED = 0;
  pinMode(7, OUTPUT);      // set pin 7 mode to output
}

void loop(){
   value_pot = analogRead(A0); // read analog value at pin A0 (potentiometer voltage)
   stat_LED = !stat_LED;
  delay (10*i2c_rcv);
  digitalWrite(7, stat_LED);
}
//received data handler function
void dataRcv(int numBytes) {
 while(Wire.available()) { // read all bytes received
    i2c_rcv = Wire.read();
 } }
// requests data handler function
void dataRqst() {  Wire.write(value_pot); // send potentiometer position }
```

Для Slave існує невелика різниця в кодуванні I2C-зв'язку. Перша різниця полягає в адресі Wire.begin(0x08) у setup(). Наступне завдання - додати до коду обробники подій для управління даними, отриманими з інших пристроїв в I2C мережі. У частині скетчу додаємо функцію Wire.onReceive(dataRcv) для реєстрації функції (обробника), яка керуватиме отриманими даними. Ім'я функції може бути будь-яким. Наступний обробник події - Wire.onRequest(dataRqst). Ця функція використовується на Slave пристроях та працює аналогічно. Єдина відмінність у тому, що вона обробляє події запиту даних. Запити даних надходять від основних пристроїв.

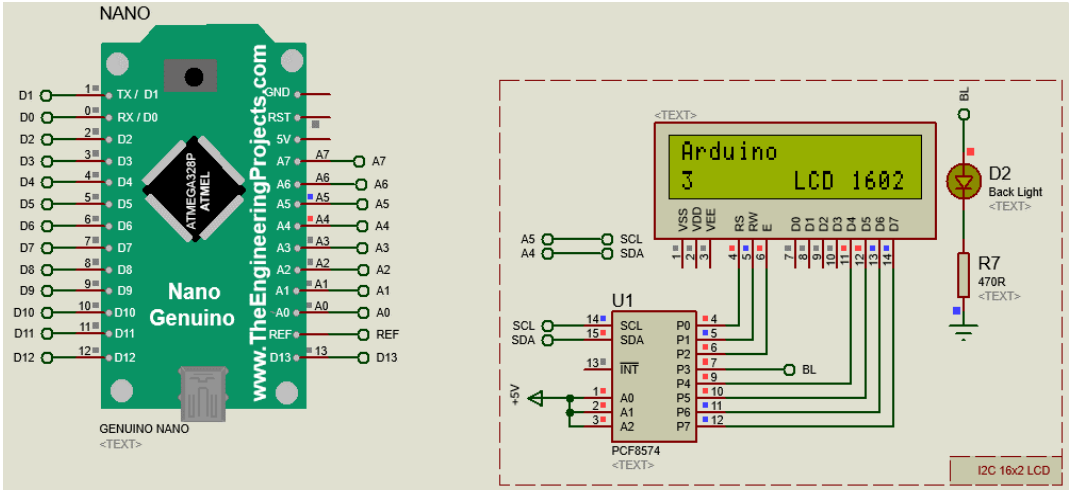
LCD індикатор з інтерфейсом I2C

Стандартна схема підключення LCD не завжди зручна, оскільки займає як мінімум 6 цифрових виходів Arduino. Тому популярними є LCD індикатори з конвертером в I2C на мікросхемі PCF8574. Для роботи з індикатором необхідна бібліотека LiquidCrystal_I2C.h.



LCD індикатор з конветором I2C PCF8547

LCD індикатор з інтерфейсом I2C



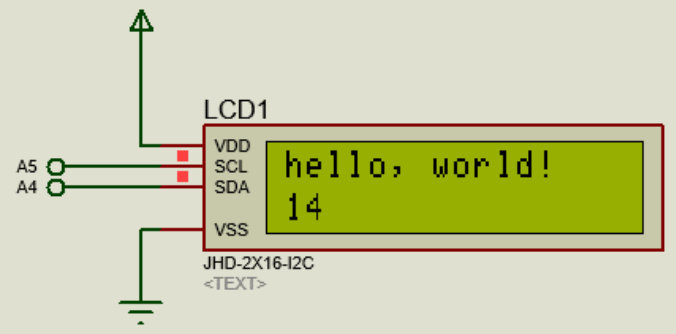

```
#include <Wire.h> // бібліотека для керування по I2C
#include "LiquidCrystal_I2C.h" // підключаємо бібліотеку для LCD1602

LiquidCrystal_I2C lcd(0x27,16,2); // конфігурація LCD індикатора

void setup()
{
  lcd.init();
  lcd.backlight();// Включаємо підсвітку дисплея
  lcd.print("Arduino");
  lcd.setCursor(8, 1);
  lcd.print("LCD 1602");
}
void loop()
{
  // Курсор на 2 рядок та 0 символ
  lcd.setCursor(0, 1);
  // Виводимо кількість секунд з моменту запуску Arduino
  lcd.print(millis()/1000);
}
```

1) Підключаємо бібліотеку:
#include <LiquidCrystal_I2C.h>
// Library for LCD
2) Створюємо Liquidcrystal_I2C об'єкт з I2C адресом, числом стовпчиків, числом рядків:
LiquidCrystal_I2C lcd(0x27, 16, 2);
// I2C address 0x27, 16 column and 2 rows

3) Ініціалізація LCD.
lcd.init();
//initialize the led
led.backlight();
//open the backlight
4) Встановлюємо курсор в початкову позицію:
led.setCursor(column_index, row_index);
5) Виводимо інформацію на LCD.
led.print("Helle World!").

LCD індикатор з інтерфейсом I2C

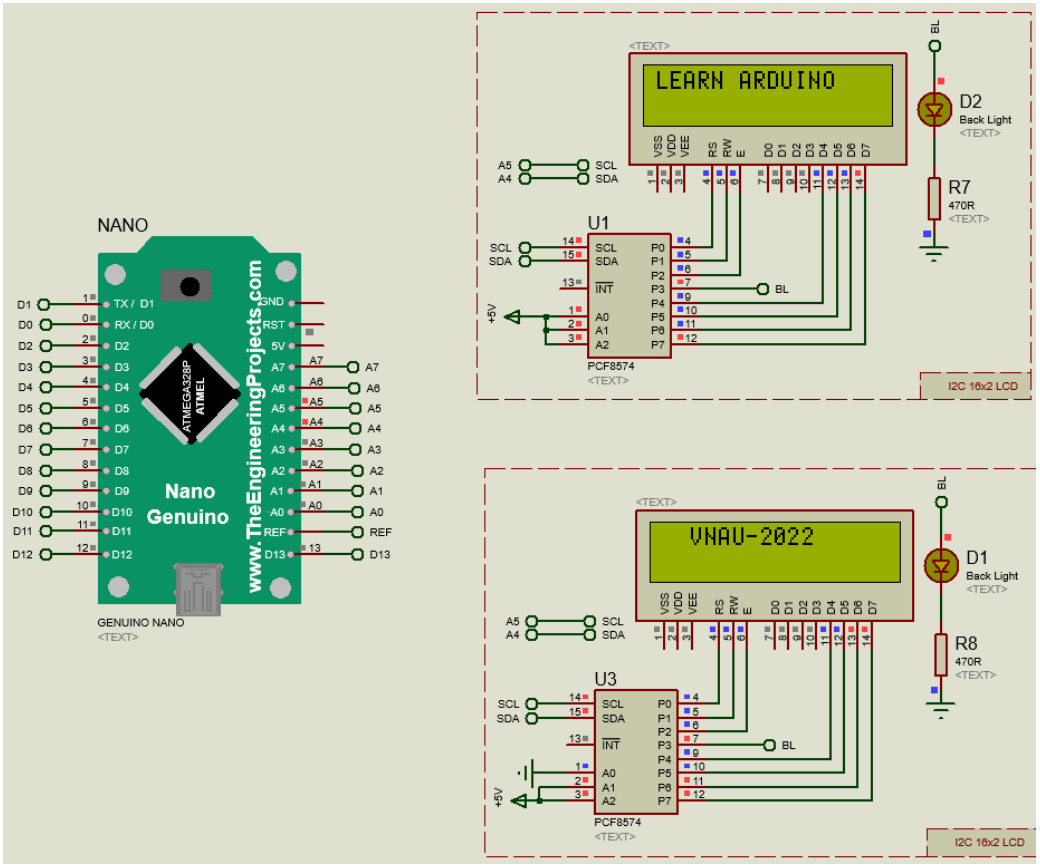
Для роботи з другими LCD_I2C індикаторами (jhd-2x16-i2c), в яких встановлений модуль керування NXP PCF2119x використовують бібліотеку DFRobot_LCD.h.




```
#include <Wire.h>
#include "DFRobot_LCD.h"

  const int colorR = 255;
  const int colorG = 0;
  const int colorB = 0;

DFRobot_LCD lcd(16,2);  //16 characters and 2 lines of show

void setup() {
    lcd.init();
    lcd.setRGB(colorR, colorG, colorB);
   //If the module is a monochrome screen, you need to shield it
    lcd.print("hello, world!");
    delay(1000);
}

void loop() {
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print(millis()/1000);
    delay(100);
}
```

Робота з 2 Slave по інтерфейсу I2C

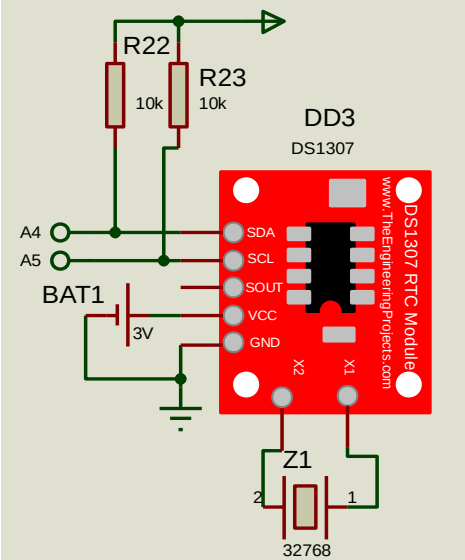
Розглянемо роботу 2 індикаторів, що підключені по шині I2C.




```
#include <Wire.h> // бібліотека для шини I2C
#include"LiquidCrystal_I2C.h" // бібліотека для 16x2 I2C

LiquidCrystal_I2C LCD1(0x27, 16, 2); // конфігурація LCD1
LiquidCrystal_I2C LCD2(0x26, 16, 2); // конфігурація LCD2

void setup() {
  LCD1.init(); // ініціалізація LCD1
  LCD2.init(); // ініціалізація LCD2
  LCD1.backlight(); // підсвітка -On
  LCD2.backlight(); // підсвітка -On
}

void loop() {
  // прокручування надпису на LCD1
  LCD1.setCursor(1, 0);
  LCD1.print("LEARN ARDUINO");
  LCD1.scrollDisplayLeft();
  // прокручування надпису на LCD2
  LCD2.setCursor(1, 0);
  LCD2.print("VNAU-2022");
  LCD2.scrollDisplayRight();
  delay(300);
}
```

DS1307

DS1307 – це годинник реального часу з екстремально точним ходом, завдяки вбудованому кварцовому резонатору з температурною компенсацією. Інтерфейс передачі даних – I2C. У мікросхемі є також вхід для підключення резервної батареї (рис. 8). При відключенні основного живлення мікросхема автоматично перемикається на роботу від резервної батареї, точність ходу від резервної батареї не порушується.

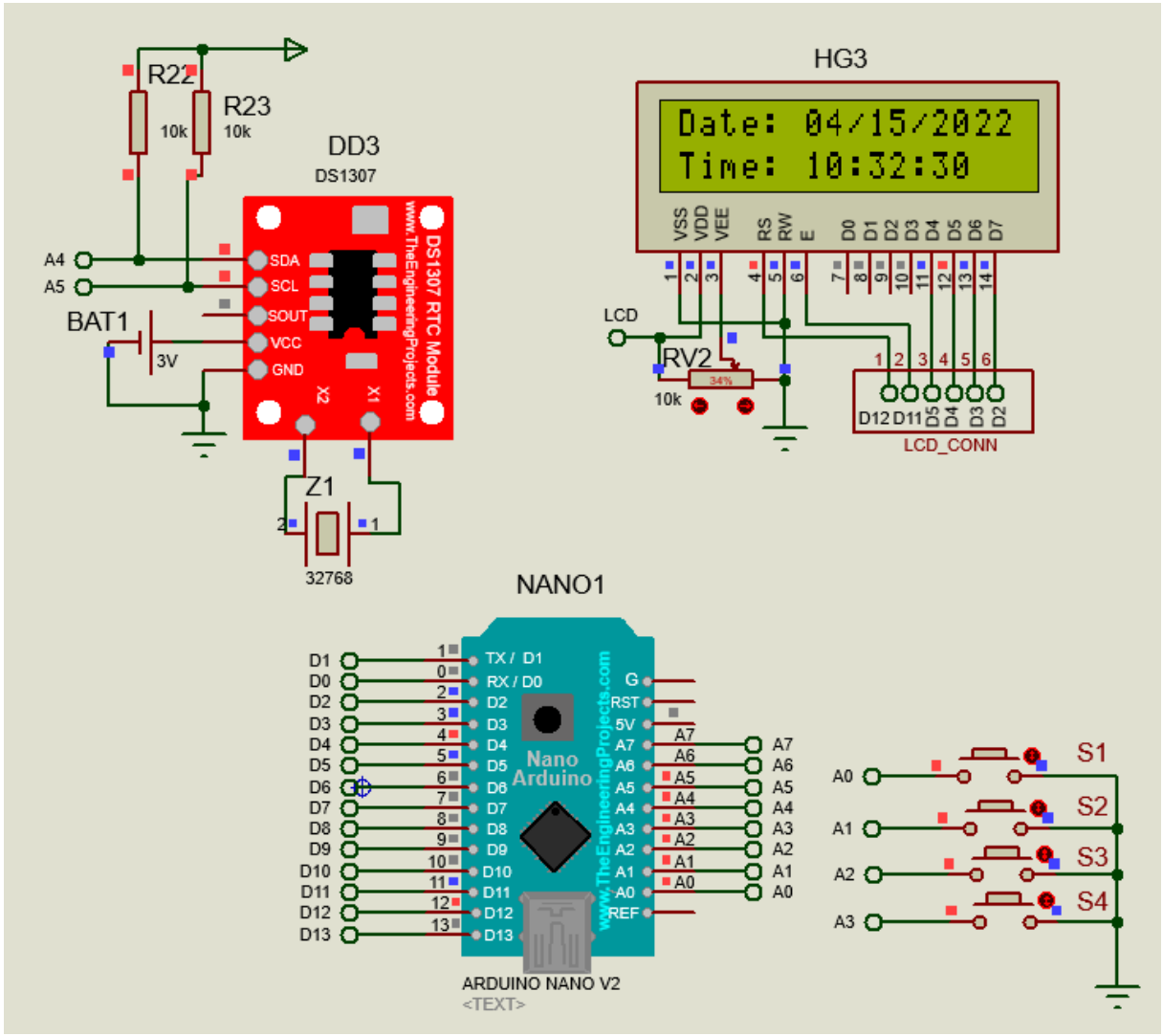


У DS1307 підтримується підрахунок секунд, хвилин, годин, днів місяця (дати), днів тижня, місяців і років (з урахуванням високосного року для місяців). Підтримується робота в 12 і 24 годинному форматі.

Для підключення DS1307 є декілька бібліотек: Wire.h, TimeLib, DS1307RTC.h, DS1307.h, iarduino_RTC.h. Бібліотеки універсальні (підходить для DS3231, DS1302). У прикладах використовується бібліотека DS1307.h, яку потрібно встановити на комп'ютер або додати в директорію з файлом проекту. Використовуються такі функції бібліотеки:

DS1307.begin() – ініціалізація роботи RTC модуля.
DS1307.getDate(clock) – отримання часу.
DS1307.setDate (Р, М, Д, ДТ, Г, Х, С) – встановлення часу (рік, місяць, день, день тижня, години, хвилини, секунди).

Робота з DS1307



Робота з DS1307

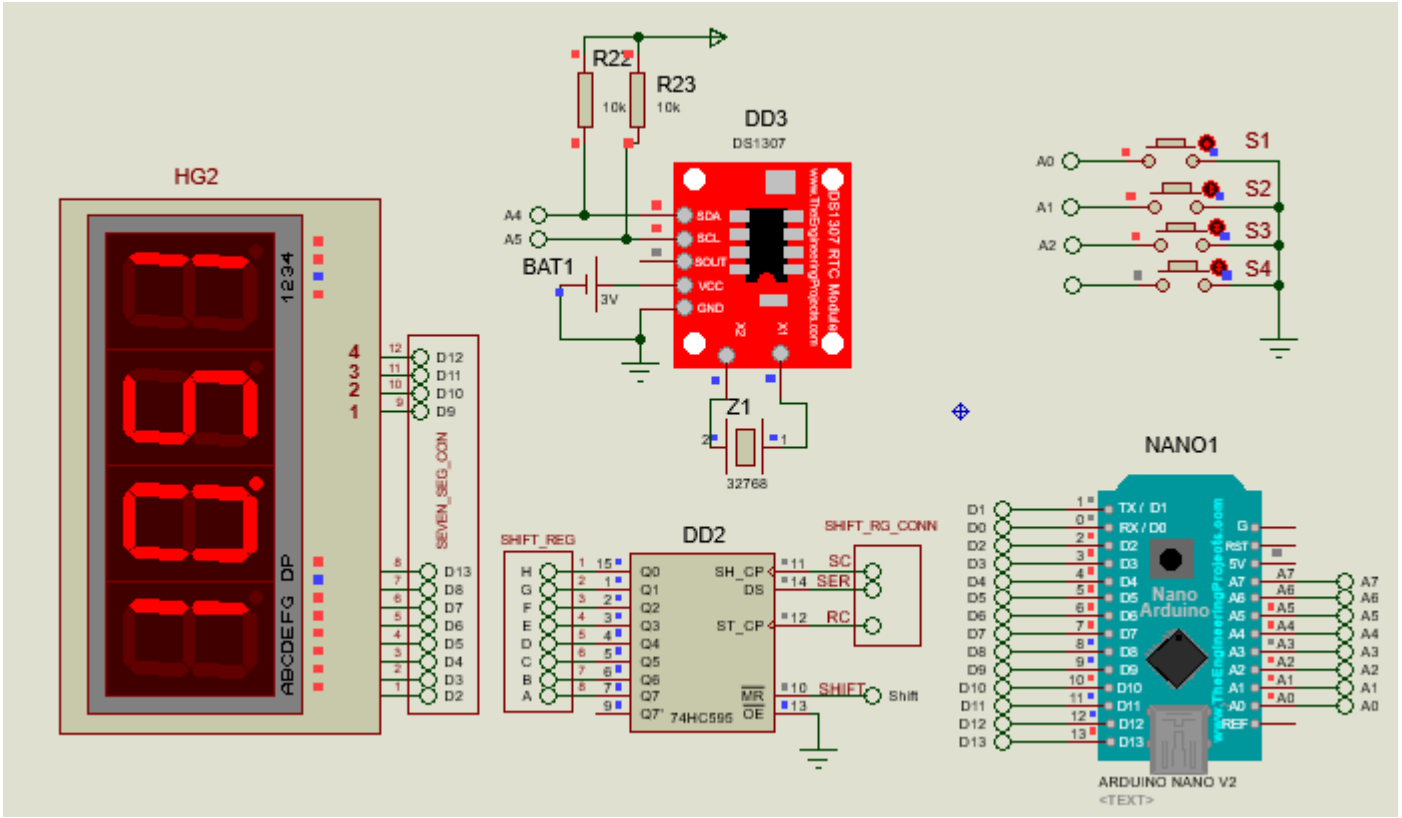
```
#include <LiquidCrystal.h>
#include "DS1307.h"
#include <Wire.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

byte SW0 = A0; //SET MINUTES
byte SW1 = A1; //SET HOUR
byte SW2 = A2; //SET DAY
byte SW3 = A3; //SET MONTH
//byte SW4 = A6; //SET YEAR
int clock[7];

void setup(){
  pinMode(SW0, INPUT);
  pinMode(SW1, INPUT);
  pinMode(SW3, INPUT);

  digitalWrite(SW0, HIGH); // pull-ups on
  digitalWrite(SW1, HIGH);
  digitalWrite(SW2, HIGH);
  digitalWrite(SW3, HIGH);
  lcd.begin(20,2);
  DS1307.begin();
}
```

```
void Print(int number){
  lcd.print(number/10);
  lcd.print(number%10);
}

void loop(){
  DS1307.getDate(clock);

  lcd.setCursor(0,1);
  lcd.print("Time: ");
  Print(clock[4]);
  lcd.print(":");
  Print(clock[5]);
  lcd.print(":");
  Print(clock[6]);
  lcd.setCursor(0,0);
  lcd.print("Date: ");
  Print(clock[1]);
  lcd.print("/");
  Print(clock[2]);
  lcd.print("/");
  lcd.print("20");
  Print(clock[0]);
```

Робота з DS1307

```
if(!digitalRead(SW0)){
 clock[5]++;
 if(clock[5]>59) clock[5]=0;
 DS1307.setDate(clock[0],clock[1],clock[2],0,clock[4],clock[5],clock[6]);
 delay(100);
}

if(!digitalRead(SW1)){
 clock[4]++;
 if(clock[4]>23) clock[4]=0;
 DS1307.setDate(clock[0],clock[1],clock[2],0,clock[4],clock[5],clock[6]);
 delay(100);
}

if(!digitalRead(SW2)){//dia
 clock[2]++;
  if(clock[2]>31) clock[2]=1;
 DS1307.setDate(clock[0],clock[1],clock[2],0,clock[4],clock[5],clock[6]);
 delay (100);
}

if(!digitalRead(SW3)){//mes
 clock[1]++;
 if(clock[1]>12) clock[1]=1;
 DS1307.setDate(clock[0],clock[1],clock[2],0,clock[4],clock[5],clock[6]);
 delay (100);
}

delay(100);
}
```

Робота з DS1307



Домашнє завдання

1) Установити поточне значення часу на годинник. Реалізувати програму, яка виводить на LCD-індикатор дату, час, прізвище автора та виводить ці значення в Монітор послідовного порту кожні 5 сек.
2) Реалізувати програму, яка виводить на LCD-індикатор поточний час та час будильника. Виставити значення часу та будильник. При спрацьовуванні будильника звучить тональний сигнал та загорається світлодіод.
3) Реалізувати програму, яка виводить на семисегментний індикатор час та дату (по черзі) з можливістю установки дати та часу.