МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИЛАДОБУДУВАННЯ

ПОБУДОВА ТА ДОСЛІДЖЕННЯ МП СИСТЕМ ІЗ ЗАСТОСУВАННЯМ ЦИФРОВИХ ДАТЧИКІВ

Методичні вказівки до лабораторних робіт з дисципліни:
«Мікропроцесорні системи»
(магістерській рівень)

Ужгород – 2023

УДК 004.056.55, 003.26
Мешко Р.О. Побудова та дослідження МП систем із застосуванням цифрових датчиків. Методичні вказівки до лабораторних робіт з дисципліни «Мікропроцесорні системи». Ужгород: в-во УжНУ, 2023, 36 с.

Укладач:
Роман Мешко – старший викладач кафедри приладобудування ДВНЗ «УжНУ»

Відповідальний за випуск – Михайло РЯБОЩУК., канд. фіз.-мат. наук, доцент кафедри приладобудування ДВНЗ «УжНУ».

Методичні рекомендації розглянуто та схвалено на засіданні кафедри приладобудування, протокол № 9 від 21.06.2023 року.

ЗМІСТ

ВСТУП

Метою вивчення навчальної дисципліни «Мікропроцесорні системи» є формування у студентів навичок підбору та визначення основних функцій управління технологічними об’єктами та основними компонентами; формуванні стійких знань з принципів побудови сучасних автоматизованих систем та застосування в них комп’ютерно-інтегрованих технологій; формування у студентів знань з апаратних та програмних мікропроцесорних засобів, необхідних для побудови сучасних цифрових систем автоматизації керування.

Лабораторна робота №1
Вивчення основ мікропроцесорної техніки на прикладі платформи ARDUINO, робота з портами вводу / виводу та створення тестового макету ІЧ охоронної системи

Мета роботи: використовуючи лабораторний набір на платформі Arduino побудувати робочий прототип ІЧ охоронної системи зі світловою (чи звуковою) тривожною індикацією руху. Створити графічно алгоритм роботи вказаного пристрою з використанням навичок використання цифрових входів / виходів та їх конфігурування, намалювати принципову електричну схему пристрою, ознайомитись із теоретичними даними ІЧ (IR – англ.) датчика руху - HC-SR501, та способами організації зовнішнього управління (як елемент зовнішнього інтерфейсу), написати тестову програму у середовищі Arduino IDE із використанням наявних цифрових входів/ виходів, на (DCC duino Nano, МК - ATmega328).

Навчитись використовувати різновиди функцій та команд мови С++ для створення тестової програми використовуючи освоєні у попередніх роботах алгоритми програмування.

Для складання схеми макету знадобляться наступні деталі, які є в кожному з лабораторних наборів Arduino:

1. Платформа Arduino Nano;
2. Кабель USB для програмування;
3. Макетна плата для прототипування;
4. З’єднувальні провідники;
5. ІЧ датчик руху для Arduino HC-SR501
6. Резисторы 1кОм – 1 шт;
7. Светлодіод червоний (Активний зумер, живлення+5В) – 1 шт;

Опис ІЧ датчика руху для Arduino HC-SR501

Рисунок 1.1. - Зовнішній вигляд ІЧ датчика руху

Рисунок 1.2. - Розміщення елементів налаштування та підключення ІЧ датчика руху

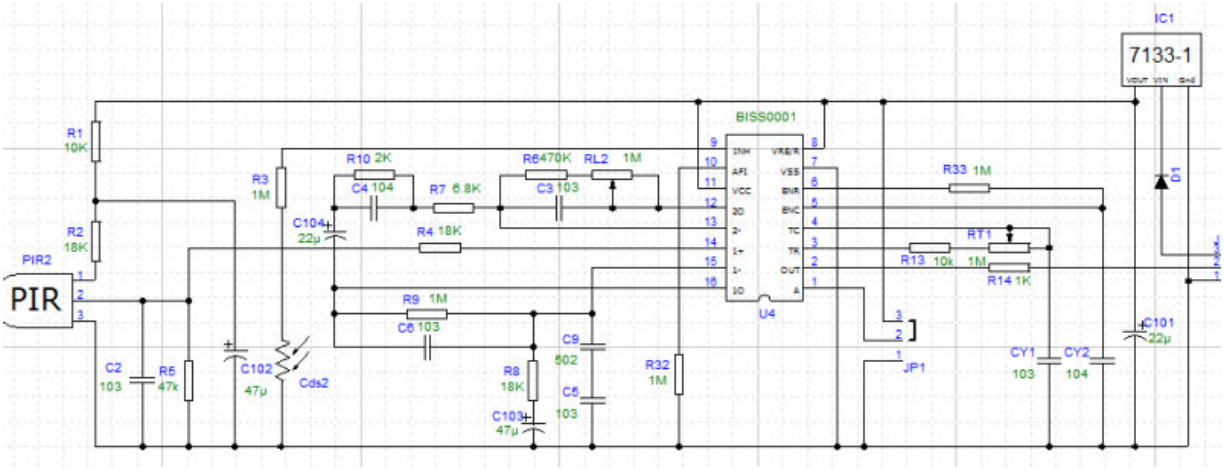


Рисунок 1.3. - Принципова схема ІЧ датчика руху - HC-SR501

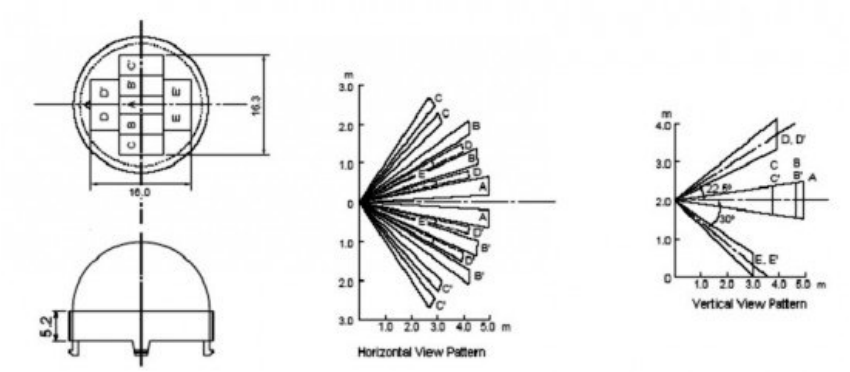
Інфрачервоний датчик руху та підключення до платформи Arduino та інших мікроконтролерів. Дозволяє виявляти рух людини або домашньої тварини на відстані до 7 метрів (чутливість можна регулювати). Має два входи живлення (+5 В і Спільний) та один цифровий вихід, за якого можна зчитувати дані. Якщо перешкод немає - на виході встановлено високий рівень (3.3В), якщо присутня перешкода – низький рівень (0В).

Налаштування модуля HC-SR501. Якщо перемичка встановлена в положення H, то на виході буде високий рівень весь час, поки датчик буде вловлювати рух, якщо в стан L, то

стан виходу буде перемикатись з високого на низький і назад приблизно раз в секунду. Для більшості проектів положення H зручніше, якщо ви керуєте пристроєм по фронту імпульсу, краще підійде положення L.

Характеристики ІЧ модуля:

- дальність виявлення: 0 - 7 м
- кут спрацьовування: 110 ° на дистанції до 7 м
- напруга живлення: 4.5 - 6 В
- вихідна напруга логічного рівня: 0 - 3.3 В
- час затримки: 0.3 - 18 секунд (регулюється)
- метод спрацьовування: L - не повторюємо перемикання; H - повторюване перемикання
- споживаний струм: 65 мА
- робочий діапазон температури: -20 - +50 град. Цельсія
- габаритні розміри: 32x24 мм



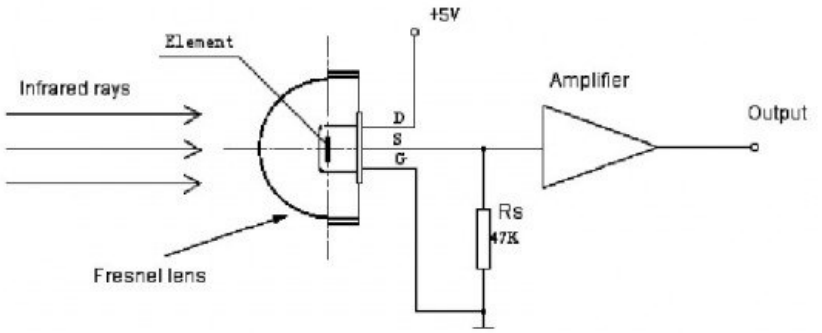


Рисунок 1.4. - Типова схема включення та діаграма направленості

Типове підключенню ІЧ модуля.

Будьте уважні з підключенням! Іноді підписи контактів приховані під лінзою Френеля (пластиковий ковпак). Лінзу можна акуратно зняти і побачити призначення контактів. Якщо виконати не правильне підключення, то датчик миттєво виходить з ладу і перестає реагувати на рух, хоча всі інші функції працюють.

Приклад скетча:

```
int ledPin = 13;
int switchPin = 2;
int value = 0;
void setup() {
        pinMode(ledPin, OUTPUT);
        pinMode(switchPin, INPUT);
}
void loop() {
        value = digitalRead(switchPin);
        if (HIGH == value) {
                digitalWrite(ledPin, HIGH);
        } else {
                digitalWrite(ledPin, LOW);
        }
}
```

Схема підключення сигналізую чого елемента - світло діода (або звукового динаміка)

Для реалізації свічення світлодіода, нам необхідно підключити його до одного з цифрових виходів Arduino які підтримують режим ШІМ. Нехай це буде лінія D10 (перелік

усіх виходів дивись у попередній роботі, або опис МК - ATmega328).

Катод світлодіода підключимо до мінуса (спільний), анод з'єднаємо з виходом D10 через струмообмежуючий резистор.

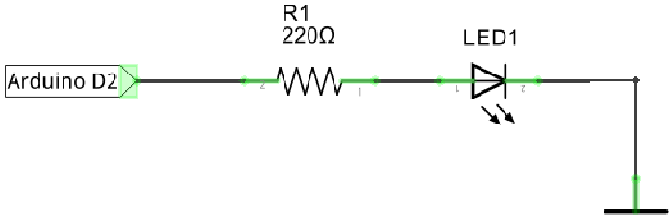


Рисунок 1.5. - Принципова схема підключення

Струмообмежуючий резистор виконує дві функції. По-перше, при підключенні до джерела живлення, світлодіод може безконтрольно пропустити через себе максимальний струму. Це може привести до виходу із ладу самого світлодіода і пошкодження виходів контролера. Звичайному світло діоду для свічення потрібно від 3-10 мА струму.

Другу функцію, яку виконує резистор - зниження напруги з 5 Вольт до 1.5 - 2 Вольт, що для багатьох стандартних світло діодів є робочою напругою.

Оцінимо номінал нашого струмообмежуючого резистора.

I = U/R;

R = U/I = (5В-2В)/5мА = 600 Ом

Розсіювана потужність резистора:

P=U*I=(5В-2В)*5мА=15мВт

Отже, ми з'ясували, що струмообмежуючий резистор для класичного вивідного світлодіода має номінал 600 Ом, розсіювана потужність 15мВт. Можемо використати зі стандартного ряду резисторів МЛТ – 610 Ом, 0,125 Вт, саме його ми і встановимо в схему.

Взагалі, якщо під рукою немає резистора на 610 Ом, можна використати будь-який інший більшого номіналу. Просто світлодіод буде горіти менш яскраво.

Програмний спосіб адаптації вхідного пристрою (аналогічно варіанту опитування клопки).

Дребезг, англ.. (Debounce - не чітке спрацювання контактів) призводить до того, що на вході мікрокотролера замість зміни стану з лог. 1. в лог. 0. при натисканні кнопки, ми отримаємо цілу серію імпульсів (як на осцилограмі вище). Щоб позбутися їх паразитного впливу потрібно виявити натискання кнопки, призупинити виконання програми і реалізувати деяку затримку. Час затримки необхідно вибрати таким чином, щоб він перевищував час дребезгу контактів. Таку ж процедуру затримки потрібно реалізувати і після виявлення відпускання кнопки.

Для більш наочного уявлення, що у нас один із двох режимів, ми зберемо простеньку схемку на 4 світлодіодах з керуванням однією кнопкою. При першому варіанті у нас по черзі спалахують з першого по четвертий світлодіоди. При другому варіанті те ж саме, але в зворотній послідовності, тобто з четвертого по перший. Кнопка у нас опитується на натискання або віджимання тільки перед початком ефекту. До тих пір, поки ефект не закінчить свою роботу, програма не реагує на натискання або віджимання кнопки.

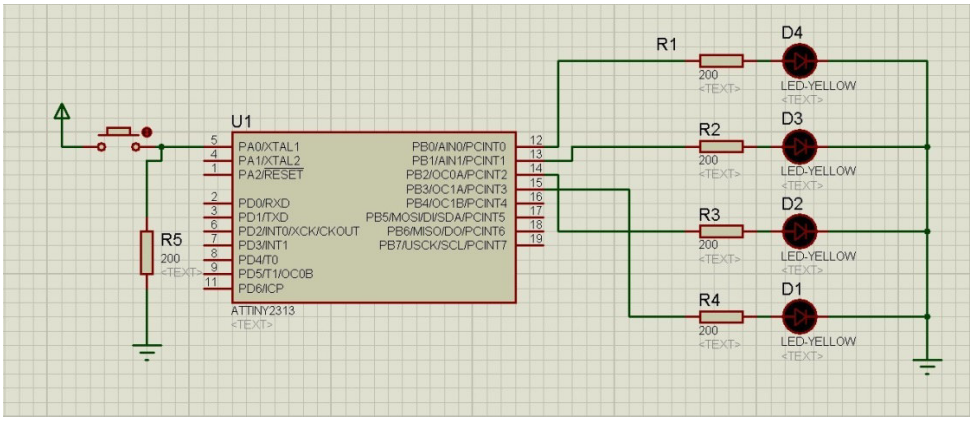


Рисунок 1.6. - Принципова схема прикладу простого інтерфейсу

У лівій частині схеми позначення кнопки і джерела живлення +5 вольт.

Із правої сторони - лінії від портів PA0, PB0, PB1, PB2 і PB3. До порту B у нас підключені світлодіоди, а до порту A - кнопка.

Номінал резистора може становити 200Ом – 10кОм. Отже, коли ми натискаємо кнопку, ми з'єднуємо порт PA0 з напругою +5 вольт живлення, і якщо ми опитуємо вивід PA0 на наявність напруги або її відсутність, ми зможемо впливати на виконання нашої програми.

Програмна реалізація опитування кнопки для приведеної нижче схеми.

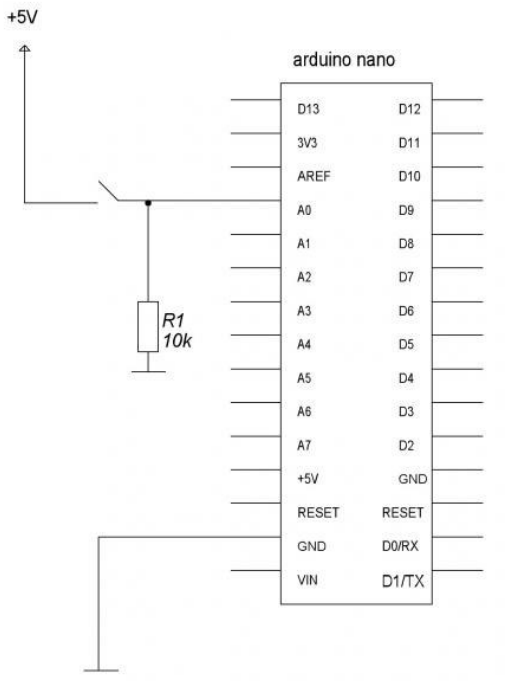


Рисунок 1.7. - Схема для тестового опитування кнопки

В даному випадку, коли кнопка відключена, вивід контролера буде підключений до спільного провідника через резистор, опір якого менший ніж внутрішній опір вивода МК.

Наводка (завада), що потрапила на пін, буде зашунтована на землю. Якщо ж підключений корисний сигнал (+ 5В) то він буде стікати в пін (незначна частина сигналу стече в землю через підтягуючий опір).

Розглянемо кілька простих програмних прикладів.

Заготовка програми з коментарями

```
void setup()
 {
   pinMode(13, OUTPUT);
    }
     void loop()
    {
 digitalWrite(13,analogRead(14)); //стан входу рівний стану виходу
 }
```

Ускладнимо програму

```
void setup()
{
    pinMode(13, OUTPUT);
    }
    void loop()

     {
     if(digitalRead(14)==HIGH)//якщо вхід  активний ...
     {
     digitalWrite(13,HIGH);//вмикаємо світлодіод
   }
   else//якщо вхід не активний...
{
 digitalWrite(13,LOW);//вимикаємо
 }
```

```
}
```

Цей приклад можна інвертувати. При спрацюванні по входу світлодіод горить, коли не активному вході - гасне.

Наступний приклад.

```
void setup()
 {
    pinMode(13, OUTPUT);
  }
   void loop()
    {
     if(digitalRead(14)==HIGH)//якщо вхід спрацював ...
      {
       digitalWrite(13,!digitalRead(13));//інвертуємо стан піна

    delay(500);//невеликий захист від "дребезга" по входу
    }
}
```

Якщо утримувати певний час активним вхід, то отримаємо миготіння світлодіода з частотою заданою функцією delay.

Якщо нам потрібно включити по утриманню активного входу.

Для цього введемо змінну val.

```
int val=0;
   void setup()
   {
    pinMode(13, OUTPUT);
    }
     void loop()
 {
    if(digitalRead(14)==HIGH)//якщо активно по входу ...
    {
```

```
      val++;// додаємо до змінної 1 при кожній зміні циклу (інкремент).

    delay(500);// невеликий захист від "дребезга" по входу спрацювання
    }
    else
    {
     val=0;
    }
    if(val>=5)
    {
     digitalWrite(13,!digitalRead(13));//інвертуємо стан піна
     val=0;
    }
 }
```

Тепер розглянемо варіант, коли нам потрібно зробити лише одну дію за одним спрацюванням по входу (попередні програми циклічно повторювали дію при утриманні активного входу)

Для цього введемо нову змінну flag (назви змінних можуть бути якими завгодно).

```
int flag=0;

void setup()
{
    pinMode(13, OUTPUT);
  }
    void loop()
 {
   if(digitalRead(14)==HIGH&&flag==0)// якщо вхід активний
    // і змінна flag дорівнює 0, то ...
```

```
    {

      digitalWrite(13,!digitalRead(13));
      flag=1;
       // це потрібно для того що б з кожним спрацюванням по
входу
        // відбувалася тільки одна дію
        // також захист від "дребезга" 100%   по входу
    }

     if(digitalRead(14)==LOW&&flag==1)// якщо вхід НЕ активний
     // і змінна flag дорівнює - 1, то ...     {

       flag=0;//обнулюємо змінну flag
    }
}
```

Алгоритм наступний. Натиснули спрацював вхід - загорівся світлодіод, утримуємо активним вхід - нічого не відбувається (світлодіод горить). Відпустили вхід (пропала перешкода на датчику) - світлодіод горить.

Для того щоб погасити світлодіод потрібно повторно спрацювати по входу.

Тобто по аналогії із кнопкою, натиснули - горить, відпустили, натиснули - не горить.

Можна інвертувати дію ось так

```
int flag=0;

 void setup()
 {
   pinMode(13, OUTPUT);
 }
 void loop()
```

```
{
   if(digitalRead(14)==HIGH&&flag==0)// якщо вхід активний
    // і змінна flag рівна 0, то ...
   {
     flag=1;
      // це потрібно для того що б з кожним спрацюванням
       // відбувалася тільки одна дія
       // плюс захист від "дребезгу" 100% по входу
   }

    if(digitalRead(14)==LOW&&flag==1)// якщо вхід НЕ
спрацював
    // і змінна flag рівна - 1, то ...
   {
      digitalWrite(13,!digitalRead(13));
     flag=0;//обнулюємо змінну flag
   }
}
```

Завантажуємо програму на Arduino, і спостерігаємо за роботою макета.

При необхідності відлагоджуємо роботу програми та конструкції у цілому.

Індивідуальне завдання

Для кожного досліджуваного макету написати скетч з різними варіантами спрацювання на перешкоду ІЧ датчика (еквівалентно для варіанту відладки спрацюванню від кнопки).

Варіанти виконання завдань:

1. Створити тестовий скетч при одиночному спрацюванні від ІЧ датчика проблимати світло діодом 3 рази із затримкою в 1сек.

2. Створити тестовий скетч при першому спрацюванні від ІЧ датчика запалити світло діод, при наступному (другому спрацюванні) погасити світло діод.

3. Створити тестовий скетч при першому спрацюванні від ІЧ датчика запалити світло діод, при наступному (другому спрацюванні) погасити світло діод із варіантом ініціалізації лічильника числа циклів.

4. Створити тестовий скетч при одиночному спрацюванні від ІЧ датчика більше 1сек. Засвітити світло діод, у противному випадку з часом спрацювання менше 1сек. ігнорувати спрацювання.

5. Запропонувати на основі існуючого макету свої варіанти технічних рішень.

Лабораторна робота №2
Організація обміну інформацією МП системита символьного LCD дисплея та платформи Arduino Nano. Освоєння принципів побудови інтерфейсу

Мета роботи: використовуючи наявну платформу Arduino Nano на практиці освоїти принципи побудови мінімального інтерфейсу застосовуючи символьний дисплейWH1602 (організація16 - символів 2 рядки (16x2), паралельний інтерфейс 4, 8 – бітовий протокол, внутрішній драйвер HD44780), ознайомитись з принципами обміну МК та зовнішнього пристрою,використовуючи порти введення/ виведення цифрової інформації.

Для складання схеми макету знадобляться наступні деталі, які є в кожному з лабораторних наборів Arduino:

1. Платформа Arduino Nano;
2. Кабель USB для програмування;
3. Макетна плата для прототипування;
4. З’єднувальні провідники;

5. Резистори 1кОм – 5 шт;
6. Перехідник на шину I2C на базі PCF8574.

Використовуючи набуті навички з основ програмування, побудувати алгоритм та реалізувати його із використанням мови високого рівня C++ програмного продукту – Arduino IDE.

Вивчення основних принципів застосування драйвера обладнання на приклад і застосування бібліотеки LCD дисплея.

Створення тестової програми відображення інформаційного повідомлення на дисплеї «Hello, world!».

Організація лічильника числа вхідних імпульсів із подальшим інкрементом даних на дисплеї, та різні варіації поставленого завдання.

Практична робота. Використовується для тестування на працездатність конструкції, коректність роботи програмного забезпечення застосовуючи програми моделювання – Proteus.

Загальні відомостіпро символьний LCD дисплей

Розглянемо основні застосування поширеного знако-символьного LCD дисплея.

У якості навчального прикладу використаємо LCD символьний дисплей (1602) з керуванням по паралельній шині (4, або 8 біт) організацією управління.

Для побудови мінімального інтерфейсу зв’язку, достатньо під’єднати наступні виводи LCD дисплея : RS, Enable, d4, d5, d6, d7.

Працювати з подібними дисплеями можна через бібліотеку Liquid Crystal, програмного середовища Arduino IDE використовуючи певний набір функцій.

Головна особливість застосування бібліотек є швидке створення прототипу програмного забезпечення та макету

пристрою у цілому. Таким чином заощаджується час на розробку.

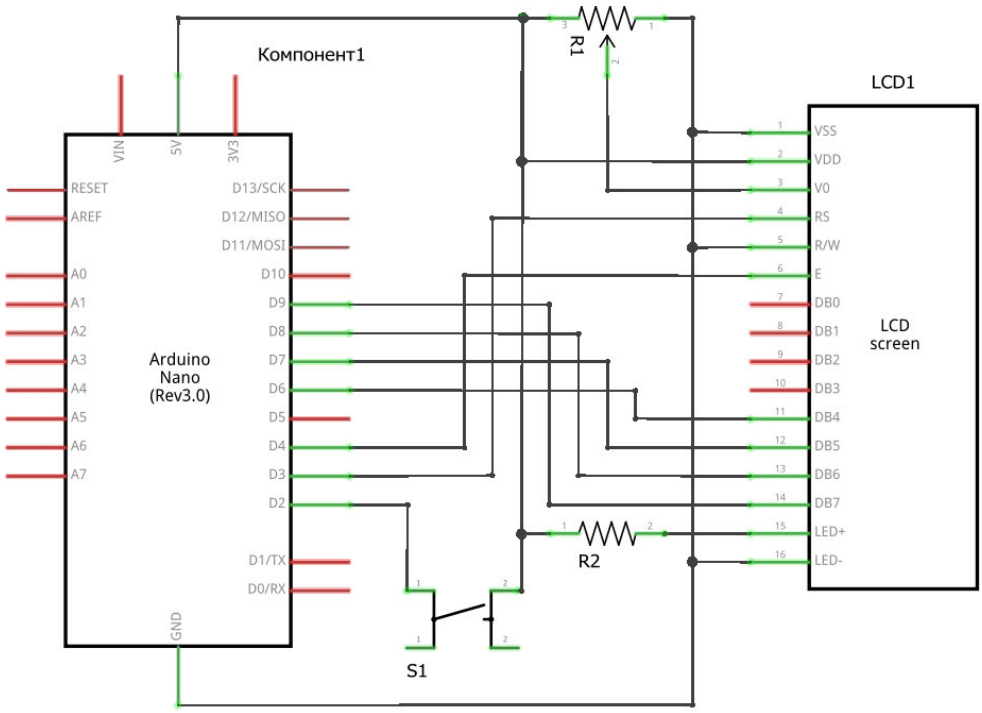


Рисунок 2.1 - Принципова схема з’єднання тестового макету.

Підключення дисплея є стандартне, як і для всієї лінійки подібних дисплеїв на цьому контролері (Рис. 12). Контраст екрану налаштовується змінним резистором (V0- pin 3). Живлення – 5VDC.

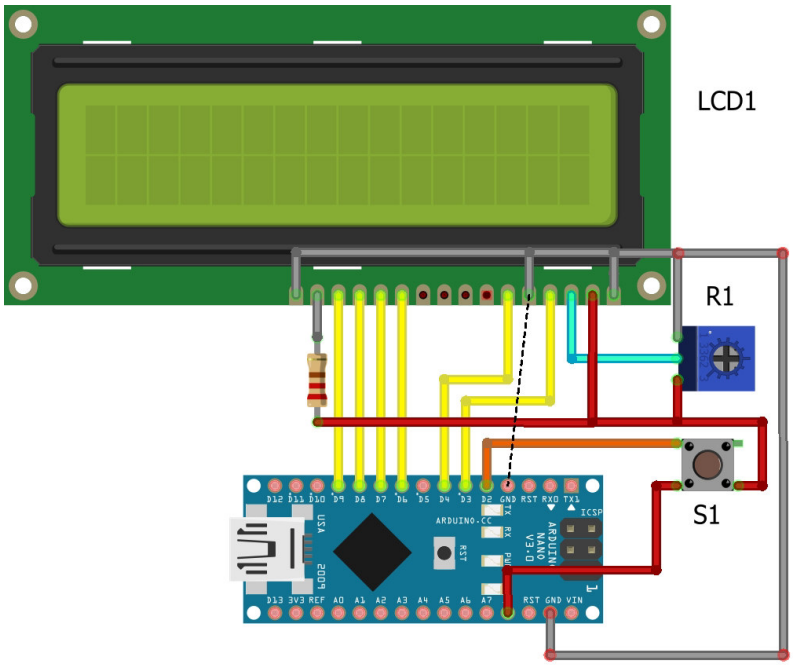


Рисунок 2.2 - Типова схема з’єднання елементів макету.

Таблиця 2.1. Приклад під’єднання дисплея (1602) з платформою ArduinoNano.

Arduino Nano	LCD 1602		Arduino Nano	LCD 1602	
Gnd	1	Vss (Gnd)	– n.c.	9	DB2 – n.c.
+5V	2	Vdd (+5V)	– n.c.	10	DB3 – n.c.
R1 pot.	3	Contrast Adj	D6	11	DB4
D3	4	RS	D7	12	DB5
Gnd	5	R/W	D8	13	DB6
D4	6	Enable	D9	14	DB7
– n.c.	7	DB0– n.c.	+5V через R2	15	back light LED +
– n.c.	8	DB1– n.c.	Gnd	16	back light LED-

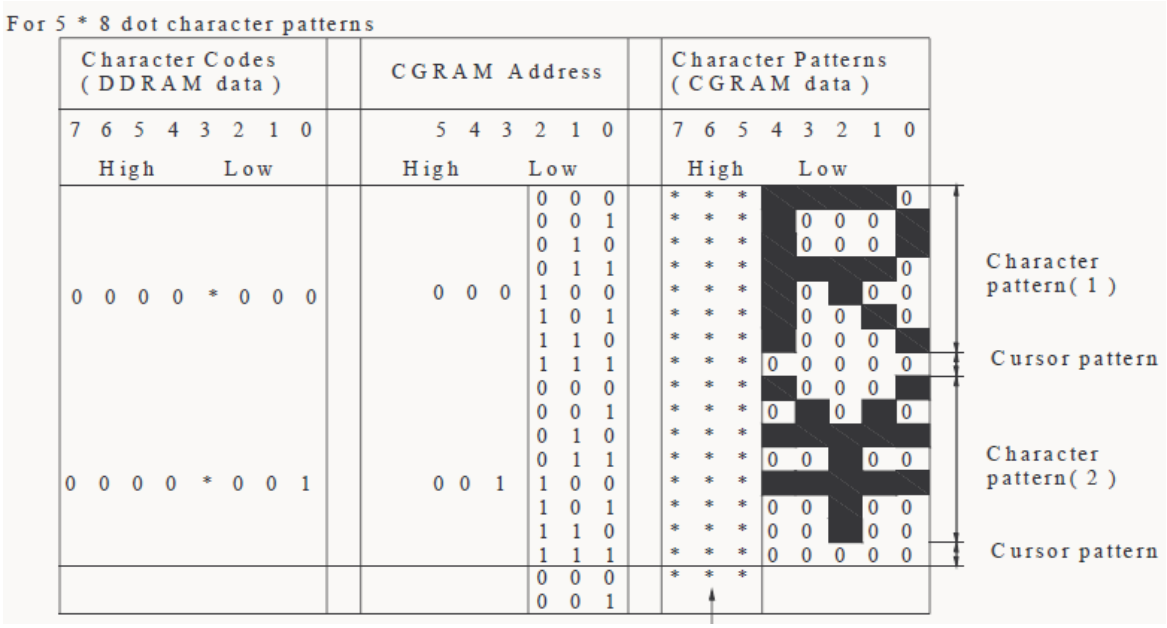


Рисунок 2.3 - Приклад організаціїсимвола узнакомісті (5x7),символа- R

Бібліотеки програмного середовища Arduino IDE для роботи з LCD – дисплеєм

Для взаємодії платформи Arduino з LCD (1602),необхідно використати бібліотекупрограми Arduino IDE:

#include<LiquidCrystal.h> - бібліотека включає в себе велику різноманітність функцій для управління дисплеєм із паралельним інтерфейсом (внутрішній драйвер LCD - Hitachi HD44780). Для більш детального опису див. HelpArduinoIDE.

Використання наявних бібліотек робить скетч (програму) простішоюта коротшою, полегшує сприйняття та зручно для аналізу існуючих програм. Після підключення до скетчу всіх необхідних бібліотек треба створити програмний об'єкт LCD-дісплей та використовувати його функції.

Опис основних функцій бібліотеки LiquidCrystal.h

LiquidCrystal() - ініціалізія бібліотеки, прив’язуючи необхідний контакт інтерфейса LCD із номером вивода платформи Arduino. Для прикладу, використаємо найпростіший варіант підключення дисплея.

Синтаксис:

constintrs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal(rs, enable, d4, d5, d6, d7);

begin() - Ініціалізація інтерфейса до LCD дисплея, та вказує розміри (ширину та висоту) дисплея. Використання даної функції повинно передувати усім іншим із LCD-бібліотеки.

Синтаксис:

lcd.begin(16, 2); //розмір дисплея 16- символів, 2 - строки

clear()-очищення LCD дисплея і переміщення курсора у верхній лівий кут (адреса- 00).

Синтаксис:

lcd.clear();

home()–Повертає курсор у ліву верхню частину LCD дисплею. Використовується для виведення наступного тексту на дисплей (декілька повідомлень). Для очищення дисплея, скористайтеся функцією clear ().

Синтаксис:

lcd.home();

setCursor() – Розміщення LCD курсора. Встановлюється місце на дисплеї, в якому буде відображатися наступний текст.

Синтаксис:

lcd.setCursor(0, 1); //встановити курсор у положення (стовпчик 0, рядок 1).

lcd.write() - запис символа в LCD дисплей.

print() – вивести текст на LCD дисплей.

Синтаксис:

lcd.print(data);

cursor()-відображає LCD-курсор: підкреслення (рядка) у тому місці, до якого буде записаний наступний символ.

blink()– Відображає мигання курсора. Якщо використовується в поєднанні з функцією cursor (), результат буде залежати від конкретного відображення.

display() - ВмиканняLCD дисплея після його вимкнення за допомогою командиnoDisplay (). Це відновить текст (та курсор), який був на відображений на дисплеї.

scrollDisplayLeft() – Зміщає вміст дисплея (текст та курсор) на одинезнакомісто у ліво.

autoscroll() - Увімкнення автоматичної прокрутки LCD-дисплея. Функція імітує ефект біжучого рядка із зміщенням з ліва на право.

leftToRight() - Встановіть напрямок відображення тексту, записаного на LCDдисплей, з ліва на право, за замовчуванням. Це означає, що наступні символи, виведені на дисплей, будуть зміщатися зліва направо, але не впливають на попередньо виведений текст (не накладаються).

createChar() – Створення спец. символа для відображення на LCD-дисплеї. Підтримується до восьми символів розміром 5x8 пікселів (від 0 до 7),використовуючи масив бітових масок bitmap для створення чорних і білих пікселів.

Зовнішній вигляд кожного спец. символу визначається масивом з восьми байтів (бітова карта), по одному для кожного рядка. П'ять найменш значущих бітів кожного байта

визначають пікселі в цьому рядку. Щоб відобразити на екрані спеціальний символ, необхідно записати функцією - write() його номер.

lcd.createChar(num, data)
num- номер (0-7) спец.символа
data- дані спец символа.

Прикладініціалізації бібліотеки та створення спеціального символа в одезнакомісто:

```
#include<LiquidCrystal.h>
LiquidCrystallcd(12, 11, 5, 4, 3, 2);
bytesmiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};
voidsetup() {
lcd.createChar(0, smiley);
lcd.begin(16, 2);
lcd.write(byte(0));
}
voidloop() {}
```

Приклад тестової програми використаний із посилання :
http://www.arduino.cc/en/Tutorial/LiquidCrystalHelloWorld

Приклад тестової програми

```
#include<LiquidCrystal.h>// підключаємо бібліотеку
```

```
// ініціалізація бібліотеки, та створення асоціативного з’єднанняLCDдисплея та виводів модуля ArduinoNano.
constintrs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystallcd(rs, en, d4, d5, d6, d7);

voidsetup() {
 // налаштування типу LCDдисплея (число стовпців, число рядків):
lcd.begin(16, 2);
 // вивести повідомлення на LCD.
lcd.print("hello, world!");
}

voidloop() {
 // встановлюємо курсор у положення  - стовбець 0, рядок 1
lcd.setCursor(0, 1);
 // вивести число секунд після скидання:
lcd.print(millis() / 1000);
}
```

Відладка конструкції, та варіанти усунення несправності макета

Якщо після завантаження скетчу не з'явилося жодної напису на дисплеї, необхідно виконати наступні дії:

- збільшити або зменшити контрастність дисплея, регулюючи напругу-V0.
- перевірити чи правильно підключено контакти, чи підключено живлення дисплея та підсвітки (стосується реального об’єкту).
- Створення власних символів.

Варто зазначити, що літери англійського алфавіту зашиті (запрограмовані) в пам'ять контролера всередині дисплея і з

ними проблем немає. Проте літери кирилиці модуль екрана не підтримує, тому необхідно їх створити самому. Дисплей підтримує до 8 користувацьких символів розміром 5x8 пікселів. Зовнішній вигляд кожного користувацького символу задається масивом з восьми байт, кожен з яких характеризує відповідний рядок.

- у випадку відсутності ознак роботи демонстраційної програми на LCD дисплеї використати тестову програму блимання вбудованого світло діода на платформі Arduino Nano, D13.

Таким чином можна протестувати правильність завантаження скетча, та працездатність самого модуля Arduino.

Приклад тестової програми використаний із посилання:
http://www.arduino.cc/en/Tutorial/Blink

```
//
thesetupfunctionrunsoncewhenyoupressresetorpowertheboard
voidsetup() {
  // initializedigitalpin LED_BUILTIN asanoutput.
pinMode(LED_BUILTIN, OUTPUT);
}

// theloopfunctionrunsoverandoveragainforever
voidloop() {
digitalWrite(LED_BUILTIN, HIGH);   // turnthe LED on (HIGH isthevoltagelevel)
delay(1000);                       // waitfor a second
digitalWrite(LED_BUILTIN, LOW);    // turnthe LED offbymakingthevoltage LOW
delay(1000);                       // waitfor a second
```

}

1. Завдання до лабораторної роботи етап макетування

2. Створити тестову програму роботи платформи Arduino, та LCDдисплея використовуючи приведену вище інформацію. Середовище для програмування – ArduinoIDE.

3. Результатом виконання лабораторної роботи є файл:

- Бінарний файл створений у ArduinoIDE – Test_LCD.hex

4. Перелік індивідуальних завдань:

- створити симуляцію на дисплеї (1602), верхній рядок, повідомлення «TestAduino»;
- створити симуляцію на дисплеї (1602), почергова поява у верхньому та нижньому рядках, повідомлення «TestAduino»;
- створити симуляцію на дисплеї (1602), верхній рядок, повідомлення «TestAduino» із ефектом біжуча стрічка;
- створити симуляцію на дисплеї (1602), верхній рядок, створення спец символана одне знакомісто  ,  ,  ,  .
- створити симуляцію на дисплеї (1602), верхній рядок, «довільне тестове повідомлення», нижній рядок – секундомір із розрядністю одиниці секунд.

Зміст звіту

1. Тема та мета роботи.

2. Текст програми із коментарямистворений у ArduinoIDE.

3. Результатом виконання лабораторної роботи є файли із розширенням (*.hex, *.pdsprj)

4. Висновки з роботи надіслати на E-mail.

Лабораторна робота № 3
Побудова та дослідження МП систем на базі платформи Arduino Nano із застосуванням цифрових датчиків: HC-SR04, DS18B20

Мета роботи: на основі лабораторного набору на платформі Arduino освоїти основні типи датчиків наявні у лабораторному наборі (цифрові та аналогові), схеми їх підключення до МК ATmega328. Створити графічно алгоритм роботи одного із датчиків з використанням цифрових входів (аналогових входів) та їх конфігурування, намалювати принципову електричну схему пристрою, ознайомитись із теоретичними відомостями датчиків, та способами організації їх опитування, написати тестову програму у середовищі Arduino IDE із використанням наявних цифрових входів на (DCC duino Nano, МК - ATmega328). Навчитись використовувати різновиди функцій та команд мови С++ для створення тестової програми використовуючи освоєні у попередніх роботах алгоритми програмування.

Для складання схеми макету знадобляться наступні деталі, які є в кожному з лабораторних наборів Arduino:

7. Платформа Arduino Nano;
8. Кабель USB для програмування;
9. Макетна плата для прототипування;
10. З’єднувальні провідники;
11. Резистори 1кОм – 5 шт;
12. Ультразвуковий датчик відстані HC-SR04 та інтерфейс TTL PWL (модуль);
13. Датчик температури Dallas DS18B20 та інтерфейс 1-Wire;
14. Датчик температури та вологостіи DHT11

І. Датчик відстані ультразвуковий HC-SR04

Рисунок 3.1. - Зовнішній вигляд датчика HC-SR04

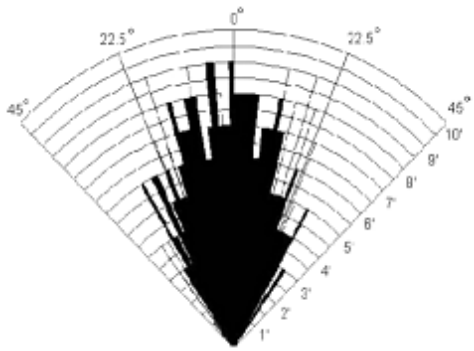


Рисунок 3.2. - Діаграма направленості датчика HC-SR04

Ультразвуковий датчик відстані HC-SR04 Ultrasonic Sensor працює за принципом ультразвукової ехолокації. Відстань визначається за часом, що пройшов між відправкою сигналу та отриманням відлуння.

Представимо приклад типового варіанту підключення ультразвукового датчика відстані HC-SR04 із платформою Arduino Nano:

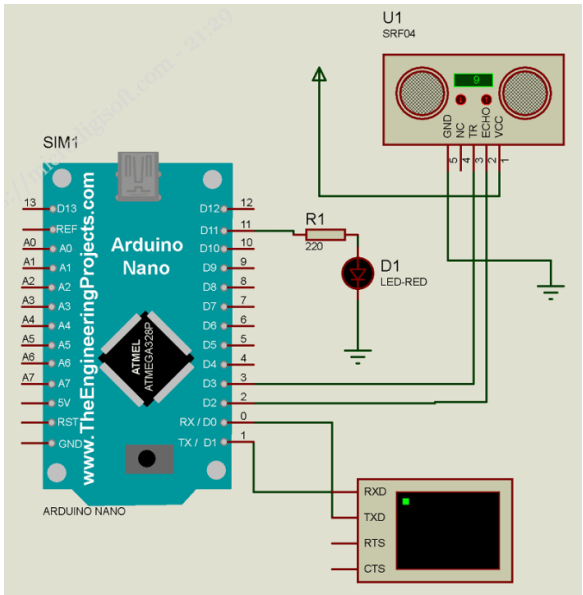


Рисунок 3.3. - Типова схема під’єднання УЗ датчика HC-SR04 до Arduino Nano.

Характеристики УЗ датчика HC-SR04:

Напруга живлення	5V(DC).
Статичний струм	до 2mA.
Вихідний рівеь TTL сигналу	(High) - 5 В, (Low) - 0В
В.Ефективний кут	$<15^0$
Детектуєма відстань	2cm-450cm
Роздільна здатність	Up to 0.3cm
Мінімальна тривалість	10us TTL імпульс
Сигнал відклику	TTL PWL сигнал

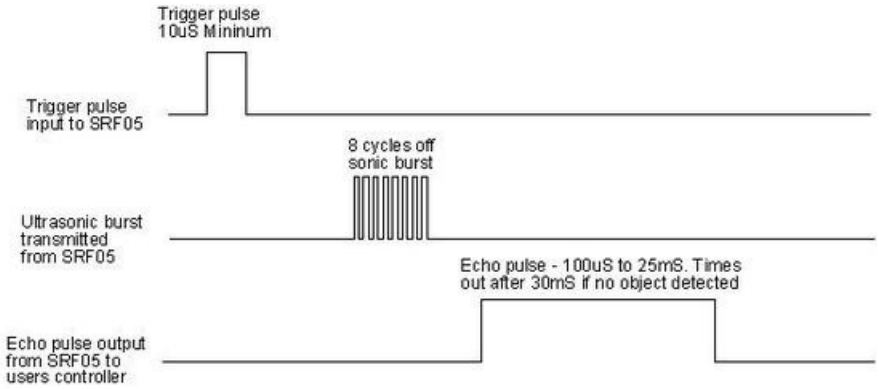


Рисунок 3.4. - Часова діаграма для пояснення принципу роботи модуля HC-SR04.

Принцип роботи модуля, та приклад програмної реалізації.

Послідовність роботи з датчиком відстані складається з декількох етапів:

1) щоб ініціалізувати надсилання сигналу датчиком відстані - дальноміром, необхідно подати сигналу (High) тривалістю 10 µs на пін Trig.

2) після встановлення високого рівня тривалістю 10 µs на пін Trig модуль генерує пачку з восьми імпульсів частотою 40 кГц і встановлює високий рівень на піні Echo.

3) після отримання відбитого сигналу модуль встановлює низький рівень на піні Echo. Знаючи тривалість сигналу на піні Echo, можемо обчислити відстань, помноживши час,

проходження звуковим імпульсом, перш ніж повернувся до модуля, із швидкістю поширення звуку в повітрі (340 м/с).

4) функція pulseIn дозволяє визначити тривалість імпульсу в µs.

5) запишемо результат роботи цієї функції у змінну duration.

6) тепер обчислимо відстань перевівши швидкість із (м/с) у (см/мкс):

Знаючи тривалість рівня сигналу (High) на виводі Echo, можемо вирахувати відстань, помноживши час проходження звукового імпульсу, який повернувся до модуля, на швидкість поширення звуку в повітрі (340 м/с).

Функція pulseIn дозволяє визначити тривалість імпульсу в µs. Запишемо результат роботи цієї функції у змінну duration.

Тепер обчислимо відстань перевівши одиниці вимірювання швидкості з (м/с) у (см/мкс):

distance = duration * 340 м/с = duration * 0.034 м/мкс

Перетворимо десятковий дріб у звичайний

distance = duration * 1/29 = duration / 29

Враховуючи те, що звук подолав відстань до об'єкта і назад, поділимо отриманий результат на 2

distance = duration / 58

Оформимо в код все вищесказане і виведемо результат у Serial Monitor ultrasonic.ino

```
// Налаштовуємо піни МК до яких підключено модуль
int trigPin = 10;
int echoPin = 11;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
```

```
  pinMode(echoPin, INPUT);
}
 void loop() {
  int duration, distance;
   digitalWrite(trigPin, LOW); // для більшої точності встановимо значення LOW на піні Trig
  delayMicroseconds(2);
    digitalWrite(trigPin, HIGH); // Тепер встановимо рівень High на піні Trig
    delayMicroseconds(10); // Затримка 10 µs
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);  // Визначаємо тривалість High рівня на піні Echo
    distance = duration / 58; // Розрахуємо відстань
    Serial.print(distance); // Виведемо значення у Serial Monitor
  Serial.println(" cm");
  delay(100);
}
```

Індивідуальне завдання

Для кожного досліджуваного макету УЗ дальноміра HC-SR04 написати скетч з різними варіантами спрацювання на перешкоду та відображення даних виміряних величин у монітор порта. Такий варіант відображення даних полегшить процес від лагодження конструкції.

Варіанти виконання завдань:

1. Створити тестову програму з можливістю вимірювання відстані та відображення даних у (міліметрах) у монітор порта.

2. Створити тестову програму з функцією сигналізації пересічення граничних меж (100, 200, 300) мм та відображення даних у (міліметрах) у монітор порта.

3. Створити тестову програму для дослідження точності вимірювання відстані та визначення граничних відстаней УЗ дальноміра.

4. Створити макет та тестову програму застосувавши для відображення відстані LCD дисплей (16x2) та відображення відстані УЗ дальноміра у (сантиметрах).

II. Датчик температури Dallas DS18B20 та інтерфейс 1-Wire

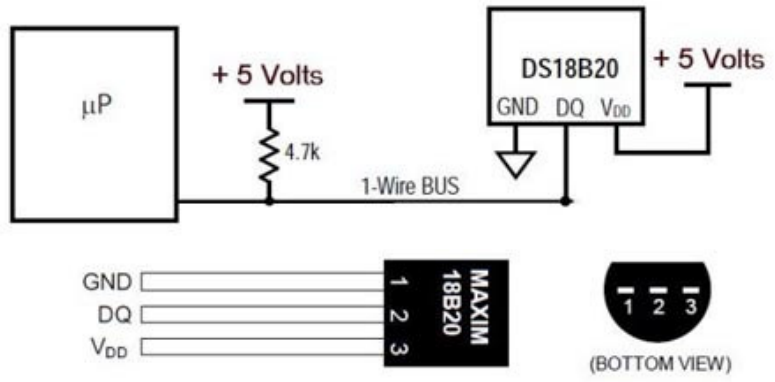


Рисунок 3.5. - Типова схема включення цифрового датчика температури DS18B20 та МК

Діапазон вимірюваних температур становить від -55° С до +125° С. Цифрове значення температури, що зчитується з приладу, є прямим безпосереднім кодом виміряного значення температури і не потребує додаткових перетворень. Програмована користувачем роздільна здатність вбудованого АЦП може бути змінена в діапазоні від 9 до 12 розрядів вихідного коду. Абсолютна похибка перетворення менше 0,5 ° С в діапазоні контрольованих температур -10 ° С до +85 ° С. Максимальний час повного 12-розрядного перетворення ~ 750мс (при роздільній здатності 12 розрядів). Для підключення потрібен підтягуючий резистор 4.7кОм (див. Рис. 2.5).

Внутрішня енергонезалежна пам'ять температурних налаштувань забезпечує запис значень верхньої та нижньої межі температур. Крім того, мікросхема містить вбудований логічний механізм пріоритетної сигналізації виходу температури за один з обраних порогів. Цифрова схема 1-Wire-інтерфейсу приладу організований таким чином, що існує теоретична можливість адресації необмеженої кількості таких пристроїв на однопровідній лінії вказаного інтерфейсу.

Термометр оснащено пам’яттю з індивідуальний 64-розрядний унікальним номером (груповий код 028H). Також внутрішнє схемне рішення дає можливість роботи без зовнішнього джерела живлення лише за рахунок застосування схеми паразитного живлення однопровідною лінією. Живлення пристрою через окремий вхід живлення здійснюється напругою від 3.0В до 5.5В.

Режим вимірювання температури.

Основна функція датчика DS18B20 – перетворення температури у цифровий код. Роздільна здатність перетворення задається у діапазоні 9, 10, 11 або 12 біт. Це відповідає роздільній здатності по температурі - 0,5 (1/2) °C, 0,25 (1/4) °C, 0,125 (1/8) °C та 0,0625 (1/16) °C. При включенні живлення стан регістру конфігурації DS18B20 встановлено на роздільну здатність - 12 біт.

Характеристики датчика температури DS18B20:

Напруга живлення	3.0..5.5 В
Діапазон вимірюваних	-
Точність показань	0.5 °C
Крок вимірювання	0.0625 °C
Інтерфейс	1-Wire
Споживаний струм	1 мА

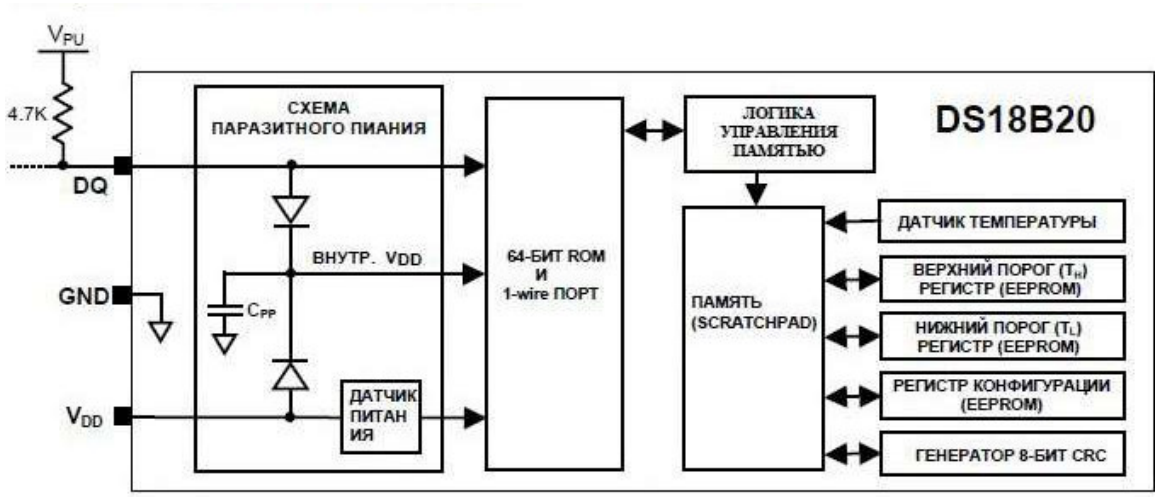


Рисунок 3.6. - Блок схема термометра DS18B20

Після включення живлення DS18B20 перебуває у стані сну (з низьким енергоспоживанням). Щоб ініціювати вимірювання температури, пристрій (Master) – МК (мікроконтролер) повинен виконати команду Перетворення Температури код- [44h]. Після завершення перетворення результат вимірювання температури буде зберігатись у 2 байтовому регістрі температури, і датчик знову перейде в стан сну.

Якщо DS18B20 увімкнено за схемою із зовнішнім живленням, то пристрій (Master) – МК може контролювати стан команди конвертації. Для цього він повинен зчитувати стан лінії -DQ (виконувати тимчасовий слот читання), по завершенні команди лінія перейде у високий стан - High. Під час виконання команди конвертації лінія утримується у низькому стані -Low.

Для варіанта живлення датчика від паразитної ємності такий спосіб не допустимий, оскільки під час операції перетворення на шині необхідно утримувати високий рівень сигналу живлення датчика

Датчик DS18B20 вимірює температуру у градусах за шкалою Цельсія. Результат виміру представляється як 16-розрядне, число у додатковому коді (рис. 2.7).

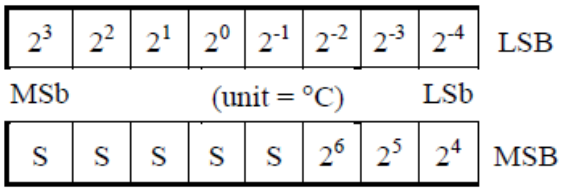


Рисунок 3.7. - Представлення результату вимірювання температури DS18B20

Біт знака (S) дорівнює 0 для позитивних значень та 1 для від'ємних значень. При роздільній здатності 12 біт, у регістра температури всі біти значущі, тобто. мають достовірні значення. Для роздільної здатності 11 біт, не визначений біт 0. Для роздільної здатності 10-бітного не визначені біти 0, 1. При роздільній здатності 9 біт, не достовірне значення мають біти 0, 1 і 2.

У таблиці 2.2 показані приклади відповідності цифрових кодів значенню температури.

Таблиця 3.2

Температура	Цифрове представлення (Binary)	Цифрове представлення (Hex)
+125^{0}C	0000 0111 1101	07D0
85^{0}C	0000 0101 0101	0550
+25.0625^{0}C	0000 0001 1001	0191
0^{0}C	0000 0000 0000	0000
-25.0625^{0}C	1111 1110 0110	FF6F
-55^{0}C	1111 1100 1001	FC90

Програмна реалізації опитування датчика DS18B20

Алгоритм отримання інформації про температуру складається з наступних етапів:

• Визначення адреса датчика, перевірка його підключення.

• На датчик подається команда прочитати температуру та перемістити виміряне значення у регістр. Процедура відбувається довше, час виконання 750 мс.

• Подається команда на читання інформації з регістру та відправлення отриманого значення в «монітор порту» - прогами Arduini IDE,

• За необхідності температура конвертується в градуси Цельсія/Фаренгейта.

Приклад тестового скетча для DS18B20

Заготовка тестової програми для роботи з цифровим датчиком виглядає наступним чином. Використовуємо стандартну бібліотеку OneWire – Arduino IDE.

```
#include <OneWire.h>
/*
* Опис взаємодії з цифровим датчиком ds18b20
* Підключення ds18b20 до 8 піна Arduino
*/
OneWire ds(8); // Створюємо об'єкт OneWire для шини 1-Wire, за допомогою якого буде реалізована робота з датчиком
void setup(){
  Serial.begin(9600);
}

void loop(){
  // Визначаємо температуру від датчика DS18b20
  byte data[2]; // Місце для значення температури
```

```
  ds.reset(); // Початок взаємодії зі Скиданням усіх попередніх команд і параметрів
  ds.write(0xCC); // Записуємо команду датчика DS18b20 запустити пошук за адресою. В нашому випадку тільки один пристрій
  ds.write(0x44); // Команда датчику DS18b20 виміряти температуру. Само значення температури ми ще не отримуємо - датчик перемістить значення у внутрішню пам'ять

  delay(1000); // Термодатчик вимірює температуру

  ds.reset(); // Підготовка перед одержанням значення виміряної температури
  ds.write(0xCC);
  ds.write(0xBE); // Запит на передачу значення реєстрів зі значенням температури
// Получаем и считываем ответ
  data[0] = ds.read();// Читаємо молодший байт значення температури
  data[1] = ds.read(); // Читаємо старший байт значення температури
  // Одержуємо загальне значення:
  //    - спочатку з’єднуємо значення (молодший із старшим байтом) значення,
   // - потім моножимо його на коефіцієнт, що відповідає роздільній здатності (для 12 бит по замовчуванню - 0,0625)
  float temperature =  ((data[1] << 8) | data[0]) * 0.0625;

  // Відображаємо отримане значення температури в моніторному порту – Arduino IDE
  Serial.println(temperature);
```

```
}
```

Версія програми опитування термодатчика запропонована ШІ(Chat GPT)

```
#include <OneWire.h> // бібліотека OneWire для зчитування даних з датчика
#include <DallasTemperature.h> // бібліотека DallasTemperature для роботи з датчиком температури

#define ONE_WIRE_BUS 2 // пін для зчитування даних з датчика
OneWire oneWire(ONE_WIRE_BUS); // ініціалізація інтерфейсу зчитування даних з датчика
DallasTemperature sensors(&oneWire); // ініціалізація датчика температури

void setup() {
  Serial.begin(9600); // ініціалізація монітора послідовного порту
  sensors.begin(); // ініціалізація датчика температури
}

void loop() {
  sensors.requestTemperatures(); // запит температури з датчика
  float temperatureC = sensors.getTempCByIndex(0); // зчитування температури в градусах Цельсія
  Serial.print("Temperature: "); // виведення тексту на монітор послідовного порту
```

```
  Serial.print(temperatureC); // виведення температури на монітор послідовного порту
  Serial.println("C"); // виведення одиниці виміру на монітор послідовного порту
  delay(1000); // затримка 1 секунда
}
```

Індивідуальне завдання

Використовуючи рекомендовану виробником схему включення та тестову програму для DS18B20 реалізувати відображення даних виміряних величин температури у монітор порта (або ж застосувати зовнішній LCD дисплей).

Варіанти виконання завдань:

1. Реалізувати варіант вимірювання температури з різною роздільною здатністю (9,10,11,12)біт.
2. Реалізувати на основі термодатчика DS18B20 терморегулятор з релейним режимом управління (On/Off).
3. Створити та відлагодити тестову прогаму для опитування по (1-Wire) мережі декількох термодатичків.
4. Створити тестовий макет (DS18B20 - Arduino) для реалізації простого терморегулятора із заданим гістерезисом, та керуванням виконавчим елементом.

Лабораторна робота №4
Побудова та дослідження МП систем на базі платформи Arduino Nano із застосуванням цифрових датчиків: DHT11, MQ-2

Мета роботи: на основі лабораторного набору на платформі Arduino освоїти основні типи датчиків наявні у лабораторному наборі (цифрові та аналогові), схеми їх підключення до МК ATmega328. Створити графічно алгоритм

роботи одного із датчиків з використанням цифрових входів (аналогових входів) та їх конфігурування, намалювати принципову електричну схему пристрою, ознайомитись із теоретичними відомостями датчиків, та способами організації їх опитування, написати тестову програму у середовищі Arduino IDE із використанням наявних цифрових та аналогових входів на (МК - ATmega328). Навчитись використовувати різновиди функцій та команд мови С++ для створення тестової програми використовуючи освоєні у попередніх роботах алгоритми програмування із використанням аналогових сигналів.

Для складання схеми макету знадобляться наступні деталі, які є в кожному з лабораторних наборів Arduino:

15. Платформа Arduino Nano;
16. Кабель USB для програмування;
17. Макетна плата для прототипування;
18. З’єднувальні провідники;
19. Резистори 1кОм – 5 шт;
20. Датчик температури та вологостіи DHT11
21. Датчик витоку газу MQ-2;

III. Датчик температури та вологості DHT11

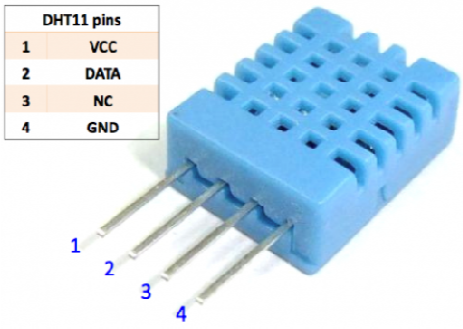


Рисунок 4.1. - Нумерація та призначення виводів DHT11

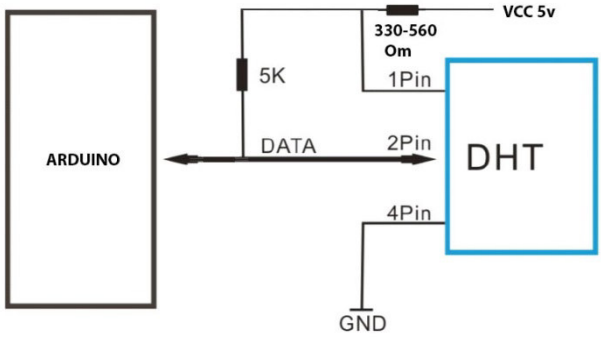


Рисунок 4.2. - Типова схема застосування

Датчик DHT11 призначений для вимірювання температури та вологості повітря. Передача даних здійснюється одним проводом з використанням власного протоколу. Може бути використаний у пристроях Arduino, AVR, PIC, ARM та ін. Для роботи з Arduino існує готова бібліотека..

Характеристики DHT11:

Модель датчика:	DHT11
Діапазон вимірювання вологості/ точність	5 - 95% RH ± 5% (макс.)
Діапазон вимірювання температури/точність	0 ~ +60 ºC ± 2% (макс.)
Напруга живлення	3.5-5.5 В
Частота опитування сенсора	не более 1 Гц
Розміри корпуса	15.5 x 12 x 5.5 мм

Інструкції послідовного зв'язку (однопровідна двонаправлена шина)

Датчик DHT11 використовує спрощений зв'язок по однопровідній (1-Wire) шині. Однопровідна шина, використовується як для передачі даних, так і для передачі керуючих сигналів.

Головний або ведений (Master- Slave) пристрій через присвоєний порт контролера із трьома станами, підключений до лінії передачі даних, так аби пристрій не надсилав дані звільняючи шину для активних пристроїв (тобто з якими у даний момент відбувається обмін даними), поки інші пристрої використовують шину. Для цього потрібен зовнішній підтягуючий резистор (5,1 кОм), так коли шина простоює шина, її статус високий рівень.

Формат передачі даних 8 біт цілі одиниці вологість + 8 біт десяті одиниці вологість + 8 біт цілі одиниці температура + 8 біт

Десяті одиниці температури, 8-бітна контрольна сума.

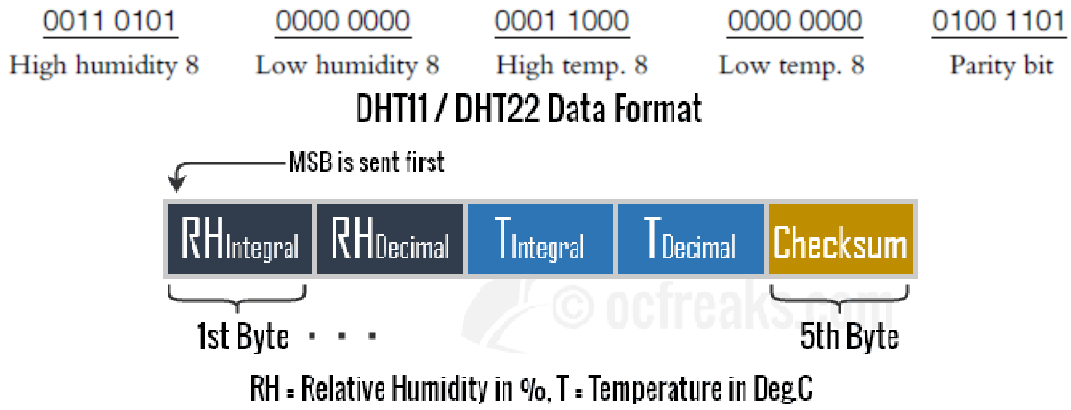


Рисунок 4.3. - Формат пакета передачі даних.

Формування даних вологості та температури.
Контрольна сума: 0011 0101+0000 0000+0001 1000+0000 0000=0100 1101
Одержані дані.
Вологість: 0011 0101=35h=53%RH
Температура: 0001 1000=18h=24^{0}C

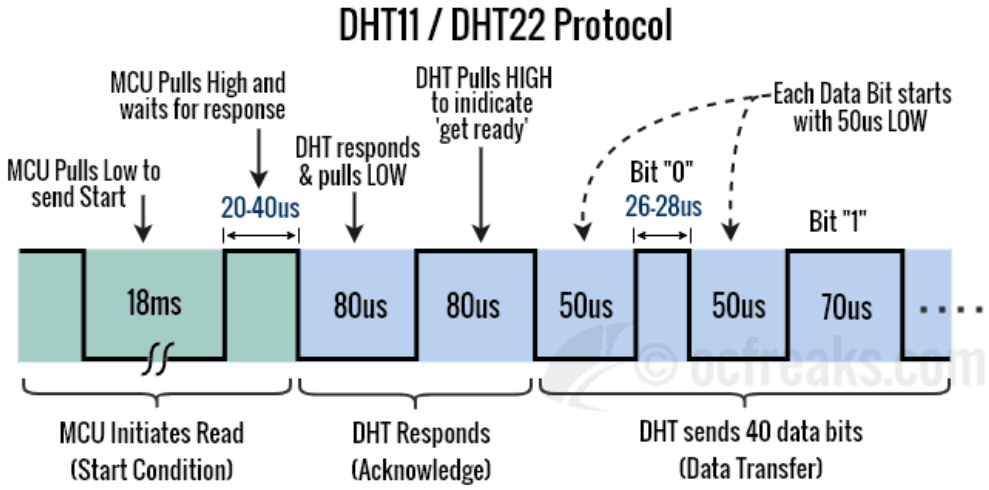


Рисунок 4.4. - Часова діаграма пакета передачі даних DH11

Приклад програми:

```
#include "DHT.h"
#define DHTPIN 2     // контакт МК для передачі даних
```

```
// вибір використовуваного типа датчика
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// ініціалізація датчика
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Зчитування температури та вологості триває 250 мілісекунд
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // перевіряємо достовірність одержаних даних
  if (isnan(t) || isnan(h)) {
    Serial.println("Error reading from DHT");
  } else {
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %t");
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" *C");
  }
}
```

IV. Датчик витоку газу MQ – 2

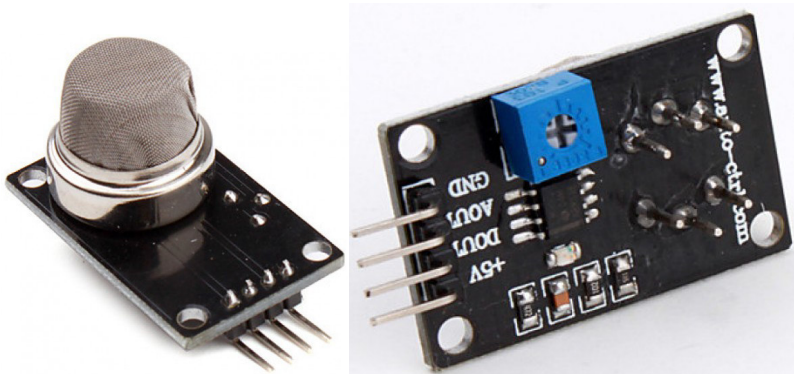


Рисунок 4.5. – Зовнішній вигляд макетної плати MQ – 2

Виходи на платі	MQ - 2
Vcc	живлення: +5 В
DO	цифровий вихід: TTL цифровий 0 і 1 (0,1 В і 5 В)
AT	аналоговий вихід: 0.1
0.3V	низька концентрація
4V	висока концентрація
GND	земля

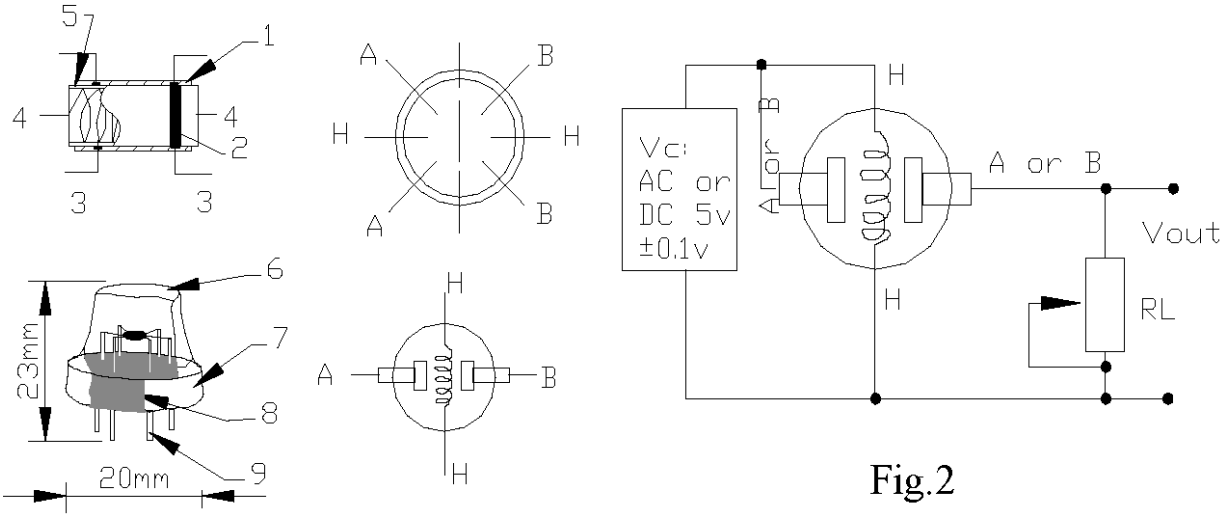


Fig. 1

Fig.2

Рисунок 4.6. - Типова схема застосування

Витік газу - це украй небезпечна ситуація, особливо в житлових будинках або на виробництві. Цей датчик використовується для своєчасного виявлення і запобігання таким випадкам в будинках, промислових і комерційних будівлях, охоронно-пожежних системах, в проектах автоматизації, для сигналізаторів витоку газів, газових детекторів, автоматичного вентиляційного устаткування.

Датчик газу MQ - 2 дозволяє виявляти наявність в навколишньому повітрі вуглеводневих газів (пропан, метан, Н-Бутан), диму (зважені частки, горіння, що є результатом), водню. Датчик можна використовувати для виявлення витоків промислового газу і задимлення.

Цей датчик реагує на появи домішок газу в повітрі. Він зроблений з високоякісного оксиду металу, тому при появі вищеперелічених газів або диму, провідність датчика росте із зростанням концентрації цього газу.

Він має високу чутливість і малий час відгуку. Чутливість може бути налагоджена за допомогою потенціометра на платі датчика.

Технічні характеристики

Модель датчика:	MQ - 2
Споживаний струм:	150 мА
Інтерфейс:	аналоговий, цифровий
Напруга живлення:	+5 В
Розміри корпуса	32 х 20х 22 мм

Сфера застосування:

- витоки газу для будинків;
- різні проекти автоматизації;
- у промислових і комерційних будівлях;
- у охоронно-пожежних системах;
- для сигналізаторів витоку газу;

- для портативних газових детекторів;
- для автоматичного вентиляційного устаткування.

Індивідуальне завдання

Використовуючи набуті навички із попередніх робіт (створення інтерфейсу для відображення виміряних даних із сенсорів) за допомогою монітор порта (Arduino IDE) (або ж застосувати зовнішній LCD дисплей).

Варіанти виконання завдань:

1. Реалізувати варіант тестової програми вимірювання температури, та відображення даних на обраному інтерфейсі.
2. Реалізувати варіант тестової програми вимірювання вологості, та відображення даних на обраному інтерфейсі.
3. Створити та відлагодити тестову прогаму для спрацювання сигналізації по аварійних рівнях вологості та температури, із можливістю зміни гістерезису спрацювання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ ТА РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Мікропроцесорні та мікроконтролерні системи : підручник. У 2 ч. Ч. 1. Мікропроце-сорні системи [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 16,7 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2020. – 361с.

2. Новацький А. О. Комп'ютерна електроніка-3. Мікропроцесорні системи. Апаратні за-соби мікропроцесорних систем : навч. посіб. / А. О. Новацький. – Київ : НТУУ «КПІ», 2015. – 470 с.: ил

3. Проектування мікропроцесорних систем: Проектування мікропроцесорних систем на базі мікроконтролерів сімейства MCS-51: Периферійні модулі мікроконтролерів сі-мейства MCS-51: навч. посіб. для студ. напряму підготовки 6.050201 «Системна ін-женерія» кафедри Автоматики та управління у технічних системах / А. О. Новацький. – Київ : НТУУ «КПІ», 2016. – 399 с.: ил.

5. http://www.atmel.com/documents/ATmega8.pdf

6. https://www.arduino.cc/en/uploads/Main/ArduinoNano Manual23.pdf

7. https://www.fecegypt.com/uploads/dataSheet/1522503 120 _ard uino%20nano.pdf

8. Фурман И.А., Краснобаев В.А., Скороделов В.В., Рысованый А.Н. Организация и про-граммированиемикроконтроллеров: Учебник. – Харьков: Эспада, 2005. – 248 с.

9. Головінський Б.Л., Лементарьов В.В., Руденський А.А. Мікропроцесорна техніка. На-вчальний посібник. Ніжин, 2007. – 120 с.

10. Гаєвський О.Ю. Цифрові автомати в автоматизації виробництв. Навч. Посіб. – К.: А.С.К., 2007. – 512 с.

11. Болюх В.Ф., Данько В.Г. Основи електроніки та мікропроцесорної техніки: навчаль-ний посібник. Харків: НТУ «ХПІ», 2011 257 с.

12. Мікропроцесорні та мікроконтролерні системи: лаб. практикум [Електронний ресурс]: навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спец. 126 «Інформаційні системи та технології» / Уклад. А. О. Новацький. – Електронні текстові дані (1 файл: 18,97 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 415 с.

13. Arduino. [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/

14. AtmelStudio. [Електронний ресурс]. MicrochipAtmel. – Режим доступу: http://www.atmel.com/tools/atmelstudio.aspx

15. Шегедин, О.І. Теоретичні основи електротехніки : навч.посіб. / О.І. Шегедин, В.С. Маляр.; Львів : Магнолія 2006, 2012. – 167с.

16. Коруд, В.І. Електротехніка : підручник для ВНЗ / В.І.Коруд, О.Є. Гамола.; за заг. ред. В.І. Коруда.- 3-тє вид., перероб. і доп.- Львів:Магнолія, 2007.- 447с.

17. Мілих, В.І. Електротехніка,електроніка та мікропроцесорна техніка:підручник для ВНЗ/В.І. Мілих, О.О.Шавьолкін;за ред.В.І.Мілих.-К.: Каравела, 2007.-688с.

Формат 60x84/16. Умовн. друк. арк. 2,55. Зам. № 27. Наклад 50 прим.
Видавництво УжНУ «Говерла».
88000, м. Ужгород, вул. Капітульна, 18. E-mail: goverla-print@uzhnu.edu.ua

Свідоцтво про внесення до державного реєстру
видавців, виготівників і розповсюджувачів видавничої продукції –
Серія Зт № 32 від 31 травня 2006 року