ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ЗАБЕЗПЕЧЕННЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ З ДИСЦИПЛІНИ

«Робототехніка»

Частина 1

Київ, 2024

Методичні рекомендації щодо забезпечення самостійної роботи студентів з дисципліни «Робототехніка»-ч.1 для студентів усіх форм навчання спеціальності 121 «Інженерія програмного забезпечення».

Методична розробка містить опис загальної інформації, завдання щодо самостійного вивчення студентами та самоконтролю, індивідуальні завдання, складаєься з трьох частин. Призначена для методичного забезпечення самостійної роботи студентів денної форми навчання, які вивчають навчальну дисципліну «Робототехніка»

Розробники:

ГОРДІЄНКО Олександр Миколайович, кандидат технічних наук, доцент кафедри комп'ютерних інформаційних систем та технологій Інститут комп'ютерно-інформаційних технологій та дизайну ПрАТ «ВНЗ «Міжрегіональна Академія управління персоналом»

КОВАЛЬ Аліна Олександрівна, викладач кафедри комп'ютерних інформаційних систем та технологій Інститут комп'ютерно-інформаційних технологій та дизайну ПрАТ «ВНЗ «Міжрегіональна Академія управління персоналом»

Розглянуто та ухвалено на засіданні кафедри комп'ютерних інформаційних систем та технологій Інституту комп'ютерно-інформаційних технологій та дизайну ПрАТ «ВНЗ «Міжрегіональна Академія управління персоналом»

Протокол №__ від «__» _____ 2024 р.

Завідувач кафедри _____ КАВУН Сергій Віталійович

Гордієнко О.М., Коваль А.О. Методичні рекомендації щодо забезпечення самостійної роботи студентів з дисципліни «Робототехніка» .-ч.1. – К.: МАУП, 2024. – 75 с.

Зміст

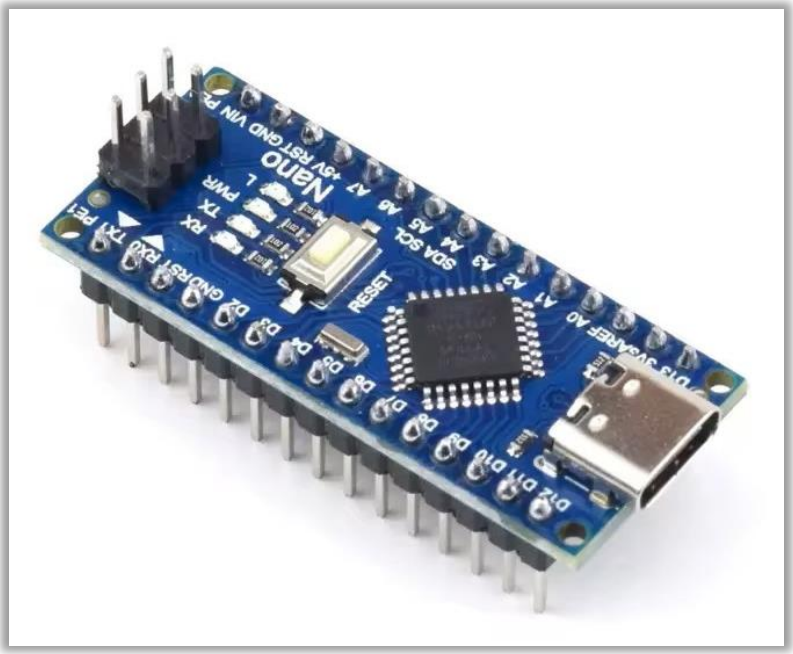
Опис та загальні характеристики

Arduino – це набір інструментів для проектування та створення електронних пристроїв, які можуть:

- Приймати сигнали від різних цифрових і аналогових датчиків;
- Підключатися до них з метою керування підконтрольними пристроями;
- працювати як автономно, так і разом з іншими системами (персональними комп'ютерами чи мікроконтролерами).

Мова програмування пристроїв Arduino створена на базі C/ C++, вона проста у вивченні, тож на сьогодні Arduino – один із найзручніших способів програмування пристроїв на мікроконтролерах. Середовище розробки

програм, Arduino IDE, має відкритий вихідний код і доступне для безкоштовного завантаження та вільного користування.



Основні характеристики плати Arduino NANO V3:

Мікроконтролер	ATmega328
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7-12 В
Вхідна напруга (гранична)	6-20 В

Цифрові Входи/Виходи	14 (6 із яких можуть використовуватись як ШІМ виходи)
Аналогові входи	8
Постійний струм через вхід/вихід	40 мА
Постійний струм для виходу 3.3 В	50 мА
Флеш пам’ять	32 Кб (ATmega328) з яких 0.5 Кб використовуються для завантажувача
Оперативна пам’ять	2 Кб (ATmega328)
EEPROM	1 Кб (ATmega328)
Тактова частота	16 МГц

Способи підключення живлення плати:

- Порт USB;
- Роз’єм Vin, для підключення зовнішньої батареї або джерела живлення.

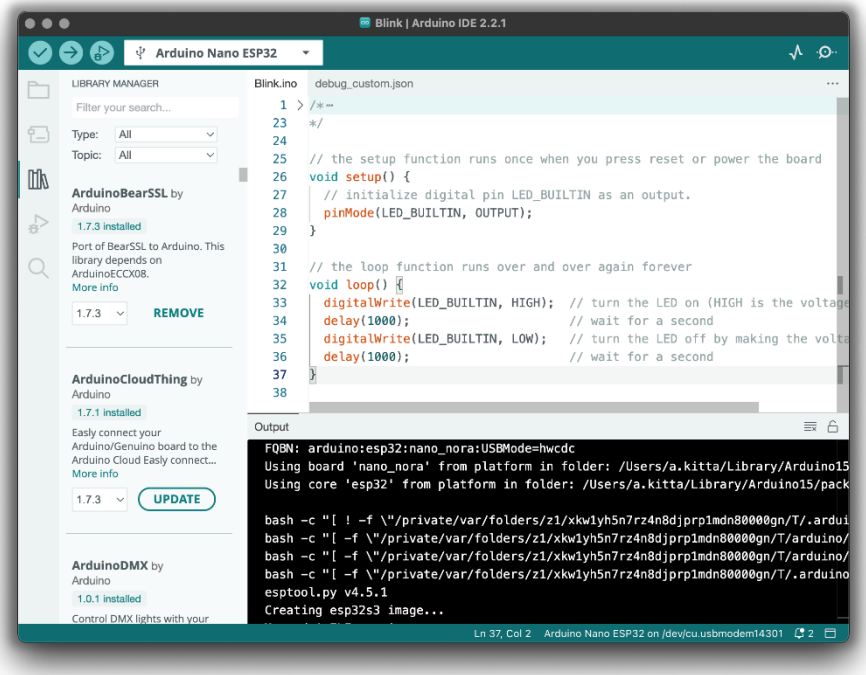
У разі підключення до різних джерел живлення, плата Arduino визначить спосіб живлення автоматично.

Встановлення програмного забезпечення

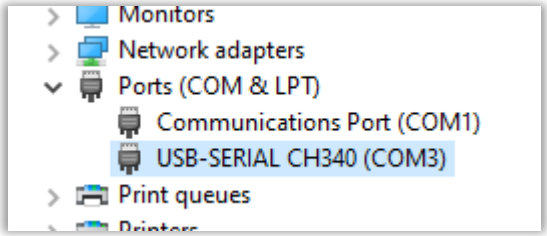
Завантажте останню версію Arduino IDE із репозиторія на GitHub:

https://github.com/arduino/arduino-ide/releases

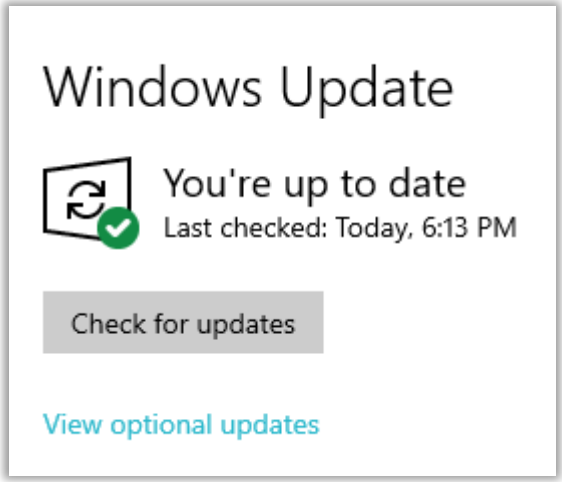
Оберіть варіант, який відповідає Вашій операційній системі. Наприклад, arduino-ide_2.3.2_Windows_64bit.exe. Встановіть Arduino IDE користуючись підказками інсталятора.



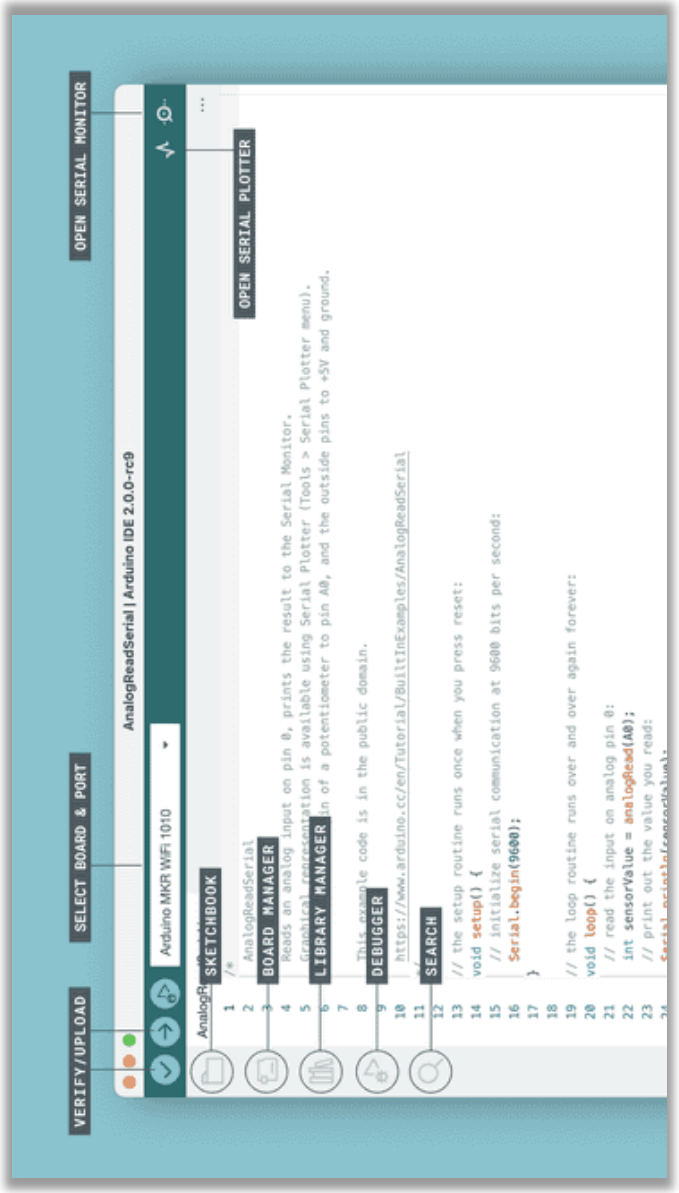
Під’єднайте плату Arduino за допомогою USB кабелю до Вашого комп’ютера. Система знайде обладнання CH340 та встановить необхідний драйвер.



Якщо драйвер не встановився, то відкрийте “Windows Update” та натисніть “Check for updates”.



Знайомство із середовищем розробки



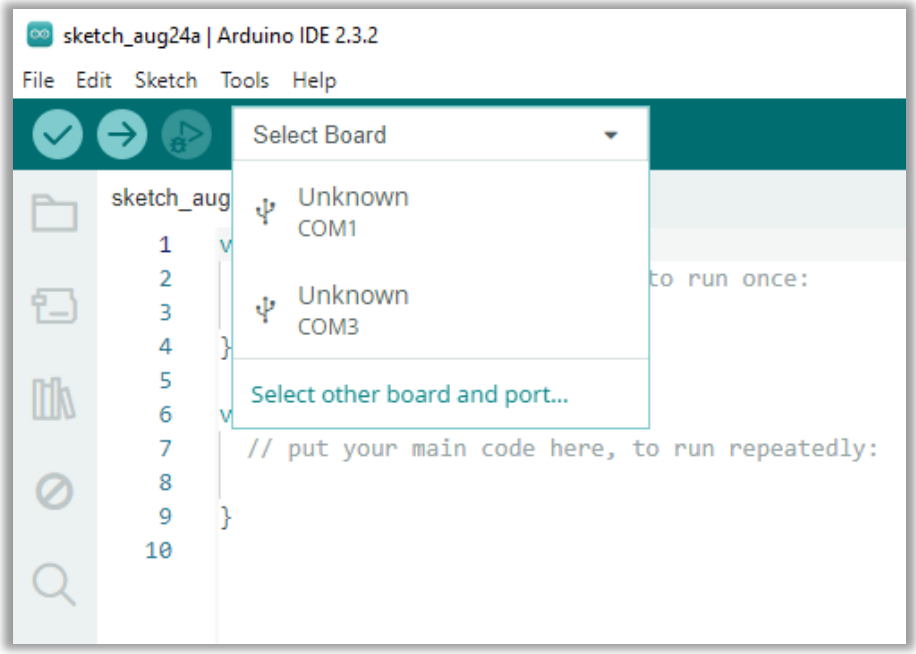
Arduino IDE — це універсальний редактор із багатьма функціями. Ви можете безпосередньо встановлювати бібліотеки, синхронізувати свої скетчі з Arduino Cloud, відлагоджувати скетчі та багато іншого. Ось деякі з основних функцій:

- Verify / Upload - компілювати та завантажити код на плату Arduino.
- Select Board & Port - виявлені плати Arduino автоматично з’являються тут разом із номером порту.
- Sketchbook -тут ви знайдете всі свої скетчі, які локально зберігаються на вашому комп’ютері. Також ви можете синхронізуватися з Arduino Cloud.
- Boards Manager - пакети драйверів Arduino та сторонніх розробників, які можна встановити.
- Library Manager - переглядати тисячі бібліотек Arduino, створених Arduino та її спільнотою.
- Debugger - тестування та налагодження програм у реальному часі.
- Search - пошук за ключовим словом у вашому коді.
- Open Serial Monitor - відкриває інструмент Serial Monitor, як нову вкладку на консолі.

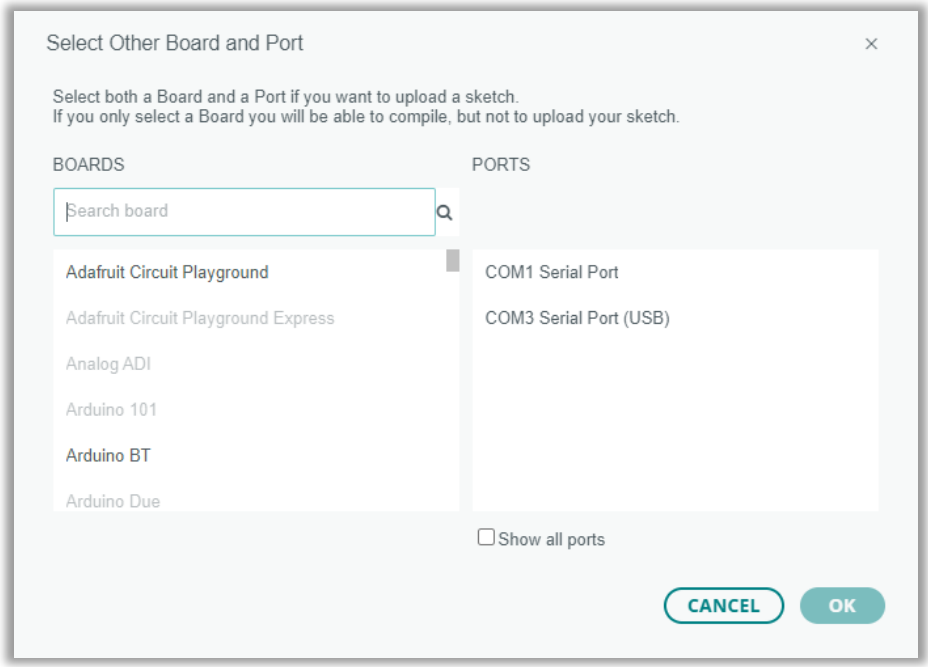
Налаштування Arduino IDE

Процес розробки електронних пристроїв:

1. Написання програми
2. Завантаження програми у мікроконтроллер
3. Тестування/відлагодження



Спершу потрібно визначити плату, що підключена до комп’ютера та порт через який підключена плата. Натисніть на випадаюче меню “Select Board” -> “Select other board and port…” – відкриється нове вікно.



В колонці “Boards” оберіть “Arduino Nano”. В колонці “Ports” – порт до якого підключена плата.

В нашому випадку це “COM3 Serial Port (USB)”.

Щоб визначити порт, відключіть плату від комп’ютера та підключіть знову. Спостерігайте, який порт буде з’являтися у списку.

Натисніть “OK”. Середовище готове до програмування.

Перша програма Blink

Більшість Arduino мають вбудований світлодіод, яким можна керувати. В Nano V3 він підключений до цифрового контакту 13, в інших платах може бути інший контакт. Константу LED_BUILTIN встановлено на правильний світлодіодний контакт незалежно від того, яка плата використовується.

```
// функція setup запускається один раз, коли ви
// натискаєте кнопку скидання або вмикаєте плату
void setup() {
  // ініціалізувати цифровий контакт LED_BUILTIN як
  // вихідний сигнал OUTPUT.
  pinMode(LED_BUILTIN, OUTPUT);
}

// функція loop виконується знову і знову безперервно
void loop() {
  // увімкнути світлодіод (HIGH - це рівень напруги)
  digitalWrite(LED_BUILTIN, HIGH);
  // почекати секунду
  delay(1000);
  // вимкнути світлодіод, зробивши напругу низькою LOW
  digitalWrite(LED_BUILTIN, LOW);
  // почекати секунду
  delay(1000);
}
```

Напишемо програму, яка вмикає світлодіод на одну секунду, потім вимикає на одну секунду, багаторазово. Далі натискаємо кнопку перевірки скетчу “Verify”.

Програма прочитає наш код і виконає пошук помилок. Якщо все вірно і помилок не знайдено, ми побачимо таке повідомлення:

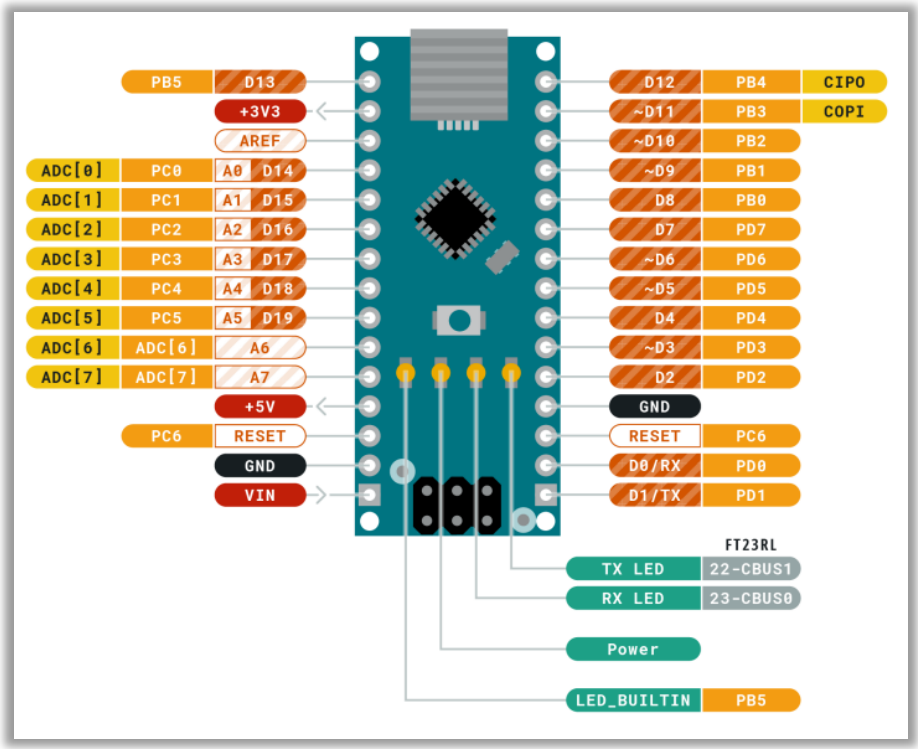
```
Sketch uses 924 bytes (3%) of program storage space.
Maximum is 30720 bytes.
Global variables use 9 bytes (0%) of dynamic memory,
leaving 2039 bytes for local variables. Maximum is 2048
bytes.
```

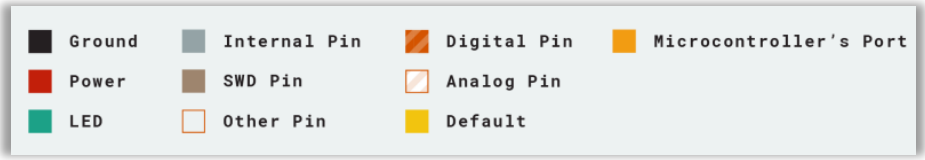
Скетч використовує 924 байтів (3%) місця зберігання для програм. Щонайбільше має бути 30720 байтів. Глобальні змінні використовують 9 байтів (0%) динамічної пам’яті, і залишають 2039 байтів для локальних змінних. Щонайбільше 2048 байтів.

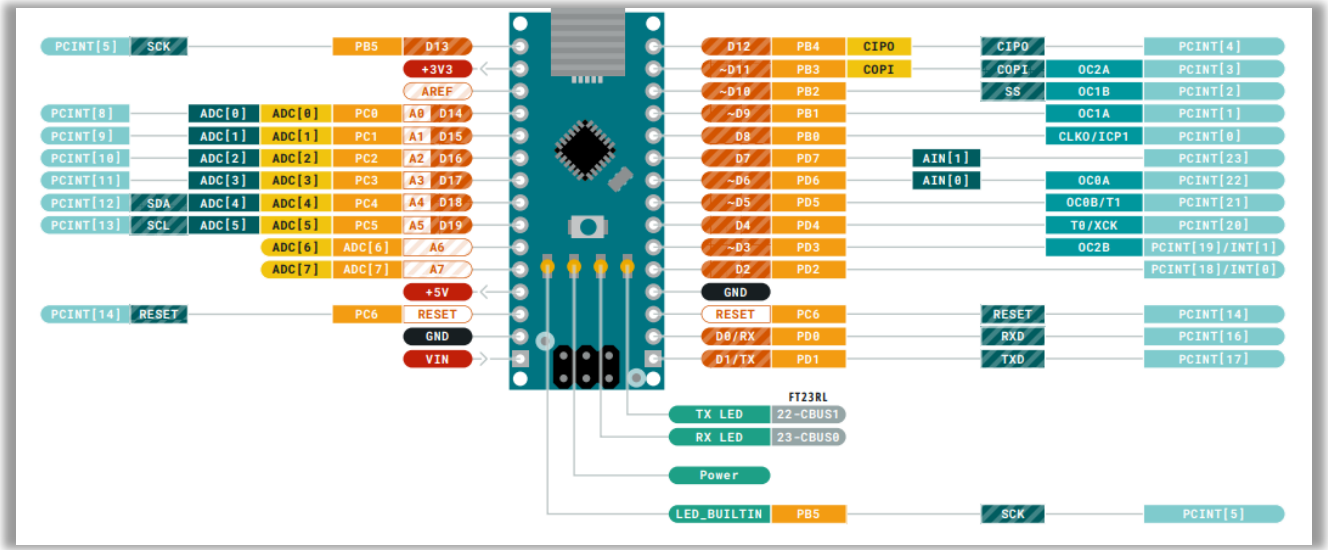
Після завершення компіляції (перевірки на помилки), в мікроконтролер можна завантажити програму. Для цього натисніть кнопку “Upload”. Плата перезавантажиться та почне блимати світлодіод. Програмування завершено.

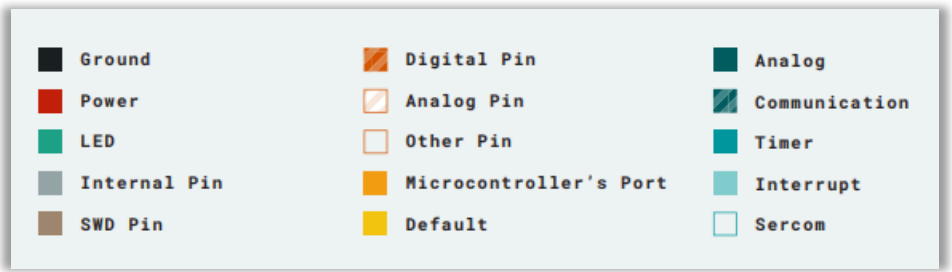
Піни мікроконтролера

Піни (контакти) мікроконтроллера:






PCINT[5] SCK PB5 D13
+3V3
AREF
PCINT[8] ADC[0] ADC[0] PC0 A0 D14
PCINT[9] ADC[1] ADC[1] PC1 A1 D15
PCINT[10] ADC[2] ADC[2] PC2 A2 D16
PCINT[11] ADC[3] ADC[3] PC3 A3 D17
PCINT[12] SDA ADC[4] ADC[4] PC4 A4 D18
PCINT[13] SCL ADC[5] ADC[5] PC5 A5 D19
ADC[6] ADC[6] A6
ADC[7] ADC[7] A7
+5V
PCINT[14] RESET PC6 RESET
GND
VIN
D12 PB4 CIPO
~D11 PB3 COPI
~D10 PB2
~D9 PB1
D8 PB0
D7 PD7
~D6 PD6
~D5 PD5
D4 PD4
~D3 PD3
D2 PD2
GND
RESET PC6
D0/RX PD0
D1/TX PD1
CIPO PCINT[4]
COPI OC2A PCINT[3]
SS OC1B PCINT[2]
OC1A PCINT[1]
CLKO/ICP1 PCINT[0]
AIN[1] PCINT[23]
AIN[0] OC0A PCINT[22]
OC0B/T1 PCINT[21]
T0/XCK PCINT[20]
OC2B PCINT[19]/INT[1]
PCINT[18]/INT[0]
RESET PCINT[14]
RXD PCINT[16]
TXD PCINT[17]
FT23RL
TX LED 22-CBUS1
RX LED 23-CBUS0
Power
LED_BUILTIN PB5
SCK PCINT[5]


Ground
Power
LED
Internal Pin
SWD Pin
Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default
Analog
Communication
Timer
Interrupt
Sercom

PIN	Soft #	Опис
TX1	1	Цифровий ввід/вивід, Serial TX
RX0	0	Цифровий ввід/вивід, Serial RX
RST		Пін перезавантаження
GND		Земля або V-
D2	2	Цифровий ввід/вивід
D3	3	Цифровий ввід/вивід, ШІМ
D4	4	Цифровий ввід/вивід
D5	5	Цифровий ввід/вивід, ШІМ
D6	6	Цифровий ввід/вивід, ШІМ
D7	7	Цифровий ввід/вивід
D8	8	Цифровий ввід/вивід
D9	9	Цифровий ввід/вивід, ШІМ
D10	10	Цифровий ввід/вивід, ШІМ, SPI SS
D11	11	Цифровий ввід/вивід, ШІМ, SPI MOSI
D12	12	Цифровий ввід/вивід, SPI MISO
D13	13	Цифровий ввід/вивід, LED, SPI SCK
3.3V		3.3 В
REF		Пін опорної напруги
A0	A0 або 14	Аналоговий пін із 8-ми бітним АЦП
A1	A1 або 15	Аналоговий пін із 8-ми бітним АЦП
A2	A2 або 16	Аналоговий пін із 8-ми бітним АЦП
A3	A3 або 17	Аналоговий пін із 8-ми бітним АЦП

A4	A4 або 18	Аналоговий пін із 8-ми бітним АЦП, I2C SDA
A5	A5 або 19	Аналоговий пін із 8-ми бітним АЦП, I2C SCL
A6	A6	Аналоговий пін із 8-ми бітним АЦП
A7	A7	Аналоговий пін із 8-ми бітним АЦП
5V		5 В або V+
RES		Пін перезавантаження
GND		Земля або V-
VIN		Пін живлення

PIN – маркування на платі

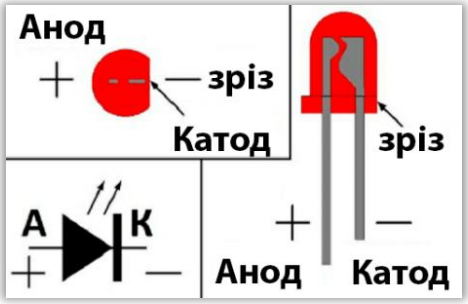
Soft # - номер піна в програмі

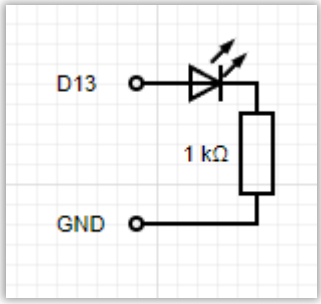
Плата має 14 цифрових пінів, позначені літерою "D". Вони можуть бути як входом, так і виходом. Робоча напруга цих пінів становить 5 В.

Аналогові піни на платі позначені літерою "A". Ці піни є входами. Вони вимірюють напругу, що надходить на них, і повертають значення від 0 до 1023 при використанні функції analogRead(). Ці піни вимірюють напругу з точністю до 0,005В.

Світлодіоди: підключення та налаштування

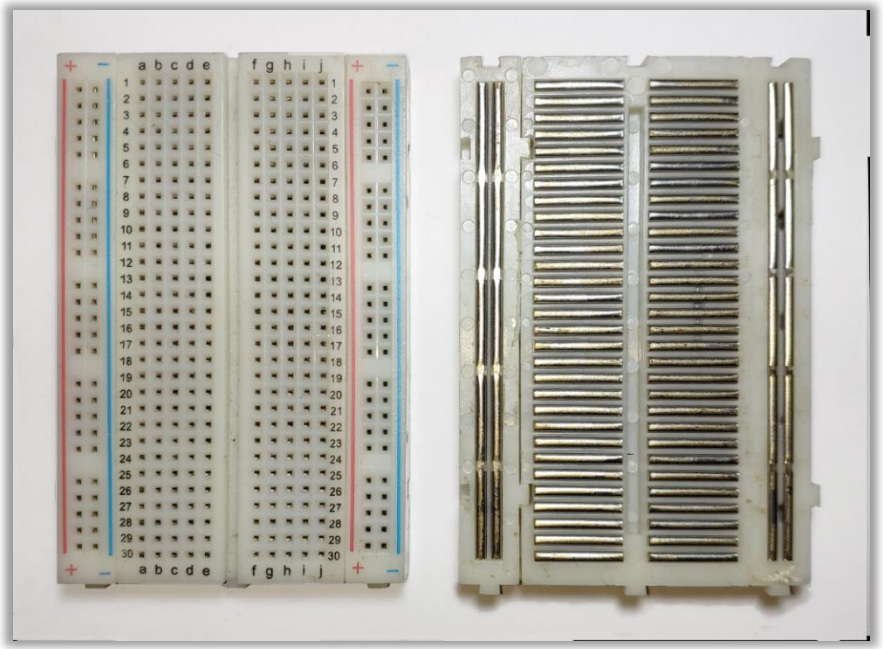
Щоб опанувати цифрові піни використаємо світлодіоди. Світлодіод – це напівпровідник, що має полярність. Якщо його неправильно під'єднати, він не буде працювати. Світлодіод має два контакти - анод (довга ніжка) і катод (коротка ніжка). Для того, щоб світлодіод засвітився, необхідно анод під'єднати до плюса, а катод - до мінуса. Окрім цього, для забезпечення нормального режиму роботи, треба послідовно під'єднати до ланцюга із світлодіодом резистор номіналом 220 Ом – 1 кОм. При цьому не важливо, до якого саме виводу світлодіода буде під'єднано резистор, головне те, що, він повинен бути у ланцюгу. Резистор обмежує струм, що протікає через світлодіод.



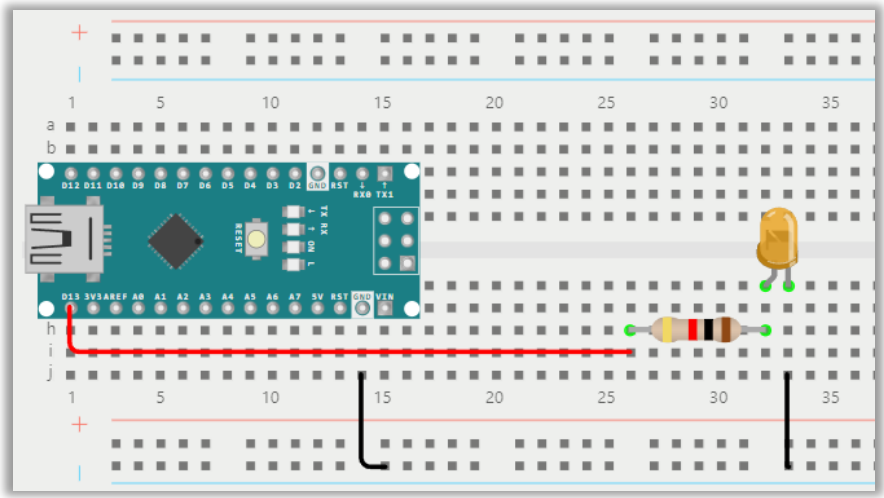


Вставте Arduino в макетну плату, як показано на малюнку.

Ми вже запрограмували плату Arduino на блимання світлодіода. Давайте під’єднаємо до піна 13 зовнішній світлодіод. Для цього в макетну плату, в будь-яке вільне місце, вставте світлодіод так, щоб кожна з ніжок стояла в окремому рядку макетної плати. В макетній платі отвори згруповані по п’ять в один рядок. Бічні - по вертикалі, по всій довжині і використовуються як шини живлення плюса, і мінуса.



Використайте червоний провід та резистор номіналом 1кОм, щоб з’єднати анод світлодіода з контактом на платі D13. Катод світлодіода під’єднайте до GND контакту плати.



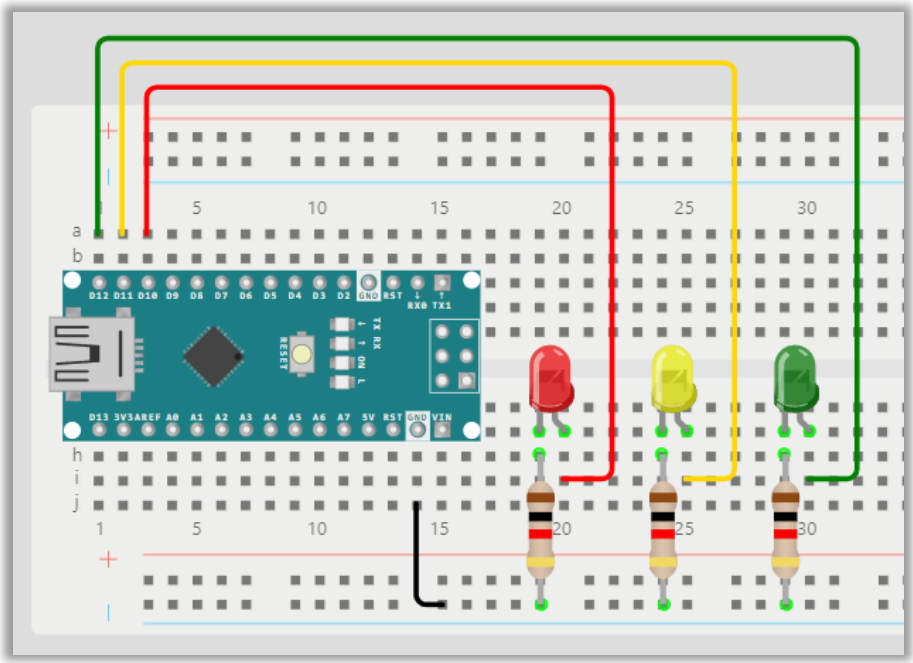
Після правильного підключення елементів до плати та подачі на неї живлення, світлодіод почне блимати синхронно зі світлодіодом на платі.

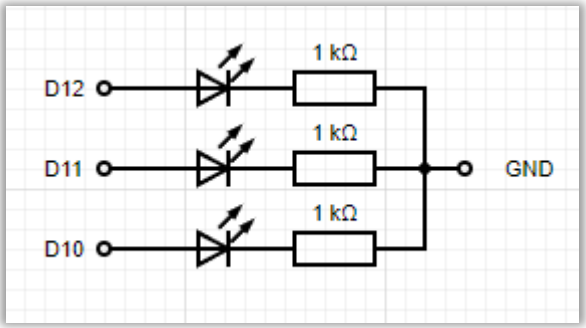
Завдання 1.

Змініть код програми таким чином, щоб світлодіод блимав у режимі “SOS”: три довгих сигнали та три коротких. Затримка між сигналами повинна бути однаковою, а тривалість “тире”- в три рази довша, аніж тривалість “крапки”.

Розробляємо світлофор

З’єднаємо ще кілька світлодіодів до плати управління та зробимо “світлофор”. Під’єднайте світлодіоди до пінів D12, D11, D10, як на малюнку нижче:





Назвемо піни 12, 11, 10 як greenPin, yellowPin, redPin відповідно. Використаємо кваліфікатор const для типу int. Тобто, greenPin буде містити цілечисло, яке неможливо змінити. Наприклад, CONST INT GREENPIN = 12;

```
const int greenPin  = 12;
const int yellowPin = 11;
const int redPin    = 10;

void setup() {
  pinMode(greenPin,  OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(redPin,    OUTPUT);
}

void loop() {
  digitalWrite(redPin,    HIGH); //червоний
  digitalWrite(yellowPin, LOW);  //стояти
  digitalWrite(greenPin,  LOW);
  delay(3000);

  digitalWrite(redPin,    HIGH); //червоний та жовтий
  digitalWrite(yellowPin, HIGH); //готуйсь
  digitalWrite(greenPin,  LOW);
  delay(1500);

  digitalWrite(redPin,    LOW);  //зелений
  digitalWrite(yellowPin, LOW);  //рухайся
  digitalWrite(greenPin,  HIGH);
  delay(5000);
```

```
  digitalWrite(redPin,    LOW);  //вимикаємо
  digitalWrite(yellowPin, LOW);  //та вмикаємо три
  digitalWrite(greenPin,  LOW);  //рази зелений
  delay(500);                    //увага, скоро жовтий

  digitalWrite(redPin,    LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,  HIGH);
  delay(500);
  digitalWrite(redPin,    LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,  LOW);
  delay(500);
  digitalWrite(redPin,    LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,  HIGH);
  delay(500);
  digitalWrite(redPin,    LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,  LOW);
  delay(500);
  digitalWrite(redPin,    LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,  HIGH);
  delay(500);

  digitalWrite(redPin,    LOW);  //жовтий
  digitalWrite(yellowPin, HIGH); //рух заборонено
  digitalWrite(greenPin,  LOW);
  delay(3000);
}
```

Щоб оптимізувати код та зменшити його розмір використаємо функції. Створимо функцію `controllLeds`.

```
const int greenPin  = 12;
const int yellowPin = 11;
const int redPin    = 10;

void setup() {
  pinMode(greenPin,  OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(redPin,    OUTPUT);
}
void loop() {
  controllLeds(HIGH, LOW,  LOW,  3000); //червоний
  controllLeds(HIGH, HIGH, LOW,  1500); //чер+жовтий
  controllLeds(LOW,  LOW,  HIGH, 5000); //зелений
  controllLeds(LOW,  LOW,  LOW,  500); //блимаємо
  controllLeds(LOW,  LOW,  HIGH, 500); //зеленим
  controllLeds(LOW,  LOW,  LOW,  500);
  controllLeds(LOW,  LOW,  HIGH, 500);
  controllLeds(LOW,  LOW,  LOW,  500);
  controllLeds(LOW,  LOW,  HIGH, 500);
  controllLeds(LOW,  HIGH, LOW,  3000); //жовтий
}

void controllLeds(int r, int y, int g, int d){
  digitalWrite(redPin,    r);
  digitalWrite(yellowPin, y);
  digitalWrite(greenPin,  g);
  delay(d);
}
```

Функція - це фрагмент програмного коду, до якого можна звертатися з різних місць програми. Основне покликання функцій в програмуванні - це оптимізація багаторазового виконання однотипного шматка програмного коду, що дозволяє уникнути повторюваних сегментів коду в програмі і марного використання наявної пам'яті. При цьому кожен раз звертаючись до функції, ми заново використовуємо код, який написано в одному екземплярі. І ще важливо те, що результат виконання функції буде змінюватися зі зміною вхідних аргументів переданих у функцію. При цьому так само стає зручно вносити правки в цей повторюваний код, адже він лежить в одному місці - в тілі функції.

Опис функції у скетчі в загальному вигляді має наступну структуру. Тип зворотніх даних, ім'я функції, вхідні аргументи, тіло функції укладене в фігурні лапки.

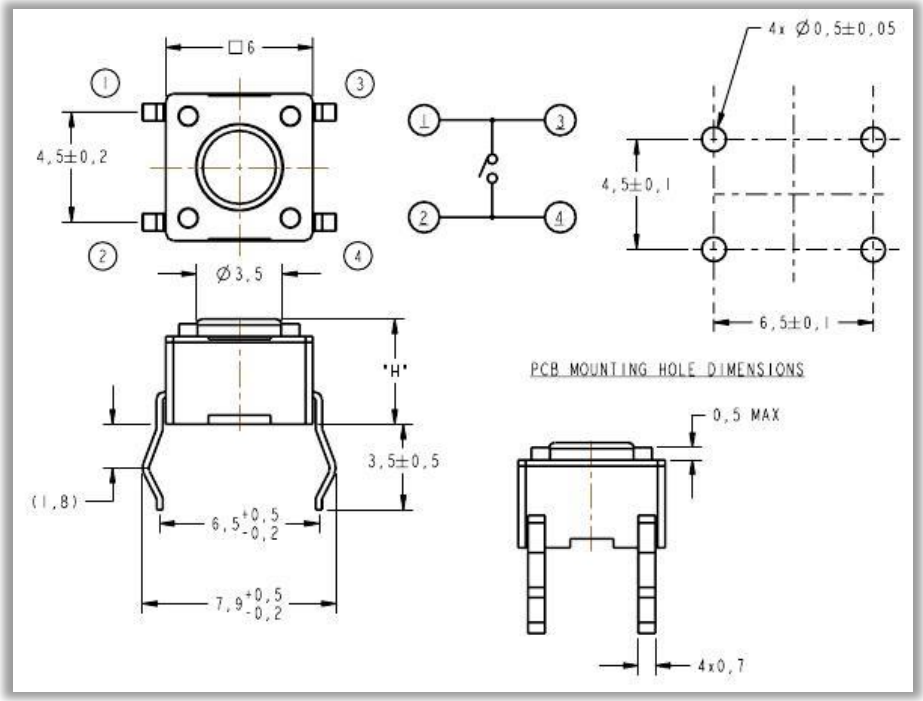
```
returnType functionName(argType argument, ...) { ... }
```

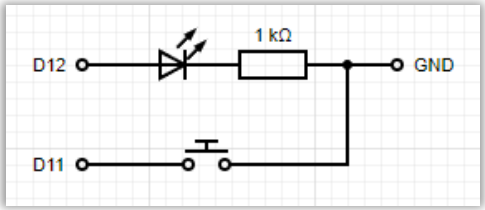
Завдання 2.

Зберіть схему і розробіть код для реалізації гірлянди із 6 світлодіодів, в якій світлодіоди будуть вмикатись і вимикатись по черзі в різних режимах.

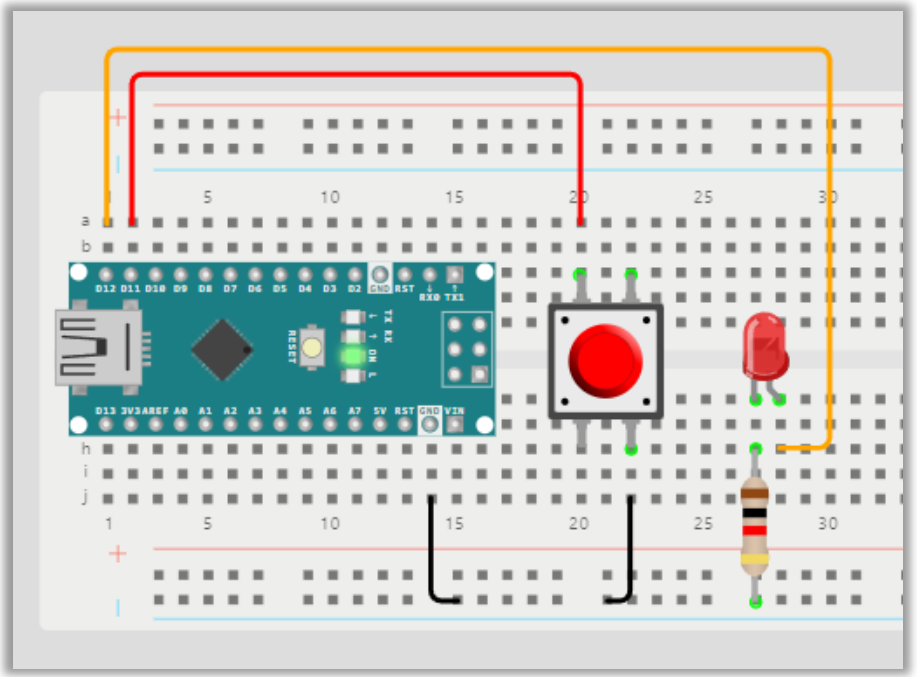
Зовнішнє управління світлодіодом

Arduino має 14 цифрових виводів, які можуть працювати вихід/вхід. Ми можемо не лише керувати пристроями, але й приймати сигнали з інших пристроїв чи модулів. Джерелом простого сигналу можна використати тактову кнопку, під'єднану до мінуса і до цифрового піна. Використання кнопки дозволяє дозволяє зробити наш пристрій інтерактивним.





Під час активації (натискання) кнопки, механіка всередині корпуса з’єднує контакти між собою і кнопка починає пропускати через себе електричний струм. Для усунення впливу наведень і запобігання помилкових спрацьовувань, потрібно встановити підтягувальний резистор номіналом від десятків до сотень кОм, або



використовувати вбудований в кристал мікроконтролера резистор на 20 кОм. Для цього в void setup() потрібно вказати pinMode(номер_піна, INPUT_PULLUP).

```
const int pinLed = 12; //пін світлодіода
const int pinKey = 11; //пін кнопки

void setup(){
  pinMode(pinLed, OUTPUT);
  //пін із світлодіодом - вихід
  pinMode(pinKey, INPUT_PULLUP);
  //пін із кнопкою – вхід із підтягуючим резистором
}

void loop(){
  //перевіряємо стан піна кнопки
  if (digitalRead(pinKey) == LOW){
    digitalWrite(pinLed, HIGH); //вмикаємо світлодіод
  }else{
    digitalWrite(pinLed, LOW); //вимикаємо світлодіод
  }
}
```

Пін кнопки буде підтягнутий до напруги +5В через внутрішній резистор. Тобто, натискаючи кнопку напруга на pinKey впаде до 0В (LOW), відпускаючи - напруга буде +5В (HIGH).

“Брязкіт контактів”

Натискання кнопки супроводжується багаторазовим замиканням контактів. Деякий час контакти «підстрибують» під час зіткнень, розмикаючи чи замикаючи електричний ланцюг. Цей процес називають “брязкотом контактів”. Через нього в момент натискання на кнопку, стан світіння світлодіода декілька разів буде змінюватись. Змінимо процедуру перевірки стану кнопки. Використаємо змінну типу boolean, яка приймає всього два значення: true або false. Зробимо затримку перед наступною перевіркою. Цього буде достатньо, що подолати брязкіт.

```
const int pinLed = 12;
const int pinKey = 11;

boolean btnPressed = false;
void setup(){
  pinMode(pinLed, OUTPUT);
  pinMode(pinKey, INPUT_PULLUP);
}

void loop(){
  btnPressed = digitalRead(pinKey) == LOW;
  digitalWrite(pinLed, btnPressed);
  delay(300);
}
```

Затримки між командами

Зробимо програму, яка буде змінювати режим роботи світлодіода. Тобто, кнопка натиснута – світлодіод блимає швидко. Кнопка відпущена – блимає повільно.

```
const int pinLed = 12;
const int pinKey = 11;

boolean btnPressed = false;

void setup(){
  pinMode(pinLed, OUTPUT);
  pinMode(pinKey, INPUT_PULLUP);
}
void loop(){
  btnPressed = digitalRead(pinKey) == LOW;
  if(btnPressed){
    digitalWrite(pinLed, HIGH);
    delay(500);
    digitalWrite(pinLed, LOW);
    delay(500);
  }else{
    digitalWrite(pinLed, HIGH);
    delay(2000);
    digitalWrite(pinLed, LOW);
    delay(2000);
  }
}
```

Поки виконується код повільного блимання наша кнопка не реагуватиме на натискання. Потрібно затиснути її та чекати, коли відбудеться перевірка стану кнопки. Це не зручно, тому потрібно використовувати інший підхід.

```
const int pinLed = 12;
const int pinKey = 11;

//поточне натискання кнопки
boolean btnPressed = false;
//попереднє натискання кнопки
boolean lastBtnPressed = false;
//стан світлодіода
boolean ledOn = false;
//час коли треба змінити стан світлодіода
unsigned long nextChange = 0;

void setup(){
  pinMode(pinLed, OUTPUT);
  pinMode(pinKey, INPUT_PULLUP);
}

void loop(){
  btnPressed = digitalRead(pinKey) == LOW;
  //якщо кнопку натиснули або відпустили
  if(btnPressed != lastBtnPressed){
    //реєстуємо зміну стану кнопки
    lastBtnPressed = btnPressed;
    //змінюємо час зміни стану світлодіода
    nextChange = 0;
```

```
    //чекаємо закінчення брязкотіння контактів
    delay(300);
  }
  //якщо поточний час більше за заданий
  if(millis() > nextChange){
    //перевіряємо який режим блимання встановити
    if(lastBtnPressed){
      nextChange = millis() + 500;
    }else{
      nextChange = millis() + 2000;
    }
    //інвертуємо стан світлодіода
    ledOn = !ledOn;
    digitalWrite(pinLed, ledOn);
  }
}
```

Функція millis() зберігає значення кількості мілісекунд, які пройшли з моменту початку роботи пристрою.

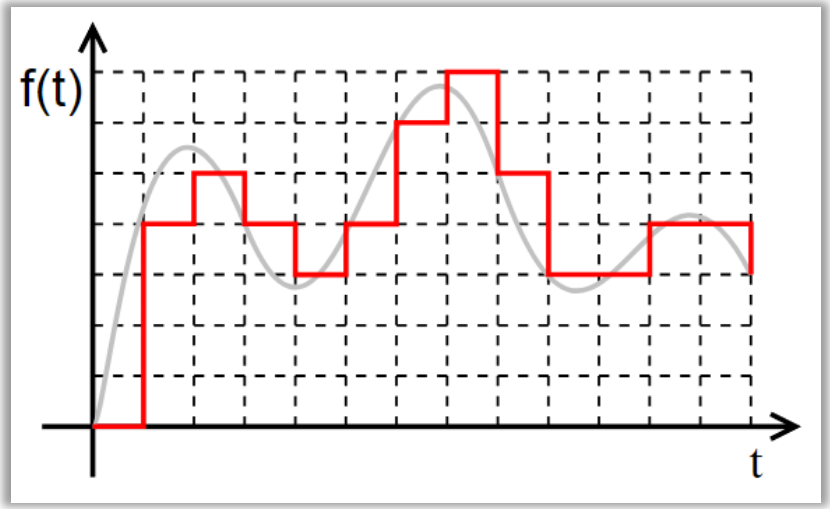
У прикладі вище ми встановлюємо, коли наступне перемикання світлодіода та спостерігаємо за зміном стану кнопки.

Завдання 3.

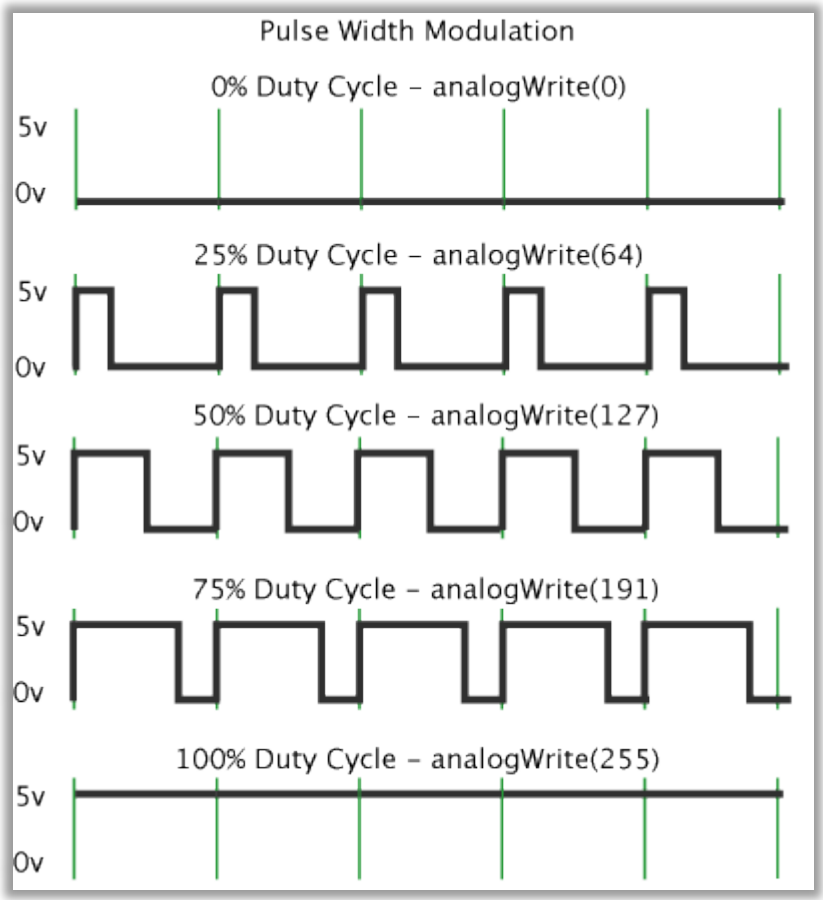
Зробіть перемикання режиму роботи світлодіода одиничним натисканням кнопки. Використайте останній приклад для модифікації.

Основи ШІМ (широтно-імпульсна модуляція)

Широтно-імпульсна модуляція, або ШІМ, — це метод отримання аналогового сигналу за допомогою цифрового управління. Цифрове управління використовується для створення прямокутної хвилі, сигналу, що перемикається між увімкненням (5В) і вимкненням (0В). Задати можна тільки час увімкнення/вимкнення. Тривалість «ввімкнення» називається шириною імпульсу. Щоб отримати різні аналогові значення, ви змінюєте або модулюєте цю ширину імпульсу. Якщо ви повторюєте цю схему ввімкнення-вимкнення достатньо швидко, наприклад, зі світлодіодом, результатом буде таке, ніби

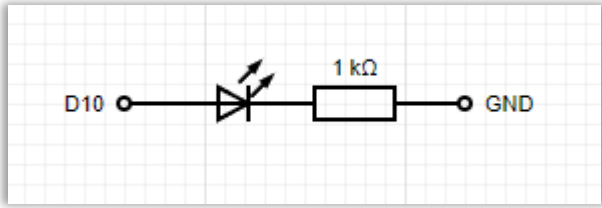


сигнал є постійною напругою від 0 до Vcc, що контролює яскравість світлодіода.



На малюнку вище зелені лінії представляють звичайний період часу. Ця тривалість або період є зворотною частоті ШІМ. Іншими словами, з частотою ШІМ Arduino близько 500 Гц, зелені лінії становитимуть 2 мілісекунди кожна. Плата Arduino здатна формувати

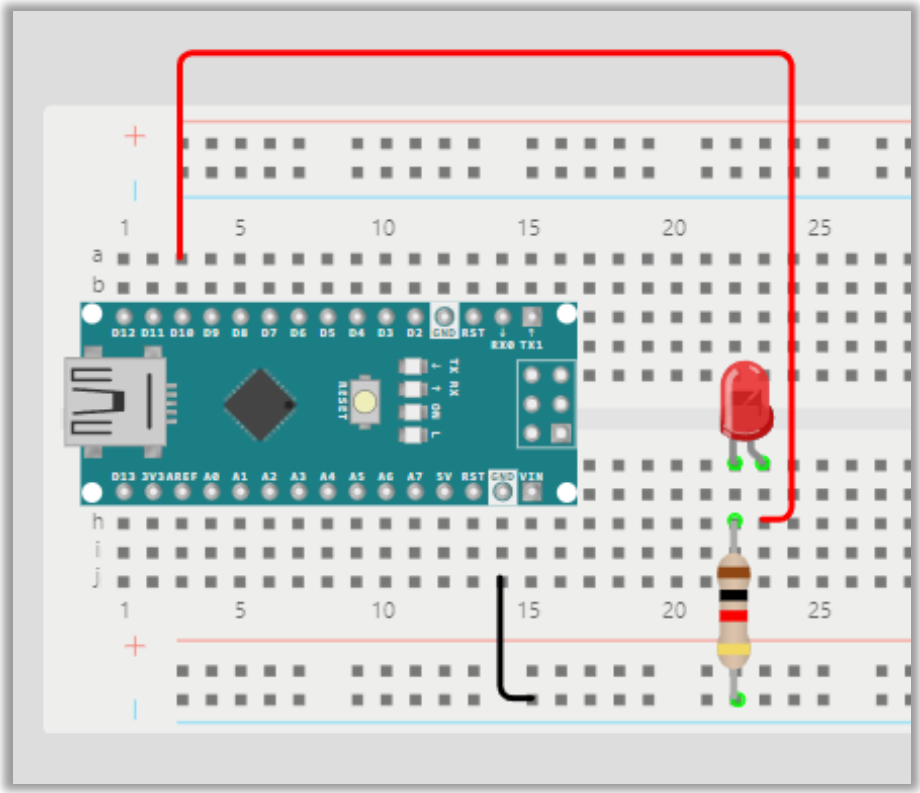
аналогову напругу в вигляді ШІМ сигналу на виводах 3, 5, 6, 9, 10 і 11. Виклик analogWrite() виконується за шкалою від 0 до 255, так що analogWrite(255) встановлює 100% робочого циклу (завжди ввімкнено), а analogWrite(128) – це 50% робочого циклу (в половину часу).



Складемо схему та розробимо програму в якій буде змінюватись яскравість світіння світлодіода:

```
const int pinLed = 10;
//назва піна для підключення світлодіода

void setup() {
  //налаштовувати пін як вихід при функції
  // analogWrite() необов’язково
}

void loop() {
  analogWrite(pinLed, 0); //0% потужності
  delay(250);

  analogWrite(pinLed, 64); //25% потужності
```

```
  delay(250);

  analogWrite(pinLed, 128); //50% потужності
  delay(250);

  analogWrite(pinLed, 192); //75% потужності
  delay(250);

  analogWrite(pinLed, 255); //100% потужності
  delay(250);
}
```



Напишемо код, в якому яскравість світлодіода буде плавно збільшуватися , підключення при цьому залишатиметься таким же.

```
const int pinLed = 10;

void setup(){}

void loop() {
  for (int i=0; i<=255; i++){
    // знак ++ називається інкремент і відповідає
    // запису i=i+1;
    analogWrite(pinLed, i);
    // запалюємо світлодіод на величину рівну i
    delay(10);
  }
}
```

В цьому скетчі ми знайомимося з циклом for, який створює змінну “i“. Змінна “i“ відповідає за яскравість світлодіода і збільшується на одиницю вказану кількість разів (у нашому випадку – 255).

Завдання 4.

Зберіть схему і створіть програму для регулювання яскравості світлодіода за допомогою двох кнопок (кнопка1- збільшити яскравість, кнопка2- зменшити яскравість).

Які бувають світлодіоди: огляд основних типів і характеристик

Світлодіод - це напівпровідниковий діод, "упакований" в оптичну оболонку (лінзу), який випромінює світло при проходженні струму. В залежності від типу напівпровідника і лінзи світлодіода, він випромінює світло різної потужності, спрямованості і в широкому колірному спектрі.

Світлодіоди бувають індикаторні та освітлювальні.

Перші використовуються, переважно, для світлової індикації роботи технічного обладнання, підсвічування приладової панелі, дисплеїв, рекламних вивісок, виробництва новорічних гірлянд, світлових табло. Цей тип світлодіодів характеризується відносно невеликими потужністю і яскравістю.

Другі володіють високою інтенсивністю світлового випромінювання і, відповідно назві, використовуються для виробництва освітлювальних приладів, інструментів внутрішнє освітлення, автомобільних фар, ліхтарів і світильників...

Існують ультрафіолетові світлодіоди, які отримали широке поширення в самих різних пристроях: від

ультрафіолетових принтерів до медичного та криміналістичного обладнання.

Крім того, світлові діоди використовуються в сільському господарстві в якості штучного світла для ефективного вирощування рослин.

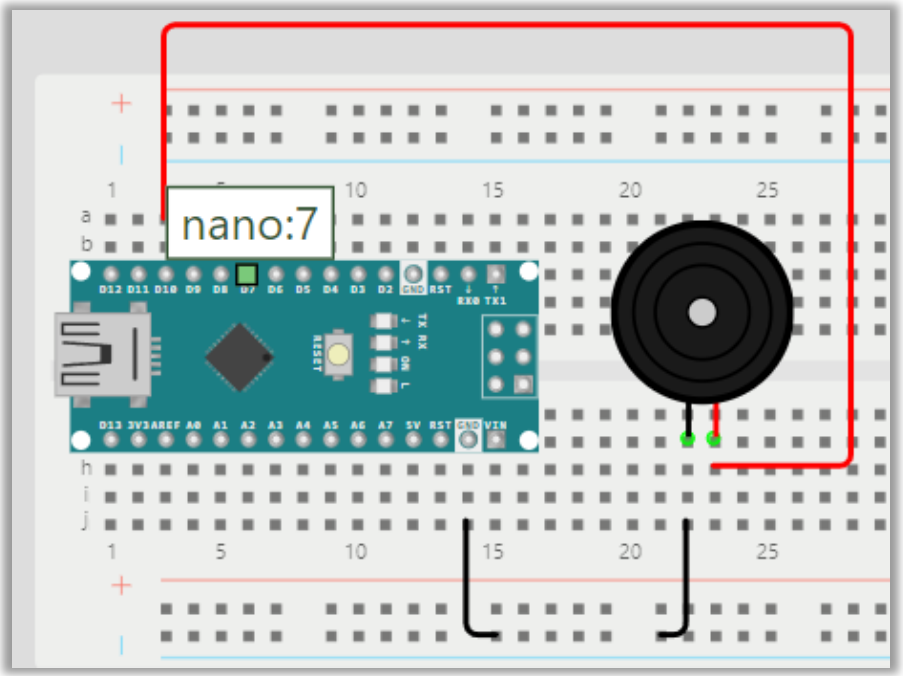
Величина струму одного світлодіода, в переважній більшості випадків, дорівнює 20мА. Струм споживання світлодіодного приладу визначається кількістю світлодіодів в ньому. Для нормальної і довговічної роботи діода необхідно забезпечувати стабільність величини струму. Для цього в конструкції приладів передбачаються стабілізатори і резистори.

Напруга повинна відповідати величині струму, інакше діод буде випромінювати слабке світіння або перегорить. Цікаво те, що напруга безпосередньо залежить від матеріалу, з якого виготовлений напівпровідник, а значить - його кольору. Таким чином, за розміром напруги можна визначити колірний спектр світлодіода, а за кольором - розмір напруги.

Інфрачервоний - 1,1-1,6 V	Червоний - 1,5-2,6 V
Помаранчевий - 1,7-2,8 V	Жовтий - 1,7-2,5 V
Зелений - 1,7-4,0 V	Блакитний - 3,2-4,5 V
Білий - 2,7-4,3 V	

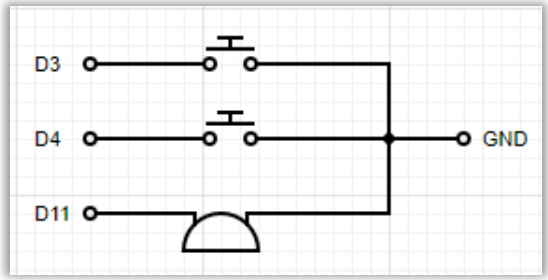
Бузер: звукова індикація

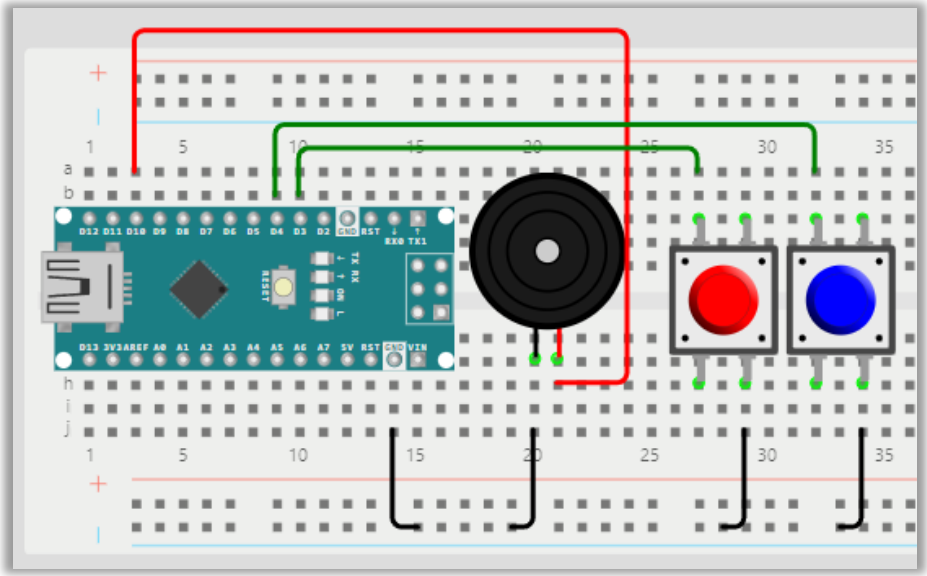
Окрім світлової індикації можна також застосовувати звукову. Бузер це прилад, який генерує звуки. Для того, щоб увімкнути звук, потрібно застосовувати спеціальну функцію tone(pin, frequency, duration). В дужках слід вказати пін підключення бузера і частоту згенерованого сигналу в герцах, а також тривалість звучання в мілісекундах (по бажанню). Для припинення генерації звуку використовують функцію noTone(pin).



```
const int pinBuzzer = 10;
void setup() { }
void loop() {
  tone(pinBuzzer, 1000, 1000);
  delay(2000);
}
```

Для керування бузером, можна під'єднати декілька кнопок для регулювання звуку:





```
const int pinBuzzer = 10;
const int pinKey1 = 3;
const int pinKey2 = 4;
int val = 1000; //частота бузера

void setup() {
  pinMode(pinKey1, INPUT_PULLUP); //кнопка 1 як вхід
  pinMode(pinKey2, INPUT_PULLUP); //кнопка 2 як вхід
}

void loop() {
  tone(pinBuzzer,val); //запуск генерації

  if(digitalRead(pinKey1) == LOW){
    delay(100); //чекаємо 100 мілісекунд і
    val= val + 500; //додаємо 500 до значення val
  }

  if(digitalRead(pinKey2) == LOW){
    delay(100); //чекаємо 100 мілісекунд і
    val= val - 500; //зменшуємо val на 500
  }
}
```

Завдання 5.

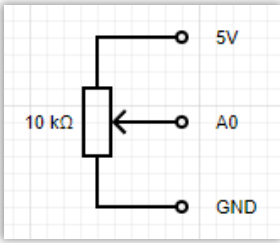
Створіть програму для відтворення мелодій.

Використайте кнопки, щоб перемикати мелодії.

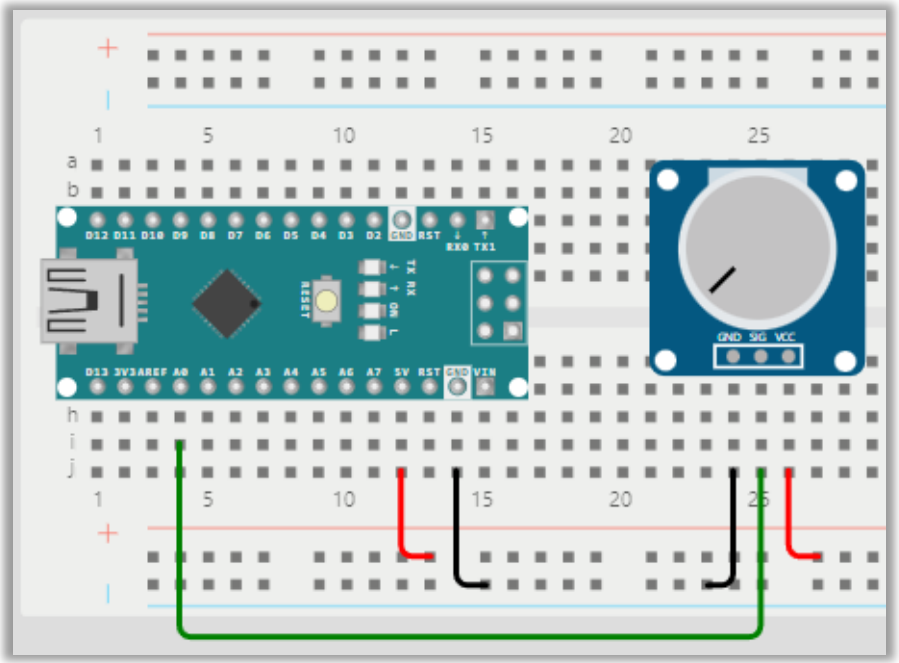
Аналого-цифровий перетворювач

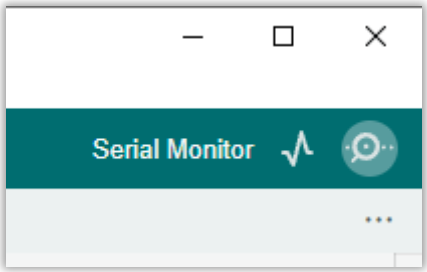
АЦП - пристрій, що перетворює вхідний аналоговий сигнал в дискретний код (цифровий сигнал), який кількісно характеризує амплітуду вхідного сигналу. Arduino Nano має 8 пінів для зчитування аналогового сигналу. Результатом вимірювання є число від 0 до 1023, тобто 1024 рівні вхідної напруги. Наприклад, зберемо схему з потенціометром, який включений між GND та +5В на крайніх контактах. Середній контакт буде виходом з напругою від 0 до +5В. Тобто максимальна напруга буде 5В, що відповідатиме 1023 рівню. Отож ділимо 5В на 1023 рівні та отримуємо ціну одного рівня 0,00488759В. Наприклад, ми встановили вороток потенціометра посередині, отримуємо вхідний рівень 512. Вирахуємо вхідну напругу:

512 x 0,00488759В ~= 2,5В



Для наочності використаємо Serial з’єднання, що відправлятиме отримане значення аналогового сигналу на монітор комп’ютера.





Для перегляду отриманих значень в програмі Arduino IDE використовуйте спеціальний монітор послідовного порту. В правому кутку натисні на кнопку Serial Monitor.

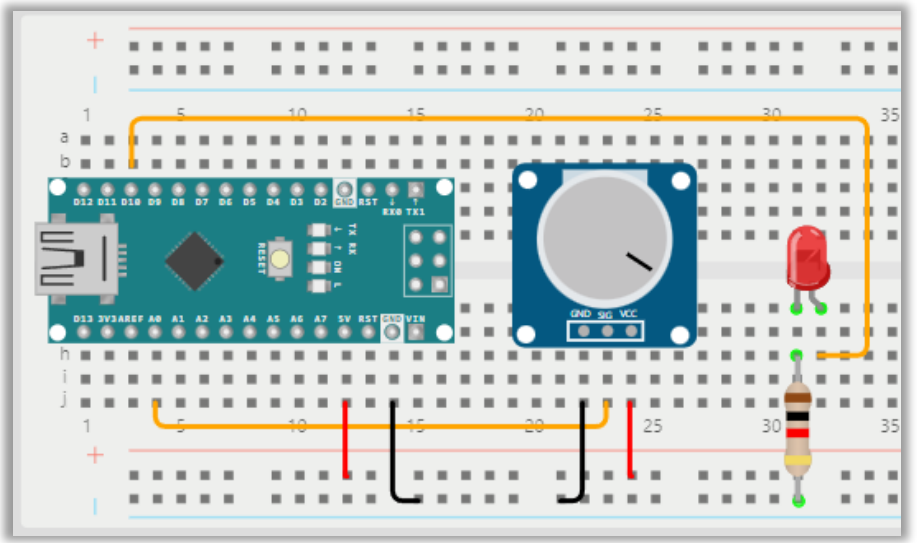
```
const int pinPot = A0;
//назва піна для під’єднання потенціометра
int val = 0;
//змінна для збереження вхідного аналогового сигналу
void setup() {
  Serial.begin(9600);
  //підключення бібліотеки Serial, для прийому і
  //передачі інформації на комп’ютер
}
void loop() {
  val = analogRead(pinPot);
  //зчитування аналогового сигналу і запис значення в
  //змінну val
  Serial.println(val);
  //відправка значення змінної val на комп’ютер
  delay(50);
  //затримка 50 мілісекунд між відправкою значення
}
```

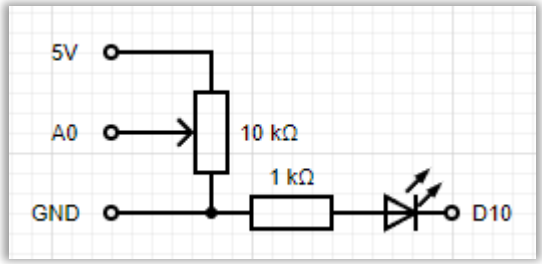
Можна помітити, що при поворотах ручки потенціометра з одного крайнього положення в інше, ми отримуємо значення в діапазоні від 0 до 1023, що відповідає вхідній напрузі.

Завдання 6.

Змініть код так, щоб ми отримували значення вхідної напруги у вольтах.

Давайте складемо схему регулювання яскравості світлодіода за допомогою потенціометра:





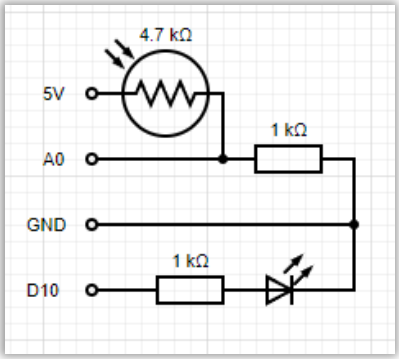
```
const int pinPot = A0; //пін потенціометра
const int pinLed = 10; //пін світлодіода
int val = 0; //значення вхідного аналогового сигналу
int brightness = 0; //значення яскравості світлодіода

void setup() {
  Serial.begin(9600);
  pinMode(pinLed, OUTPUT);
}

void loop() {
  //зчитування аналогового сигналу
  val = analogRead(pinPot);
  //відправка на комп'ютер
  Serial.println(val);
  delay(50);
  //перетворюємо значення змінної val у діапазон
  //0...255
  brightness = map(val, 0, 1023, 0, 255);
  //запалюємо світлодіод згідно значенню brightness
  analogWrite(pinLed, brightness);
}
```

Завдання 7.

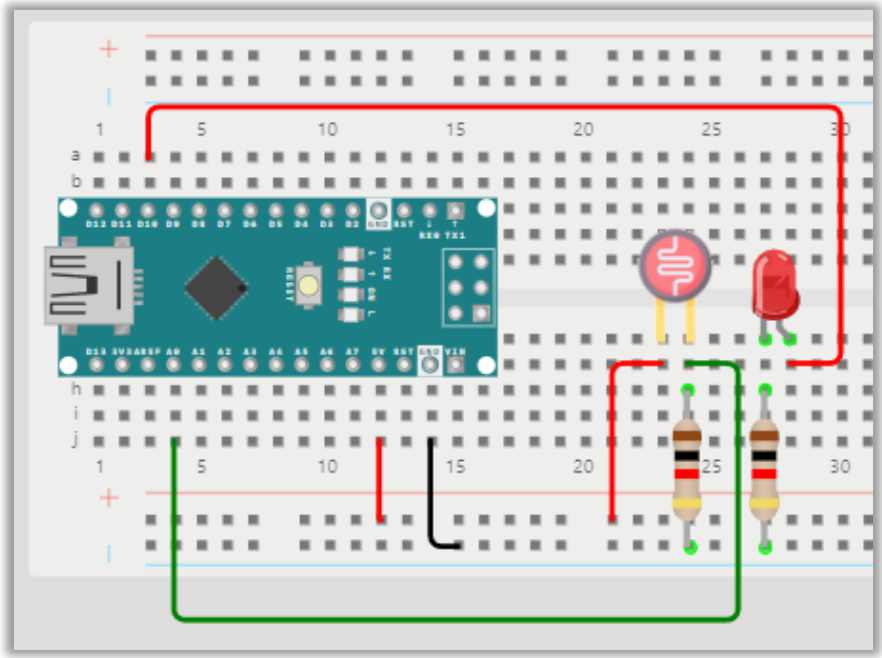
Використовуючи схему підключення із прикладу, напишіть код, в якому за допомогою потенціометра буде обиратись один із п'яти режимів світіння світлодіода.

Фоторезистор:
вимірювання рівня освітленості

Фоторезистори дозволяють визначати інтенсивність освітленості. Вони по своїй суті є резисторами, які змінюють свій опір залежно від того, скільки світла потрапляє на їхні чутливі елементи. З їх допомогою можна вирішувати завдання типу: «навколо темно чи світло?», «чи є щось перед датчиком (чи може щось обмежувати надходження світла)?», «яка з ділянок найбільш освітлена?» і т.д.



В цьому прикладі аналогове значення керує яскравістю світлодіода. Чим темніше, тим яскравіше буде світити світлодіод. Не забудьте, що світлодіод треба під’єднувати до ШІМ контакту, в цьому випадку використовується цифровий вивід D10.



```
const int pinLed = 10; //пін світлодіода
const int pinFoto = A0; //пін фоторезистора
int bright = 0; //яскравість світлодіода
int value = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  value = analogRead(pinFoto); //читаємо фоторезистор
  Serial.println(value); //відправляємо в комп’ютер
  delay(50); //затримка
  //перетворюємо діапазон значень 0..1024 в 0..255
  bright = map(value,0,1024,0,255);
  analogWrite(pinLed, bright); //формуємо ШІМ сигнал
}
```

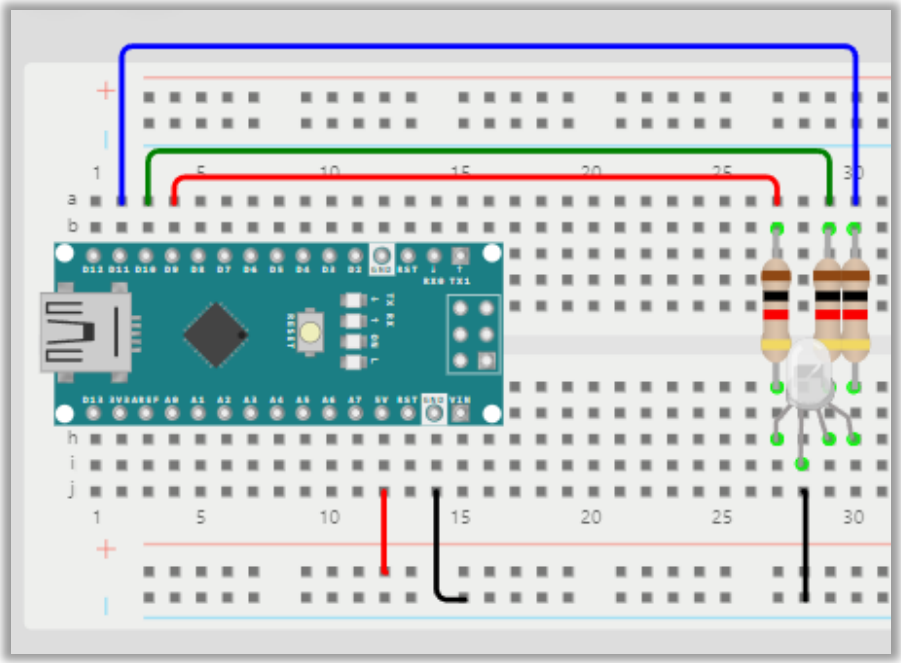
Перш за все необхідно провести налаштування та дізнатися значення аналогового сигналу в умовах хорошої освітленості. Для цього відкриваємо монітор порту та аналізуємо значення, що віддає фоторезистор. Обираємо найбільше значення. Підставляємо значення, яке ми отримали, замість “1024”.

Завдання 8.

На базі схеми підключення з прикладу, розробіть код, в якому світлодіод буде вмикатися при поганому освітленні і гаснути при хорошому рівні освітлення (нічник).

RGB світлодіод

У RGB світлодіода чотири виводи, по аноду на кожен із трьох світлодіодів і один спільний контакт, до якого підключаються всі негативні полюси світлодіодів (катоди). Під час окремого керування кожним із трьох вбудованих світлодіодів - червоного, синього і зеленого кольорів, можна отримати будь-який відтінок.



```
const int pinLedRed = 9;
const int pinLedGreen = 10;
const int pinLedBlue = 11;

void setup() {}

void loop(){
  analogWrite(pinLedRed,255); //запалюємо червоний
  analogWrite(pinLedBlue,0);
  analogWrite(pinLedGreen,0);
  delay(1500);
  analogWrite(pinLedRed,0); //запалюємо синій
  analogWrite(pinLedBlue,255);
  analogWrite(pinLedGreen,0);
  delay(1500);
  analogWrite(pinLedRed,0); //запалюємо зелений
  analogWrite(pinLedBlue,0);
  analogWrite(pinLedGreen,255);
  delay(1500);
  analogWrite(pinLedRed, 255); //запалюємо білий
  analogWrite(pinLedBlue, 255);
  analogWrite(pinLedGreen, 255);
  delay(1500);
}
```

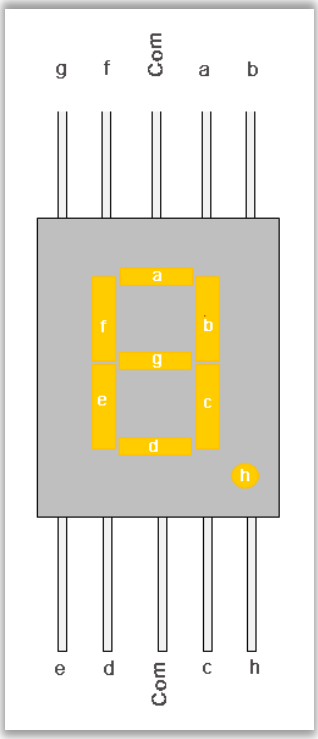
Завдання 9.

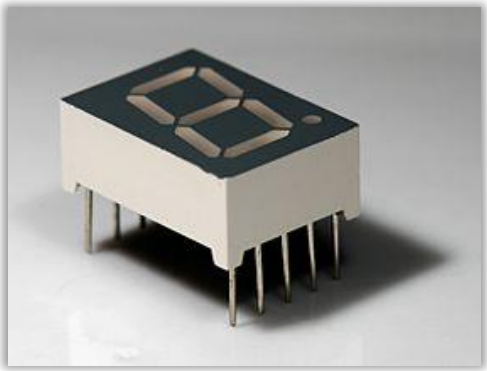
На базі схеми підключення із прикладу, створіть код, в якому відтінки світіння RGB світлодіода будуть встановлюватися в довільному порядку. Використайте оператор random(), який генерує випадкове значення.

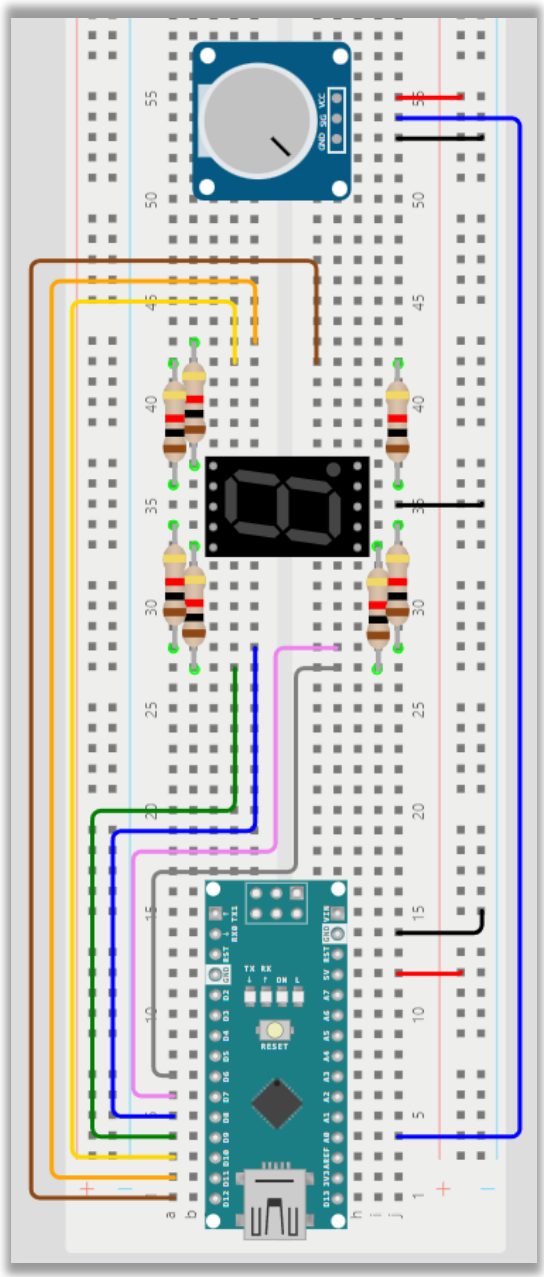
Відображення інформації:
7-сегментний індикатор

Під час конструювання складних чи автономних пристроїв, не завжди є можливість під’єднувати комп’ютер для виводу значень, тому для цього використовують різні дисплеї та індикатори.

Одним із простих варіантів індикації значення 0-9 є 7-сегментний індикатор. Як і RGB світлодіод, він містить в собі декілька світлодіодів. Світлодіоди у ньому розміщені в формі вісімки. Запалюючи відповідні сегменти, можна отримати різні цифри та символи.






GND SIG VCC
RESET
TX RX
ON

```
const int pinLedA = 10;
const int pinLedB = 11;
const int pinLedC = 12;
const int pinLedD = 7;
const int pinLedE = 6;
const int pinLedF = 8;
const int pinLedG = 9;
const int pinPot = A0;
byte segment[10] = {
  0b00111111, //0
  0b00000110, //1 маска вказує, який світлодіод
  0b01011011, //2 запалювати – 1, а який погасити - 0
  0b01001111, //3
  0b01100110, //4 0b0-g-f-e-d-c-b-a це порядок
  0b01101101, //5 світлодіодів у рядку
  0b01111101, //6
  0b00000111, //7
  0b01111111, //8
  0b01101111  //9
};
int number = 0;
byte mask = 0;

void setup() {
  pinMode(pinLedA, OUTPUT);
  pinMode(pinLedB, OUTPUT);
  pinMode(pinLedC, OUTPUT);
  pinMode(pinLedD, OUTPUT);
  pinMode(pinLedE, OUTPUT);
  pinMode(pinLedF, OUTPUT);
  pinMode(pinLedG, OUTPUT);
}
```

```
void loop() {
  //зчитуємо сигнал з потенціометра і
  //перетворюємо його в діапазон 0...9
  number = map(analogRead(pinPot), 0, 1023, 0, 9);
  mask = segment[number];
  showNumber(mask);
}

void showNumber(byte mask){
  digitalWrite(pinLedA, bitRead(mask, 0));
  digitalWrite(pinLedB, bitRead(mask, 1));
  digitalWrite(pinLedC, bitRead(mask, 2));
  digitalWrite(pinLedD, bitRead(mask, 3));
  digitalWrite(pinLedE, bitRead(mask, 4));
  digitalWrite(pinLedF, bitRead(mask, 5));
  digitalWrite(pinLedG, bitRead(mask, 6));
}
```

`digitalWrite(pinLedA, bitRead(mask, 0));`

`pinLedA` – пін світлодіода A

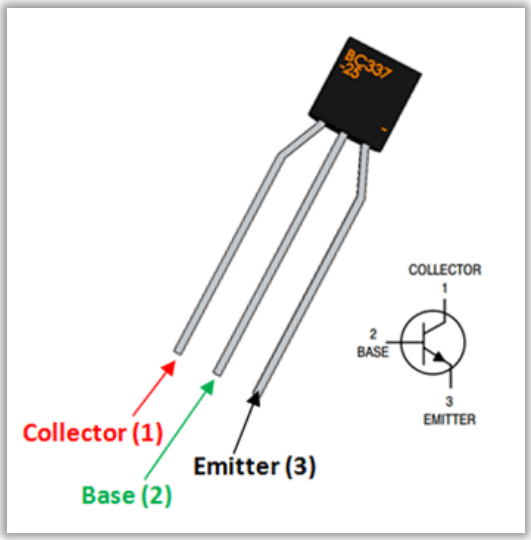
`mask` – поточна маска цифри, яку відображаємо

`bitRead(mask, 0)` – читаємо біт під індексом 0 (з права на ліво) з маски. Якщо результат 1, то запалюємо світлодіод A, інакше гасимо.

Завдання 10.

На основі схеми підключення з прикладу напишіть код, в якому на 7-сегментному індикаторі можна виводити символи: A, b, C, d, E, F, H, J, L, P, U.

Динамічна індикація: підключення декількох 7-сегментних індикаторів



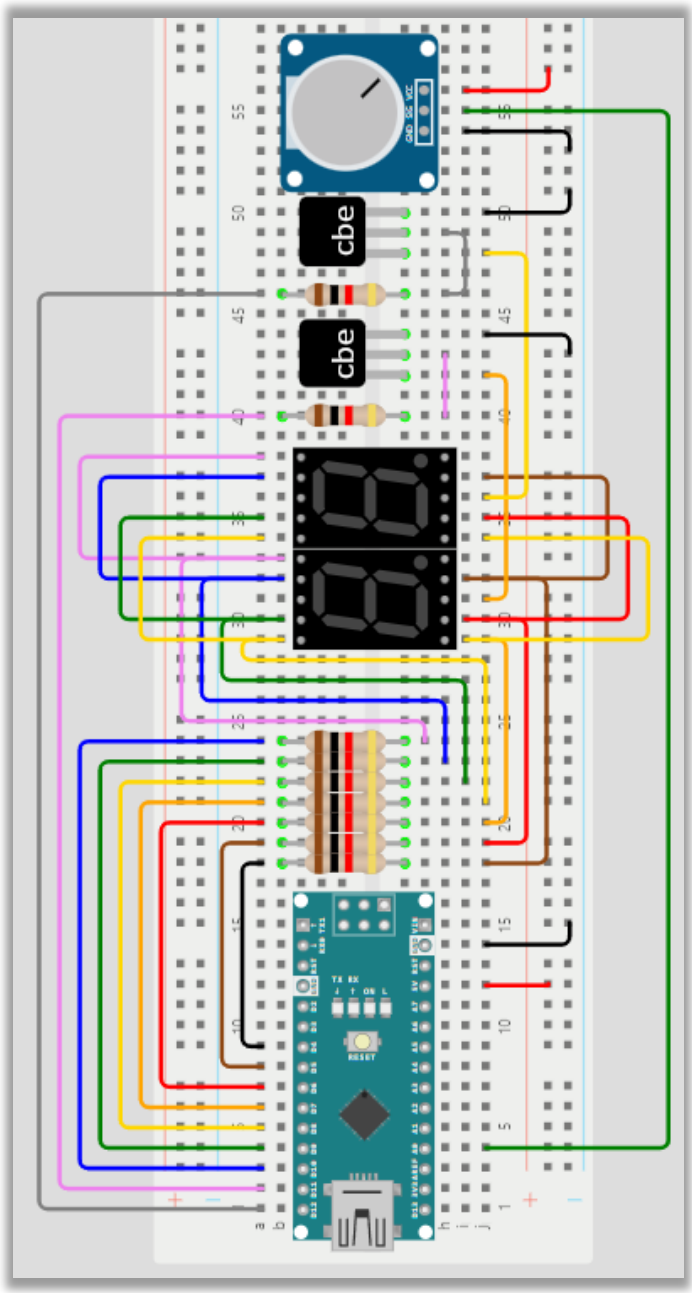
При підключенні декількох 7-сегментних індикаторів до одних і тих же пінів мікроконтролера, виникає проблема з вмиканням різних сегментів на різних індикаторах. Для вирішення цієї проблеми необхідно вмикати спочатку один індикатор з необхідними для нього сегментами, а потім другий. Під час дуже швидкого перемикання, завдяки інерції людського зору, мозок не встигає помітити, що сегменти погасли, та сприймає всю картину. Для керування великими і потужними індикаторами, які споживають великий струм, під час запалювання всіх сегментів необхідно використовувати транзистори в ключовому режимі. Транзистор схожий на електронну кнопку, як тільки на базу транзистора подають струм,

транзистор відкривається та пропускає значний (від десятків міліампер до десятків ампер) струм. Струм вливається в колектор, витікає з емітера. Вище представлена розпіновка n-p-n транзистора BC337.

Складемо схему, яка буде виводити значення з потенціометра у діапазоні від 0 до 99. Для отримання десятків використаємо операцію ділення. Результат зберігається у цілочислений тип. Та операцію modulus, що дає нам число після коми в результаті ділення.

Завдання 11.

Виведіть значення з потенціометра у діапазоні від -9 до 99.


cbe
cbe

```
const int pinSegL = 11;
const int pinSegR = 12;
const int pinLedA = 9;
const int pinLedB = 10;
const int pinLedC = 4;
const int pinLedD = 5;
const int pinLedE = 6;
const int pinLedF = 8;
const int pinLedG = 7;
const int pinPot = A0;
byte segment[10] = {
  0b00111111, //0 0b0-g-f-e-d-c-b-a
  0b00000110, //1
  0b01011011, //2
  0b01001111, //3
  0b01100110, //4
  0b01101101, //5
  0b01111101, //6
  0b00000111, //7
  0b01111111, //8
  0b01101111  //9
};
int number = 0;
byte maskL = 0;
byte maskR = 0;
void setup() {
  pinMode(pinSegL, OUTPUT);
  pinMode(pinSegR, OUTPUT);
  pinMode(pinLedA, OUTPUT);
  pinMode(pinLedB, OUTPUT);
  pinMode(pinLedC, OUTPUT);
```

```
  pinMode(pinLedD, OUTPUT);
  pinMode(pinLedE, OUTPUT);
  pinMode(pinLedF, OUTPUT);
  pinMode(pinLedG, OUTPUT);
}

void loop() {
  number = map(analogRead(pinPot), 0, 1023, 0, 99);
  maskL = segment[number / 10]; // division
  maskR = segment[number % 10]; // modulus

  showNumber(0b0); // вимикаємо всі світлодіоди
  digitalWrite(pinSegL, HIGH);// вмикаємо лівий
  digitalWrite(pinSegR, LOW); // вимикаємо правий
  showNumber(maskL); // запалюємо світлодіоди
  delay(5); // 5 мілісекунд світимо
  showNumber(0b0); // вимикаємо всі світлодіоди
  digitalWrite(pinSegL, LOW);// вимикаємо лівий
  digitalWrite(pinSegR, HIGH);// вмикаємо правий
  showNumber(maskR); // запалюємо світлодіоди
  delay(5); // 5 мілісекунд світимо
}
void showNumber(byte mask){
  digitalWrite(pinLedA, bitRead(mask, 0));
  digitalWrite(pinLedB, bitRead(mask, 1));
  digitalWrite(pinLedC, bitRead(mask, 2));
  digitalWrite(pinLedD, bitRead(mask, 3));
  digitalWrite(pinLedE, bitRead(mask, 4));
  digitalWrite(pinLedF, bitRead(mask, 5));
  digitalWrite(pinLedG, bitRead(mask, 6));
}
```

Дисплей LCD1602

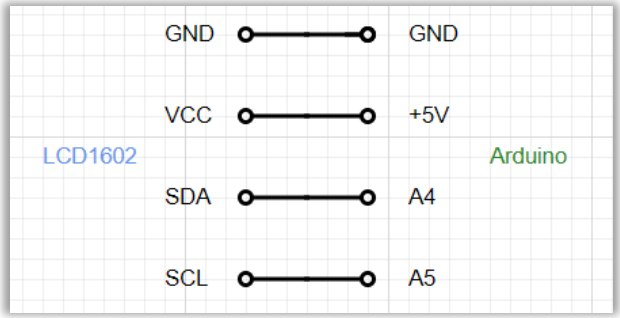
LCD дисплей має значно більший функціонал ніж 7-сегментний індикатор. Екран має два рядки по 16 комірок у кожному. Кожна комірка складається із пікселів

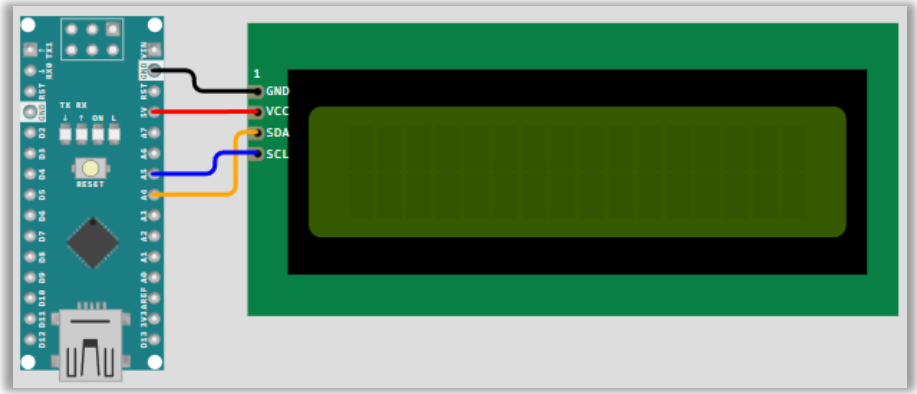
розміром 5*7, всього - 1120 точок. LCD дисплей укомплектовано платою розширення для підключення по **I²C** шині (Inter-Integrated Circuit). Шина використовує дві двонапрямлені лінії: послідовна лінія даних (SDA, *Serial DAta*) і послідовна лінія тактування (SCL, *Serial CLock*).

У кожного пристрою, що працює по I^2C шині, є унікальна адреса, за якою Arduino звертається до нього. До одної шини I2C можуть бути підключені до 128 пристроїв. Для роботи з модулем потрібно встановити спеціальну бібліотеку “LiquidCrystal_I2C_V112”. Підключення дисплею до Arduino NANO виконується з допомогою 4 проводів “мама-тато”.

Далі потрібно визначити адресу саме вашого I2C пристрою. Найпростішим способом є послідовний перебір всіх 128 адрес. Якщо для існуючої адреси передача пройшла успішно, тоді функція передачі повертає число “0”.

Частіше за все дисплеї з синім екраном мають номер 0x27, з зеленим екраном 0x3F.






```
#include <Wire.h>     //під'єднуємо бібліотеку Wire
byte error, address; //створюємо дві змінні

void setup() {
  Wire.begin(); //ініціалізація інтерфейсу I2C
                //на пінах A4, A5
  Serial.begin(9600);
}

void loop() {
  //послідовній перебір 128 адрес
  for (address = 0; address < 127; address++ ){
    //з'єднуємось з пристроєм по адресі
    Wire.beginTransmission(address);
    //закінчуємо зв'язок та читаємо результат
    error = Wire.endTransmission();
    // якщо 0 - помилок нема, зв'язок відбувся
    if(error == 0){
      Serial.print("Знайдено: ");
      Serial.print(address, DEC);
```

```
      //Виводимо адресу у шістнадцятковій системі
      Serial.print(" (0x");
      Serial.print(address, HEX);
      Serial.println(")");
    }
  }
  delay(1000);
```

Виведемо на екран кількість секунд з моменту запуску Arduino:

```
//підключення бібліотеки для роботи з LCD1602
#include <LiquidCrystal_I2C.h>
//налаштовуємо адресу дисплею, кількість символів в
//рядку, і кількість рядків
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  lcd.init(); //ініціалізація інтерфейсу
  lcd.backlight(); //вмикання підсвітки дисплея
}
void loop() {
  lcd.clear(); //очищення дисплея
  lcd.setCursor(0, 0); //курсор у верхній лівий куток
  //виводимо кількість секунд з моменту запуску Arduino
  lcd.print(millis()/1000);
  delay(1000); //пауза 1 секунда між оновленням екрану
}
```

Створимо новий символ. Для цього необхідно створити масив із вісьмох значень, в якому вказати маску символу. Де 1 - це піксель, який необхідно засвітити. Давайте створимо свій знак смайлика:

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

byte smile[8]={
  0b00000,
  0b01010,
  0b01010,
  0b00000,
  0b10001,
  0b01110
};
void setup() {
  lcd.init(); //ініціалізація
  lcd.backlight();
  lcd.createChar(1, smile); //створюємо символ
}
void loop() {
  lcd.clear();
  lcd.setCursor(4,0); //курсор в 4 комірку 0 рядка
  lcd.print("life is"); //виводимо текст
  lcd.setCursor(2,1); //курсор в 2 комірку 1 рядка
  lcd.write(1); //виводимо смайл
  lcd.print(" Beautiful "); //виводимо текст
  lcd.write(1); //виводимо смайл
  delay(1000); //затримка між циклами
}
```

Створимо анімацію процесу завантаження. Для цього нам потрібно створити декілька символів, які будемо перемальовувати в одній комірці:

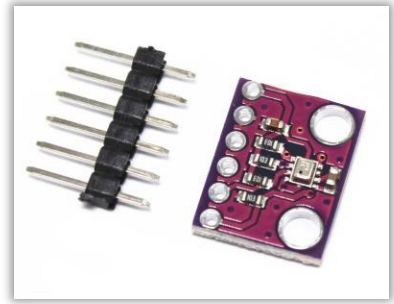
```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

byte bar1[8] = {0b10000,0b10000,0b10000,0b10000,
                0b10000,0b10000,0b10000,0b10000};
byte bar2[8] = {0b11000,0b11000,0b11000,0b11000,
                0b11000,0b11000,0b11000,0b11000};
byte bar3[8] = {0b11100,0b11100,0b11100,0b11100,
                0b11100,0b11100,0b11100,0b11100};
byte bar4[8] = {0b11110,0b11110,0b11110,0b11110,
                0b11110,0b11110,0b11110,0b11110};
byte bar5[8] = {0b11111,0b11111,0b11111,0b11111,
                0b11111,0b11111,0b11111,0b11111};
int progress = 0;
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.createChar(1, bar1);
  lcd.createChar(2, bar2);
  lcd.createChar(3, bar3);
  lcd.createChar(4, bar4);
  lcd.createChar(5, bar5);
}
void loop() {
  lcd.clear();
  //переносимо курсор в 2 комірку 0 рядка
  lcd.setCursor(2,0);
  lcd.print("Loading 0%");
```

```
  delay(250);
  //обираємо комірку у другому рядку
  for(int i=0; i < 16; i++){
    for(int j = 1; j < 6; j++){ //обираємо символ
      lcd.setCursor(2,0);
      lcd.print("Loading ");
      progress = (i*5+j) / 0.8;
      //0.8 – 16 комірок по 5 пікселів = 80
      //отримаємо ціну одного відсотка 80/100=0.8
      lcd.print(progress);
      lcd.print("%");
      //курсор на i-ту комірку 1 рядка
      lcd.setCursor(i,1);
      lcd.write(j); //виводимо символ
      delay(250);
    }
  }
  delay(1000);
}
```

Завдання 12.

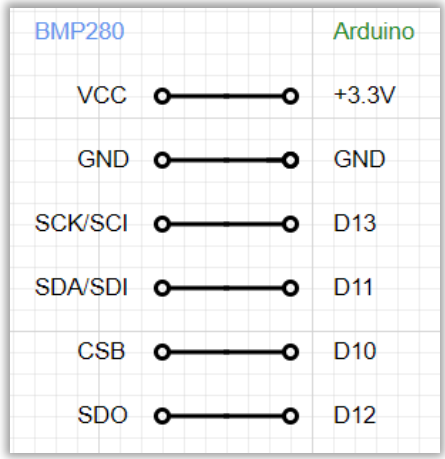
Виведіть значення з потенціометра від 0 до 100%. У рядку 0 надпис “Volume %”. У рядку 1-прогрес бар.

Метеостанція - цифровий барометр BMP280



Для створення метеостанції нам потрібен модуль BMP280 і дві бібліотеки “Adafruit_BMP280_Library” і “Adafruit_Sensormaster”.

Цей модуль вимірює тиск. Також в модуль вбудовано термосенсор, який показує температуру сенсора для розрахунку тиску, його не можна використовувати як термометр повітря. Він потрібен для розрахунку тиску. Також, можна виміряти висоту над рівнем моря, якщо знати тиск на рівні моря у даний момент. Наприклад, виміряємо тиск на першому поверсі будинка, та внесемо



це значення у програму. Підіймемось на четвертий поверх – користуючись функціями барометра, отримаємо висоту на яку ми піднялись.

```
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
Adafruit_BMP280 bme(10); //налаштування на d10 пін
//Задаємо тиск на рівні моря у Паскалях
#define SEALEVELPRESSURE_HPA (1013.25)
void setup()
{
  lcd.init(); //ініціалізація інтерфейсу I2C
  lcd.backlight(); //вмикання підсвітки дисплея
  bme.begin(); //інійіалізація інтерфейсу SPI
}
void loop(){
  lcd.clear(); //очищення дисплея
  lcd.setCursor(0, 0); //курсор на перший рядок
  lcd.print("Press = "); //виводимо текст
  lcd.print(bme.readPressure()/133.3224);
  //виводимо значення в мм .рт. ст.
  lcd.setCursor(0, 1); //курсор на другий рядок
  lcd.print("Alt = "); //виводимо текст
  lcd.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  lcd.println(" m");
  delay(2000); //затримка між замірами
}
```

Модуль під’єднується по SPI шині до виводу із напругою 3.3 вольта, що знаходиться біля D13. Враховуйте, що модуль чутливий до живлення (3.3 в), тож у випадку підключення до шини із живленням 5 вольт, може вийти з ладу.

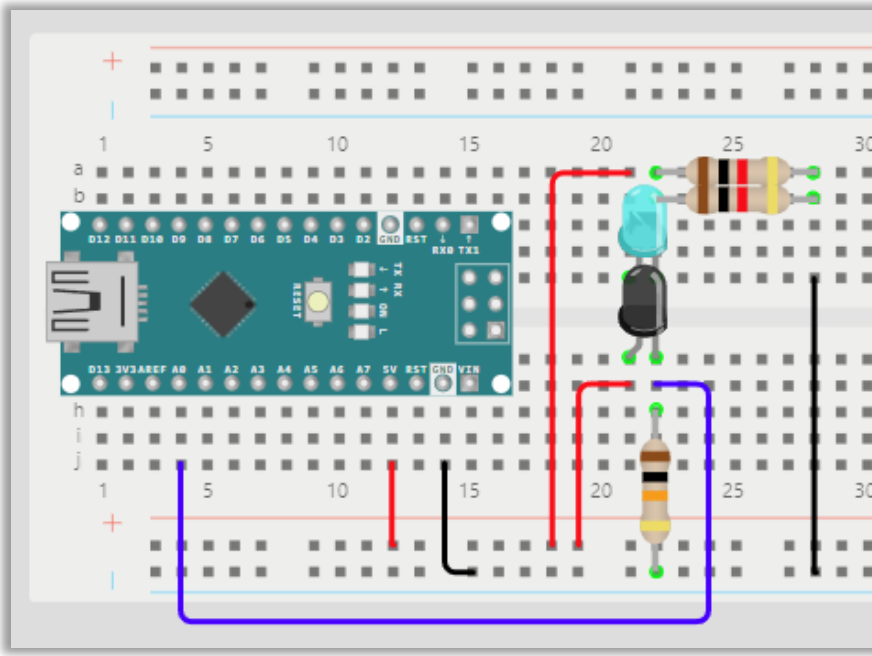
Завдання 13.

Додайте кнопку, яка буде встановлювати поточне значення тиску, як тиск на рівні моря.

Інфрачервоний відбивач

Датчик TCRT5000 випускає інфрачервоний сигнал (довжина хвилі 950 нано метрів) і потім ловить відображення цього інфрачервоного сигналу від поверхні. Залежно від типу і кольору поверхні змінюється інтенсивність відбитого сигналу. З цієї інтенсивності можна приблизно судити про відтінки кольору поверхні, також датчик може використовуватися як альтернатива датчику Холла, для визначення швидкості обертання. Складається з світлодіода який випускає світло в інфрачервоному діапазоні і фототранзистор. Робоча дистанція 12мм.

```
int inPin = A0; //пін для підключення емітера
int val = 0;    //значення датчика
void setup() {
  Serial.begin(9600);
}
void loop(){
  val = analogRead(inPin); // читаємо значення
  Serial.println(val); // виводимо отримане значення
  delay(100);
}
```



Завдання 14.

Напишіть програму, яка буде перемикати світлодіод, якщо палець на сенсорі більше 3-х секунд.