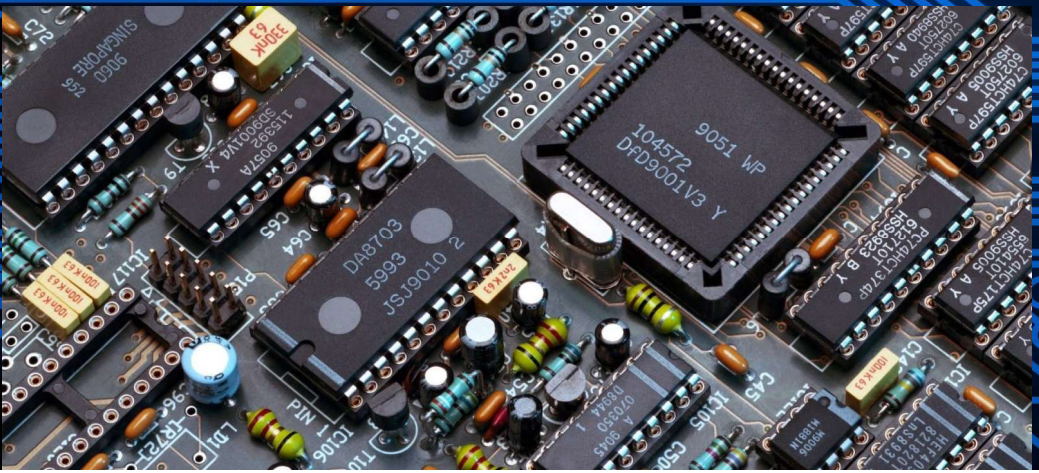
Безпека інтернет речей

Лекція №4



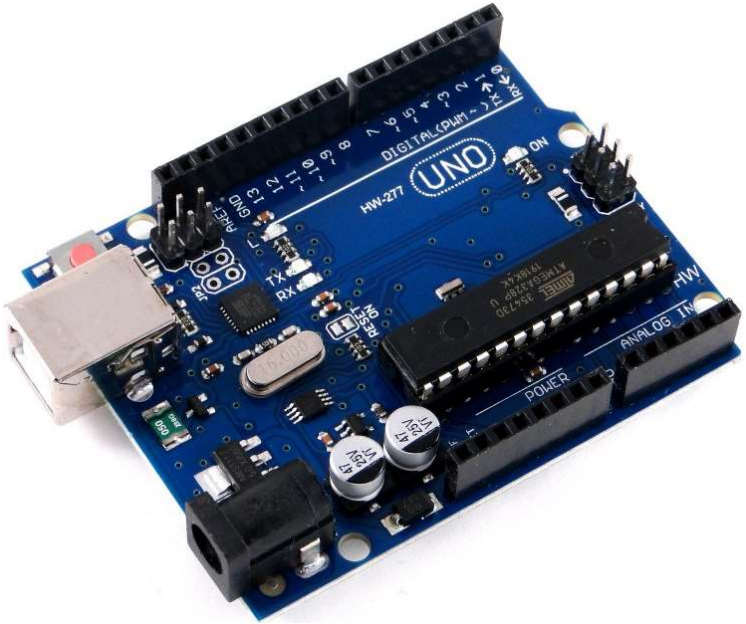
Лекцію проводить:

доц. Лимаренко Вячеслав Володимирович

к.т. 066-0708586

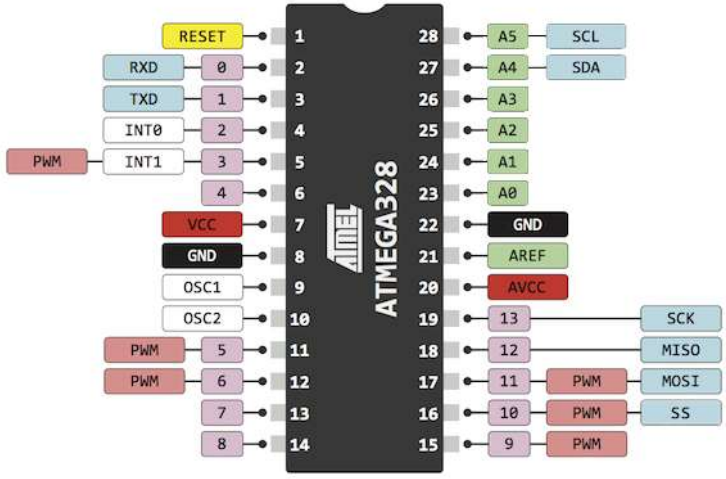
Налагоджувальна плата Arduino Uno R3 з МК ATmega328

Arduino Uno – це пристрій на основі мікроконтролера ATmega328. До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШІМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) та кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю. Uno як перетворювач інтерфейсів USB-UART використовує мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI.



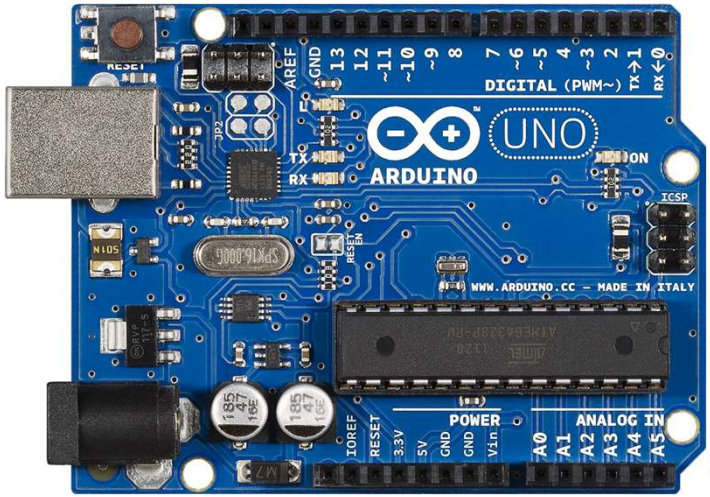
Характеристики:

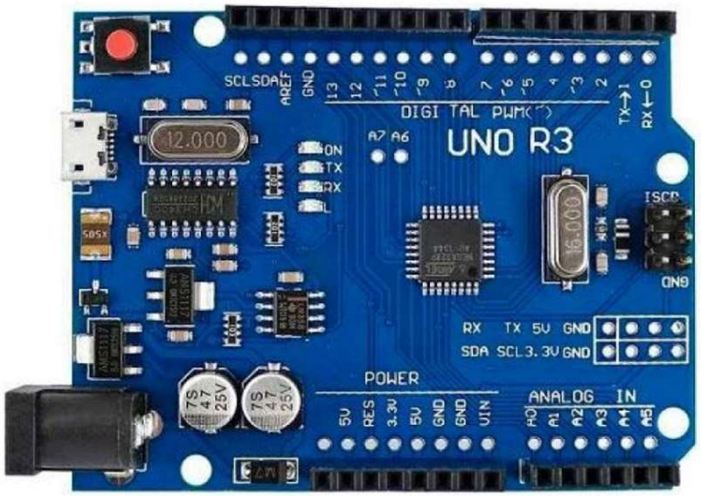
- мікроконтролер ATmega328;
- робоча напруга 5В;
- напруга живлення (рекомендована) 7-12В;
- напруга живлення (гранична) 6-20В;
- цифрові входи/виходи 14 (з них 6 можуть використовуватися як ШІМ-виходи);
- аналогові входи 6;
- максимальний струм одного виводу 40 мА;
- максимальний вихідний струм виведення 3.3V 50 мА;
- Flash-пам'ять 32 КБ (ATmega328), з яких 0.5 КБ використовуються завантажувачем;
- SRAM 2 КБ (ATmega328);
- EEPROM 1 КБ (ATmega328);
- тактова частота 16 МГц.

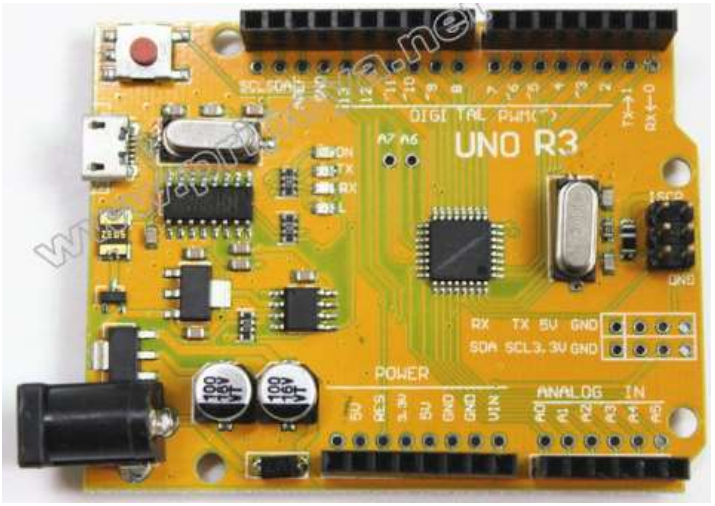


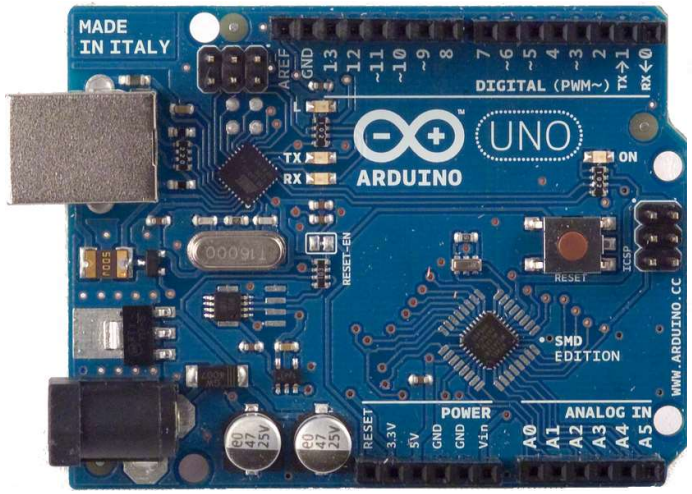
ARDUINO
OPEN-SOURCE
COMMUNITY

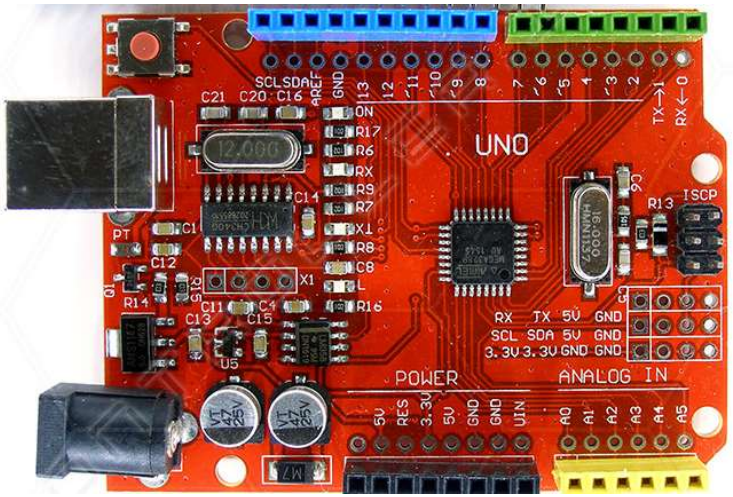
Все це Arduino Uno R3

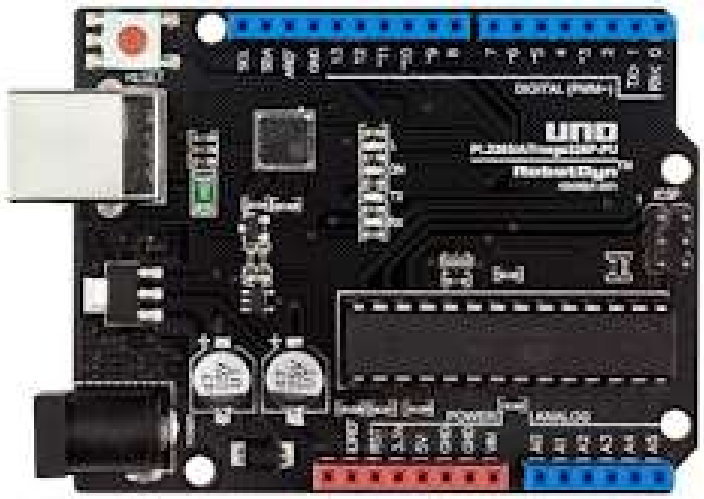




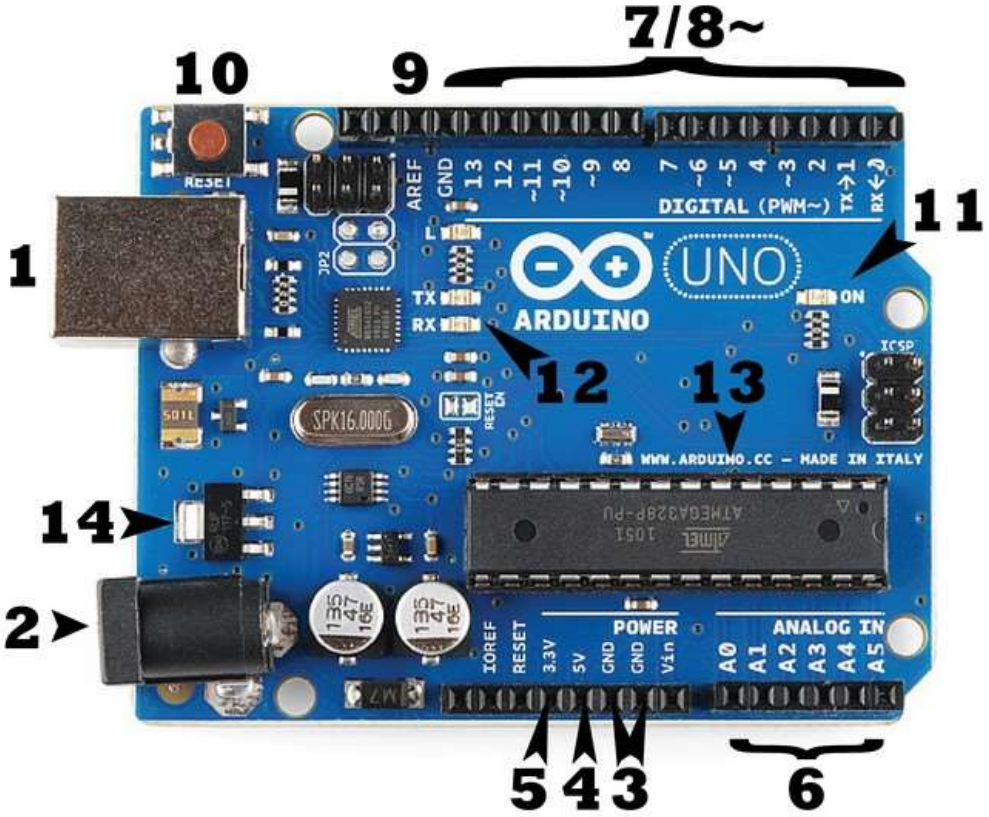






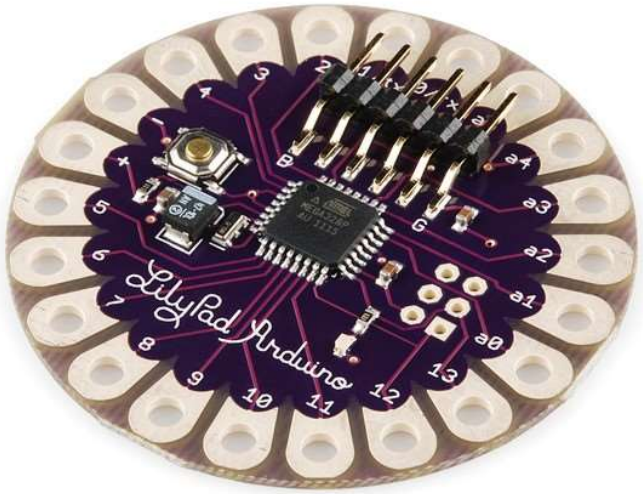


Склад налагоджувальної плати Arduino Uno R3 з МК ATmega328

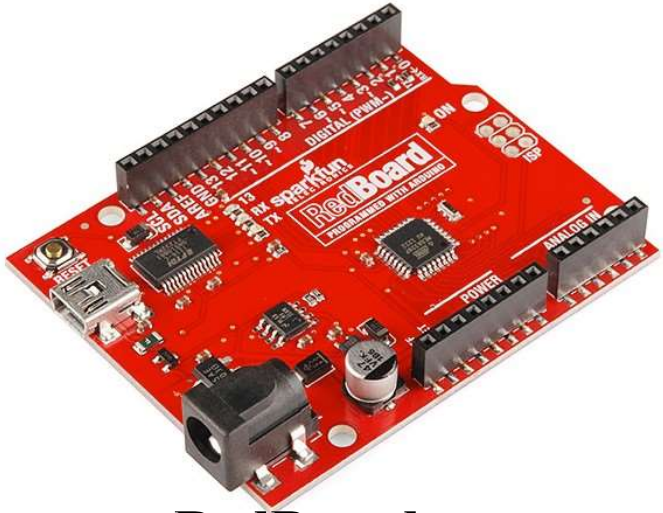


1 Роз'єм USB;
2 Гніздо для зовнішнього джерела живлення;
3 Роз'єм (пін) GND (земля);
4-5 Роз'єми (піни) 5V та 3.3V;
6 Рознімання (піни) Analog (аналоговий сигнал);
7 Рознімання (піни) Digital (цифровий сигнал);
8 Роз'єм (пін) PWM (режим ШІМ-модуляції);
9 Роз'єм (пін) AREF (встановлення максимального значення напруги на аналогових входах (від 0 до 5 вольт));
10 Кнопка скидання (Reset Button);
11 Світлодіод «on»;
12 Світлодіоди TX (transmit, передача) та RX (receive, прийом);
13 Мікроконтролер (ATmega від компанії ATMEL);
14 Регулятор напруги.

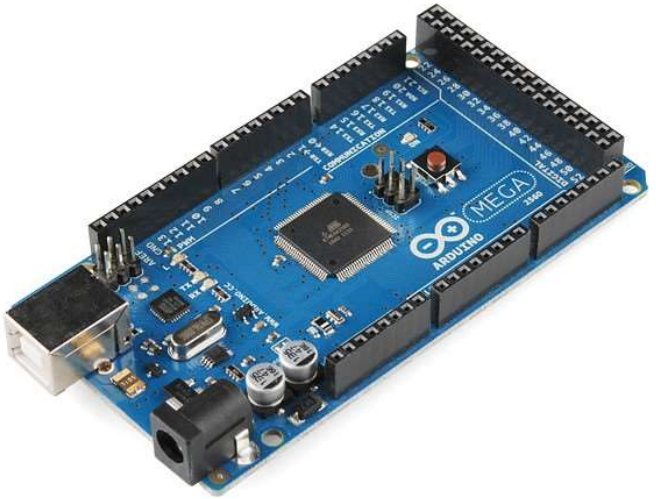
Види плат Arduino



LilyPad Arduino

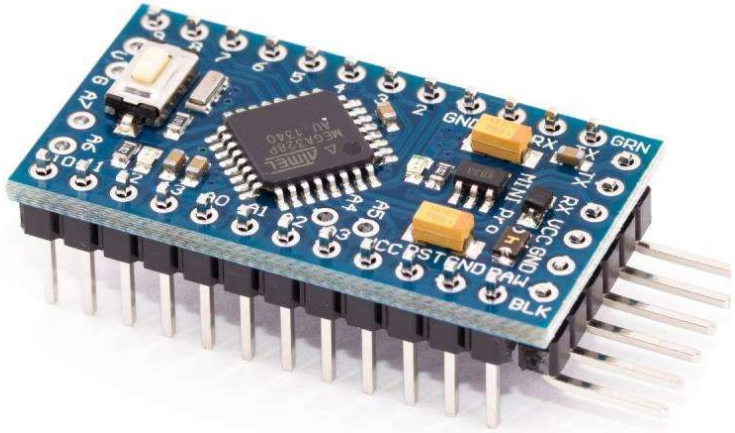


RedBoard



Arduino Mega (R3)

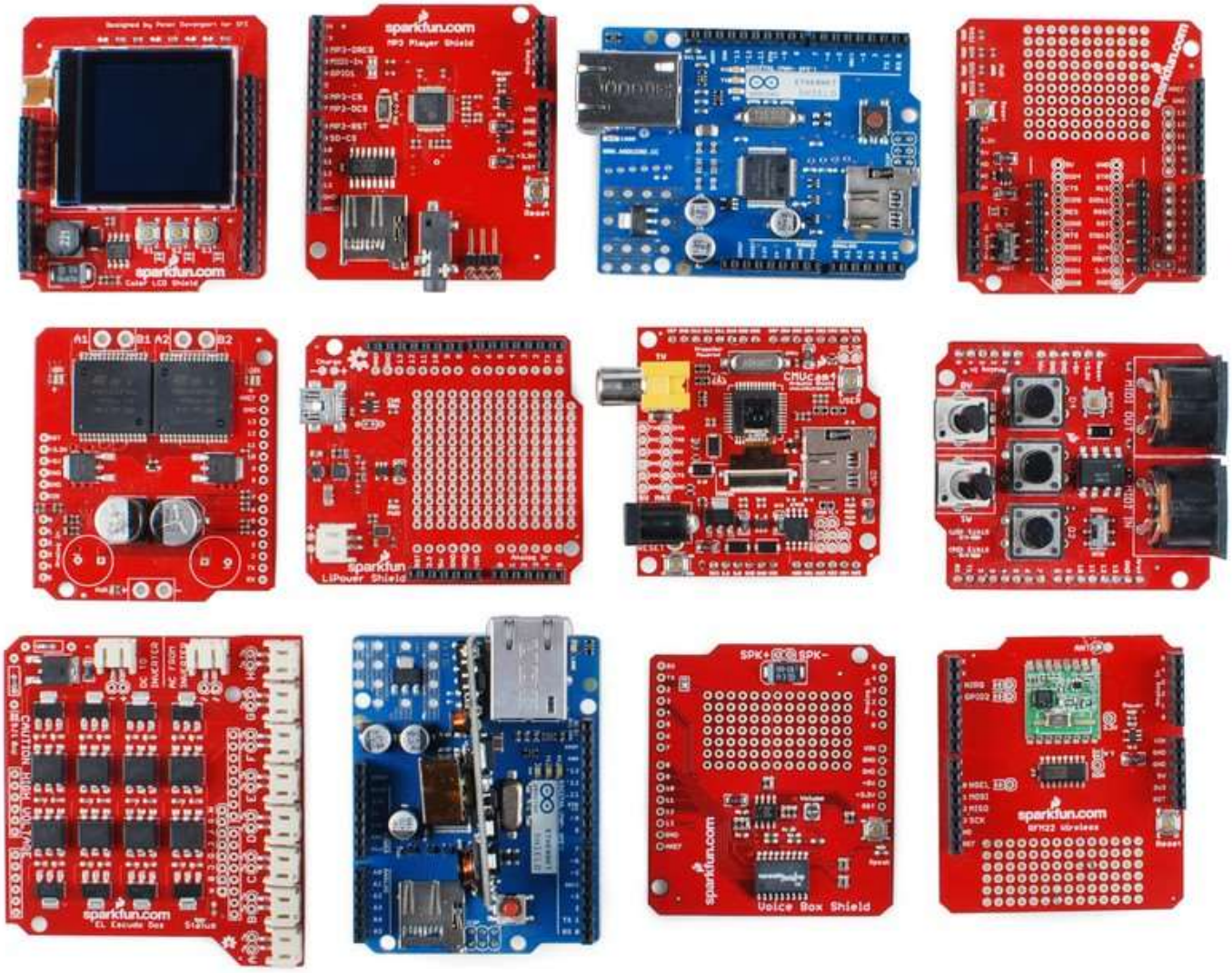
Arduino Leonardo

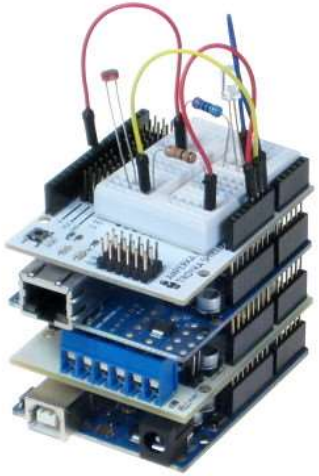


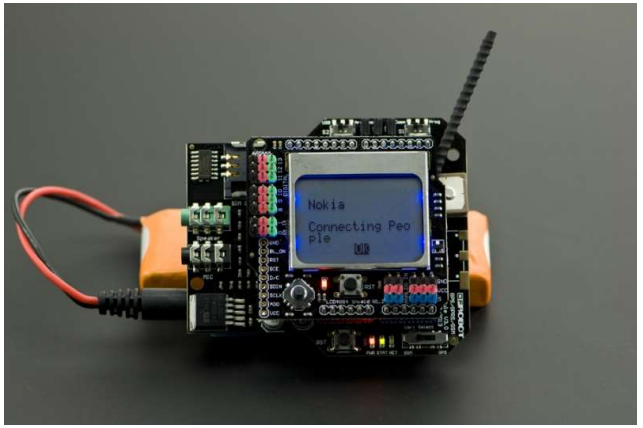
Arduino Pro Mini

Давачі (сенсори)

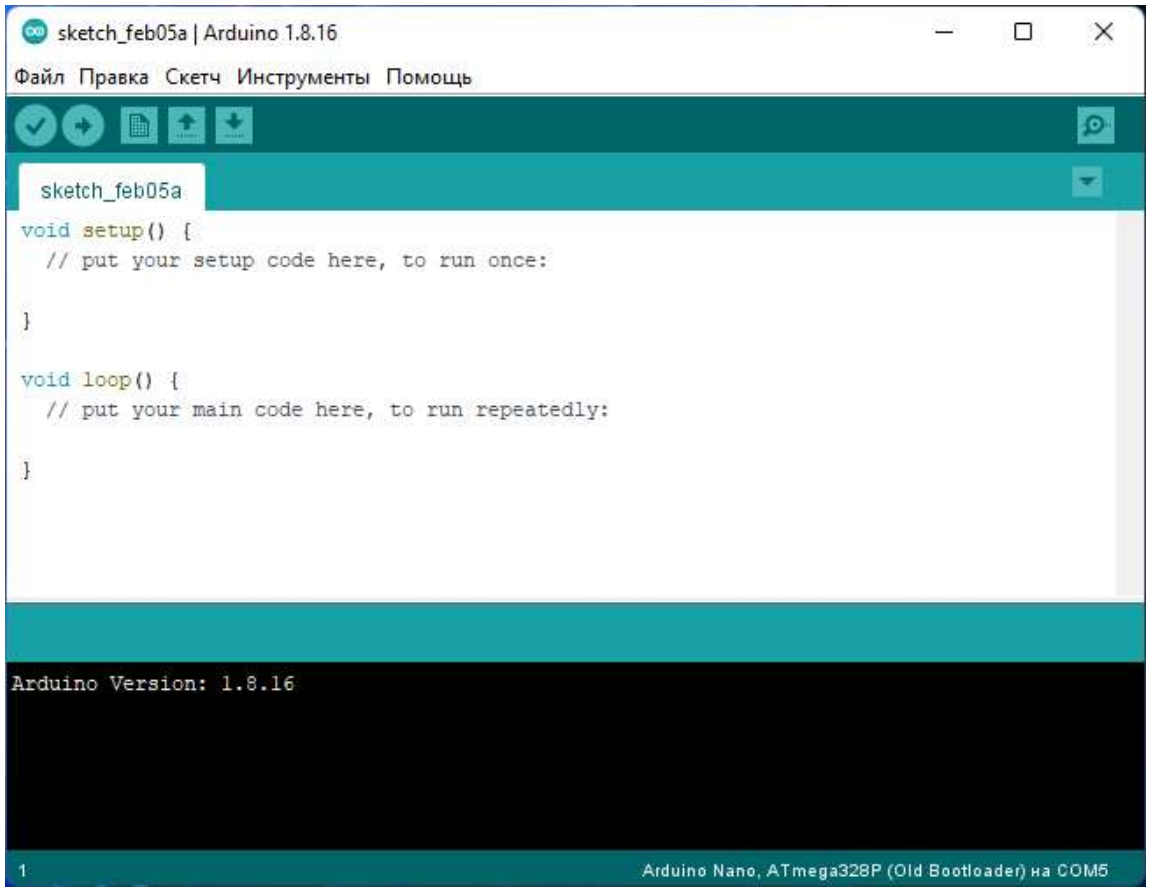
Шилди (Shields) для Arduino







Arduino IDE



Arduino IDE базується на мові Wiring, яка, в свою чергу, – на Processing, який, – на C/C++. Мова скомпонована з бібліотекою AVR Libc і дозволяє використовувати будь-які її функції.

Як і в інших С-подібних мовах програмування є ряд правил написання коду. Ось базові з них:

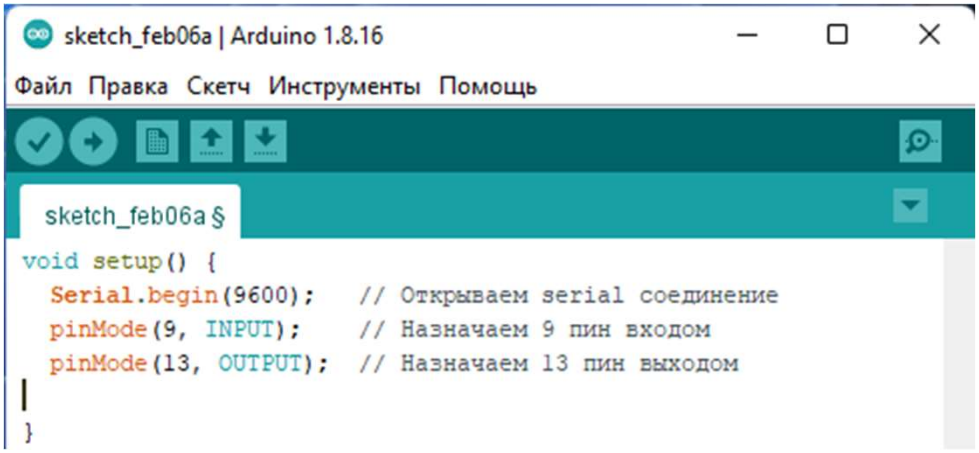
- після кожної інструкції необхідно ставити знак крапки з комою (;);
- перед оголошенням функції необхідно вказати тип даних, що повертається функцією або void, якщо функція не повертає значення;
- Також необхідно вказувати тип даних перед оголошенням змінної;
- Коментарі позначаються: // *рядковий* та /* *блоковий* *блоковий* */

Структура програм

Всі прошивки Arduino повинні містити мінімум 2 функції. Це *setup()* та *loop()*.

Функція *setup()*

Функція *setup()* виконується на самому початку і лише 1 раз одразу після увімкнення або перезавантаження вашого пристрою. Зазвичай у цій функції декларують режими пінів, відкривають необхідні протоколи зв'язку, встановлюють з'єднання з додатковими модулями та настроюють підключені бібліотеки. Якщо для вашої прошивки нічого подібного робити не потрібно, то функція все одно має бути оголошена. Ось стандартний приклад функції *setup()*:



Входи та виходи Arduino Uno R3 з МК ATmega328

З використанням функцій *pinMode()*, *digitalWrite()* і *digitalRead()* кожен із 14 цифрових виходів може працювати як вхід або вихід. Рівень напруги на виходах обмежений 5В. Максимальний струм, який може давати або споживати один вихід, становить 40 мА. Всі виходи пов'язані з внутрішніми резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім цього, деякі виходи плати можуть виконувати додаткові функції:

– послідовний інтерфейс: виходи 0 (RX) та 1 (TX). Використовуються для отримання (RX) та передачі (TX) даних за послідовним інтерфейсом. Ці виходи з'єднані з відповідними виходами мікросхеми ATmega16U2, що виконує роль USB-UART-перетворювача;

– зовнішні переривання: виходи 2 і 3. Можуть бути джерелами переривань, що виникають при фронті, спаді або низькому рівні сигналу цих виходів;

– ШІМ: виходи 3, 5, 6, 9, 10 та 11. За допомогою функції *analogWrite()* можуть виводити 8-бітові аналогові значення у вигляді ШІМ-сигналу;

– інтерфейс SPI: виходи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Із застосуванням бібліотеки SPI дані виходи можуть здійснювати зв'язок з інтерфейсом SPI;

– світлодіод: 13. Вбудований світлодіод, приєднаний до виходу 13. При надсиланні значення *HIGH* світлодіод вмикається, при відправленні *LOW* – вимикається.

Arduino Uno має 6 аналогових входів (A0 – A5), кожен з яких може представити аналогову напругу у вигляді 10-бітного числа (1024 різних значення). За замовчуванням, вимірювання напруги здійснюється відносно діапазону від 0 до 5 В. Однак верхню межу цього діапазону можна змінити, використовуючи виведення AREF і функцію *analogReference()*. Крім цього, деякі аналогові входи мають додаткові функції:

– TWI: вихід A4 або SDA та вихід A5 або SCL. З використанням бібліотеки *Wire* ці виходи можуть здійснювати зв'язок по інтерфейсу TWI.

Крім перерахованих на платі, існує ще кілька виходів:

– AREF. Опорна напруга аналогових входів. Може використовуватися функцією *analogReference()*;

– Reset. Формування низького рівня (LOW) на цьому виході призведе до перезавантаження мікроконтролера. Зазвичай цей вихід служить для функціонування кнопки скидання на платах розширення.

Структура програм

Функція *loop()*

Функція *loop()* виконується після функції *setup()*. Loop у перекладі з англійської означає «петля». Це говорить про те, що функція зациклена, тобто буде виконуватися знову і знову. Наприклад мікроконтролер ATmega328, який встановлений у більшості плат Arduino, виконуватиме функцію loop близько 10 000 разів на секунду (якщо не використовуються затримки та складні обчислення).

Приклад:

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // вмикаємо LED (HIGH - це високий рівень напруги (5В))
  delay(1000);                       // чекаємо 1 с
  digitalWrite(LED_BUILTIN, LOW);    // вимикаємо LED (LOW - це низький рівень напруги (0В))
  delay(1000);                       // чекаємо 1 с
}
```

Гарний довідник по командам Arduino знаходиться за адресою: https://doc.arduino.ua/ru/prog/

GPIO Arduino

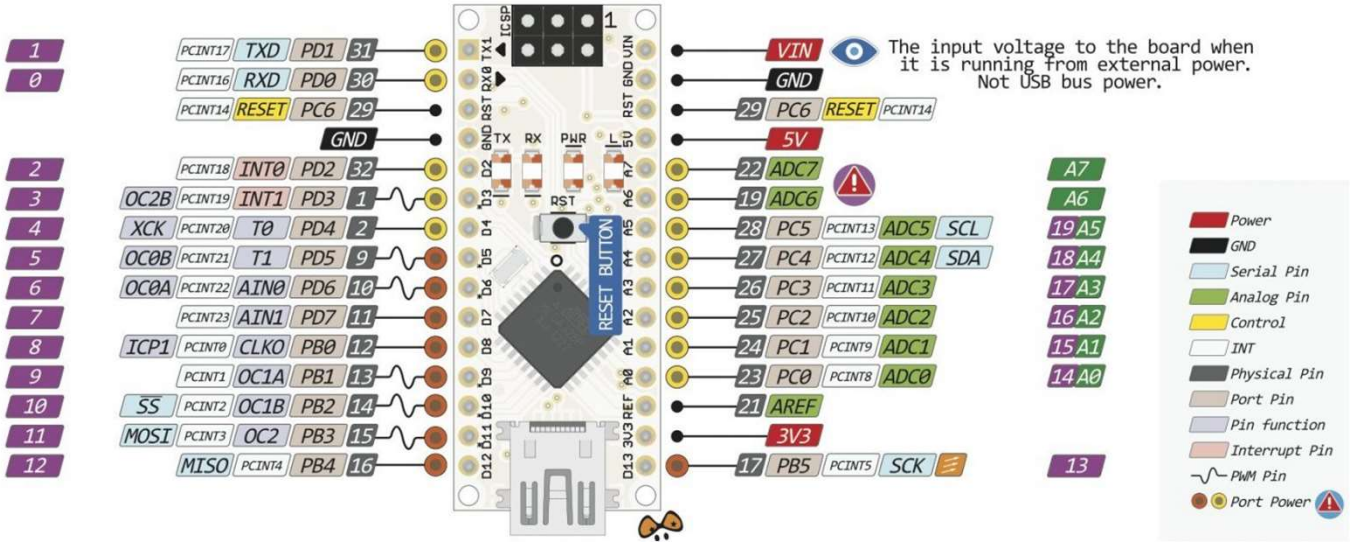
GPIO (з англ. General Purpose Input-Output, входи-виходи загального призначення) – на платі підписані як D0-D13 та A0-A5. «Офіційно» вони називаються PD*, PB* і PC* (замість зірочки – цифра). PD/PB/PC – об'єднання виходів в порти по кілька штук (не більше 8). GPIO мають кілька режимів роботи: вхід (INPUT), вихід (OUTPUT) та вхід з підтяжкою до живлення, вбудованим у МК резистором на 20 кОм (INPUT_PULLUP).

Всі GPIO піни в режимі входу можуть прийняти сигнал з напругою від 0 до 5 вольт

Негативна напруга або напруга, що перевищує 5.5 В, призведе до виходу піна або навіть самого МК з ладу.

Напруга 0-2.5 В вважається низьким рівнем (LOW), 2.5-5.5 – високим рівнем (HIGH). Якщо GPIO нікуди не підключено, тобто, «висить у повітрі», він приймає випадкову напругу, що виникає через різноманітні наведення.

GPIO у режимі виходу (OUTPUT) є транзисторними виходами мікроконтролера та можуть видати напругу 0 або VCC (напруга живлення МК). Максимальний струм, який можна зняти з GPIO виходу Arduino - 40 мА. Максимальний струм з усіх пінів обмежений 200 мА.



Аналогові і цифрові GPIO Arduino. Режими роботи GPIO

Цифровий пін може перебувати у двох станах, вхід та вихід. У режимі входу пін може зчитувати напругу від 0 до напруги живлення МК, а режимі виходу – видавати таку ж напругу. Режим роботи вибирається за допомогою функції *pinMode(pin, mode)*, де *pin* це номер піна, а *mode* це режим:

- mode – режим роботи;
- INPUT – вхід;
- OUTPUT – вихід;
- INPUT_PULLUP – підтягнутий до живлення вхід.

За замовчуванням всі піни налаштовані як входи (INPUT).

```
pinMode(i, OUTPUT); // призначаємо пін виходом
```

Виведення цифрового сигналу

Цифровий пін в режимі виходу (OUTPUT) може генерувати цифровий сигнал, тобто, видавати напругу. Так як поняття «цифровий» зазвичай пов'язане з двома станами, 0 і 1, цифровий пін може видати 0 або 1, точніше сигнал низького або високого рівня. Сигнал низького рівня це 0 Вольт, грубо кажучи в цьому стані пін підключається до GND мікроконтролера. Сигнал високого рівня підключає пін до VCC мікроконтролера, тобто живлення.

Для подачі цифрового сигналу використовується функція *digitalWrite(pin, value):*

- pin – цифровий пін МК, підписаний на платі як D;
- value – рівень сигналу: HIGH високий, LOW низький. Також можна використовувати цифри 0 та 1.

Для роботи в режимі виходу пін повинен бути переведений у стан OUTPUT за допомогою *pinMode()*, інакше він не видасть напругу, яку можна виміряти або чимось керувати.

Приклад, в якому піни ініціалізуються як виходи, і на них подається сигнал:

```
void setup() {
pinMode(10, OUTPUT); // D10 як вихід
pinMode(A3, OUTPUT); // A3 як вихід
pinMode(19, OUTPUT); // A5 як вихід
digitalWrite(10, HIGH); // високий сигнал на D10
digitalWrite(A3, 1); // високий сигнал на A3
digitalWrite(19, 1); // високий сигнал на A5
}
void loop() {}
```

Пін, налаштований як OUTPUT, має сигнал LOW за замовчуванням

Читання цифрового сигналу

Цифровий пін може «вимірювати» напругу, але повідомити він може тільки про її відсутність (сигнал низького рівня, LOW) або наявність (сигнал високого рівня, HIGH). Мікроконтролер спокійно може працювати з логічними пристроями, які шлють йому високий сигнал з напругою 3.3V, він такий сигнал прийме як HIGH.

Не можна подавати на цифровий пін напругу вище за напругу живлення мікроконтролера.

Для читання рівня сигналу на піні використовується функція *digitalRead(pin)*, де пін – номер піна згідно з підписом на платі. Це піни, підписані як D, а також піни A0-A5 у Arduino Nano/Uno/Pro Mini. Ця функція повертає 0, якщо сигнал низького рівня, і 1 – якщо високого.

Простий приклад роботи з пінами в режимі читання:

```
void setup() {
Serial.begin(9600); //налаштування швидкості роботи пору
}
void loop() {
Serial.println(digitalRead(5)); //читаємо пін D5
}
```

Цей код виводитиме в порт сигнал на піні D5. Якщо підключити його дротом до VCC – отримаємо 1, якщо GND – отримаємо 0.

Читання аналогового сигналу

«Аналогові» піни можуть приймати напругу від 0 (GND) до опорної напруги і перетворювати її на цифрове значення, просто в якісь умовні одиниці. АЦП має розрядність в 10 біт, тобто, ми отримуємо виміряну напругу у вигляді числа від 0 до 1023. Функція, яка оцифровує напругу, називається *analogRead(pin)*, дана функція приймає як аргумент номер аналогового піна і повертає отримане значення. Сам пін повинен бути налаштований як INPUT (вхід). Пін вказується, як «аналоговий»:

- просто номер А-піна (наприклад, 0);
- номер з літерою А (наприклад, А0);
- порядковим номером GPIO: А0 – 14 пін, А1 – 15 пін… А7 – 21.

Приклад, який опитує пін A0:

```
int value1 = analogRead(0); // зчитати напругу з піна A0
int value2 = analogRead(A0); // зчитати напругу з піна A0
int value3 = analogRead(14); // зчитати напругу з піна A0
```

Зберігати отримане значення розумно у змінній типу *int*, тому що значення варіюється від 0 до 1023.

Правила використання аналогових контактів

- якщо раніше виходи використовувалися як цифрові порти, використовувати команду *analogRead* некоректно;
- для коректної роботи необхідно налаштувати входи як аналогові. І навпаки, якщо аналоговий цифровий порт виступав зі значенням HIGH, його зворотна установка призведе до підключення резистора, що підтягує;
- не рекомендується швидке перемикання між аналоговими портами, оскільки це неминуче призведе до некоректної роботи системи;
- якщо ви використовуєте аналоговий порт як цифровий, в налаштуваннях необхідно задати функцію *PinMode()*, а порту присвоїти номер відповідного йому цифрового порту з 14 (для входу A0) по 19 (для входу A5).

Виведення аналогового сигналу

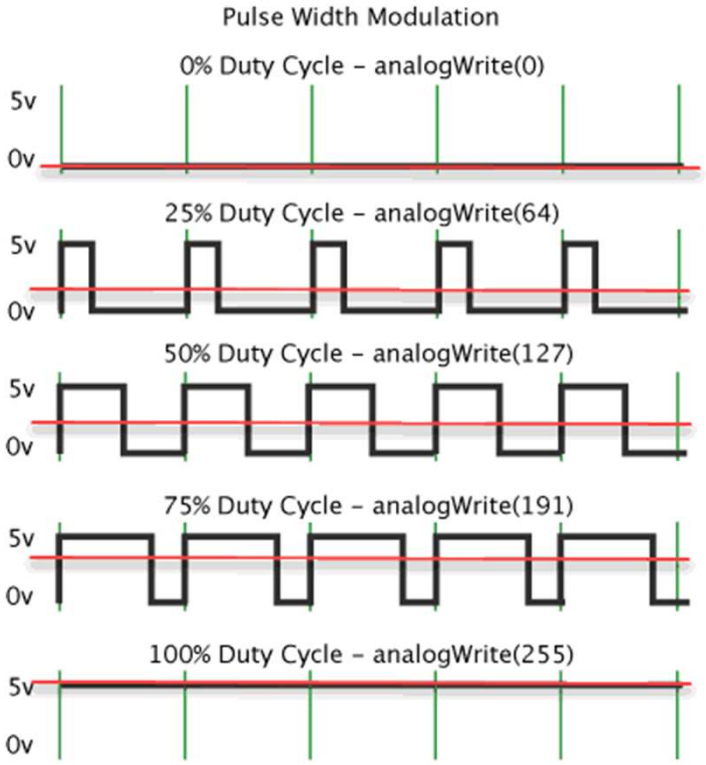
Arduino не містить вбудованого цифро-аналогового перетворювача (ЦАП), але можна використовувати цифровий сигнал з широтно-імпульсною модуляцією (ШІМ) для реалізації функцій по роботі з аналоговим виходом. Функція, що використовується для виведення ШІМ сигналу: *analogWrite(pin, value)*:

- *pin* – це номер виводу, який використовується для ШІМ виходу;
- *value* – це число, пропорційне коефіцієнту наповнення сигналу.

Коли значення = 0, на виході завжди логічний нуль. Коли значення = 255, на виході завжди логічна одиниця.

На більшості плат Arduino, ШІМ функції доступні на виходах 3, 5, 6, 9, 10 та 11. Частота ШІМ сигналу на більшості виходів становить приблизно 490 Гц. На Uno та подібних платах виходи 5 та 6 працюють на частоті приблизно 980 Гц.

Щоб порівняти аналогове вхідне значення, яке знаходиться в діапазоні від 0 до 1023, з вихідним ШІМ сигналом, що знаходиться в діапазоні від 0 до 255, можна використовувати функцію *map(value, fromLow, fromHigh, toLow, toHigh)*. Ця функція має п'ять параметрів: у першому зберігається аналогове значення, інші рівні відповідно 0, 1023, 0 і 255.



Приклад програми з використанням ШІМ:

```
const int pwm = 3;     // позначення виведення 3, як змінна 'pwm'

void setup()
{
  pinMode(pwm,OUTPUT);  // встановити режим виведення 3, як вихід
}

void loop()
{
  analogWrite(pwm,25);  // встановлення коефіцієнта заповнення, що дорівнює 25
  delay(50);       // затримка 50 мс
  analogWrite(pwm,50);
  delay(50);
  analogWrite(pwm,75);
  delay(50);
  analogWrite(pwm,100);
  delay(50);
  analogWrite(pwm,125);
  delay(50);
  analogWrite(pwm,150);
  delay(50);
  analogWrite(pwm,175);
  delay(50);
  analogWrite(pwm,200);
  delay(50);
  analogWrite(pwm,225);
  delay(50);
  analogWrite(pwm,250);
}
```

Лекцію закінчено
Дякую за увагу