УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ УПРАВЛІННЯ ПРОЦЕСАМИ ПЕРЕВЕЗЕНЬ

Кафедра вищої математики та фізики

МЕТОДИЧНІ РОЗРОБКИ

З ВИКОРИСТАННЯ ПЛАТФОРМИ ARDUINO У НАВЧАЛЬНОМУ ПРОЦЕСІ

Харків – 2021

Методичні розробки з використання платформи Arduino у навчальному процесі розглянуто і рекомендовано до друку на засіданні кафедри вищої математики та фізики 9 лютого 2021 р., протокол № 2.

Призначено для студентів усіх освітніх програм як додатковий матеріал; можна також використовувати для проведення гуртків з фізики.

Укладачі:

доценти А. Т. Котвицький,
К. А. Котвицька

Рецензент

доц. Н. В. Глейзер

ЗМІСТ

ВСТУП

Методичні розробки з використання платформи Arduino у навчальному процесі спрямовані на підвищення мотиваційної складової при вивченні фізичних основ. Передусім домогтися цього результату передбачається за рахунок об'єднання цифрових технологій, зокрема використання мікроконтролерів серії Atmega на платформі Arduino та базових основ електротехніки, які вивчаються в розділі електрики в курсі фізики. Поєднання сучасних інформаційних технологій і традиційних методів навчання дає змогу зробити процес вивчення нового матеріалу більш цікавим і актуальним, що дозволяє зрозуміти практичну цінність здобутих знань.

Реальний навчальний фізичний експеримент вимагає використання сучасних засобів вимірювання та аналізу фізичних процесів, що у свою чергу спрощує розуміння та розвиває модельне мислення.

Заняття 1. МІКРОСХЕМИ. МІКРОКОНТРОЛЕРИ. АПАРАТНІ ПЛАТФОРМИ

Одноплатний комп'ютер (SBC, англ. *single-board computer*) — самодостатній комп'ютер, зібраний на одній друкованій платі, на якій встановлені мікропроцесор, оперативна пам’ять, системи вводу-виводу та інші модулі, необхідні для функціонування комп'ютера.

Існує велика кількість виробників одноплатних комп’ютерів з використанням мобільної архітектури ARM. Серед них найбільш поширені подано в таблиці 1.

Таблиця 1

Raspberry Pi,
Banana Pi,
Orange Pi,
Cubieboard,
Odroid

Потрібна наявність операційної системи (подібної до LINUX).

Апаратні платформи — це мікроконтролери, а не повноцінні комп'ютери. На них немає операційної системи (таблиця 2).

Здебільшого апаратні платформи використовують для побудови систем автоматики та робототехніки. Їхнє головне завдання — керування пристроями.

Таблиця 2

Particle Photon
Teensy
Netduino
Ti MSP430 LaunchPad
Arduino

Одноплатні апаратні платформи — це не аналог, а альтернатива одноплатних комп’ютерів.

Мікроконтролер – це самостійна комп'ютерна система, яка має процесор, допоміжні схеми та пристрої вводу-виводу даних, що розміщені у загальному корпусі (таблиця 3).

Таблиця 3

Мікроконтролер	Загальна мікросхема
	ULN2003A L293D

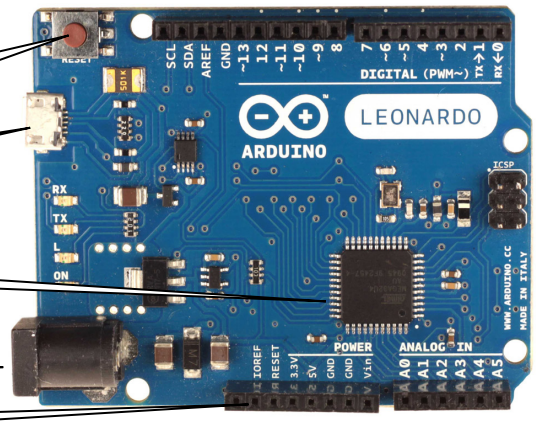
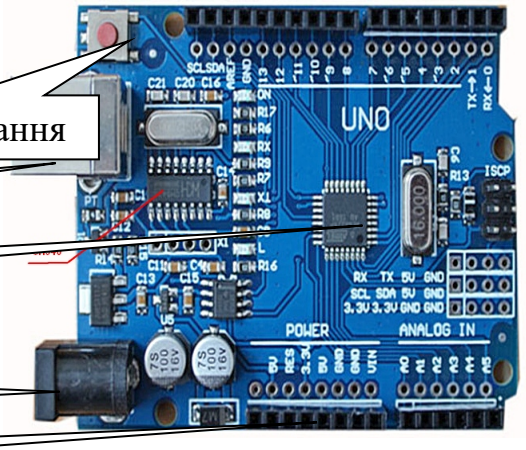
Запишемо основні відмінності роботи мікроконтролера та мікросхеми (таблиця 4).

Таблиця 4

Мікроконтролер	Звичайна мікросхема
0) програмуємо мікроконтролер; *1) подаємо на одні виходи певні сигнали;* *2) отримуємо з інших виходів у відповідь сигнали (відповідно до закладеної програми)*	*0) відсутнє;* *1) подаємо на одні виходи певні сигнали;* *2) отримуємо з інших виходів у відповідь сигнали (відповідно до конструкції мікросхеми)*

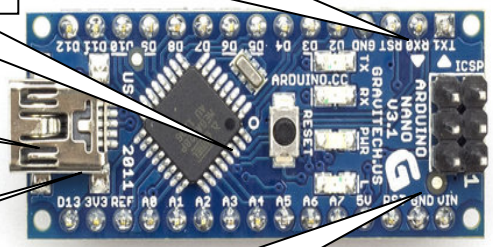
Найчастіше використовують такі типи мікроконтролерів: PIC, AVR, MSP430, STM32, де AVR (здебільшого) – мікроконтролер Arduino-платформи. Розглянемо деякі типи Arduino (таблиця 5) [1, 2]:

Таблиця 5

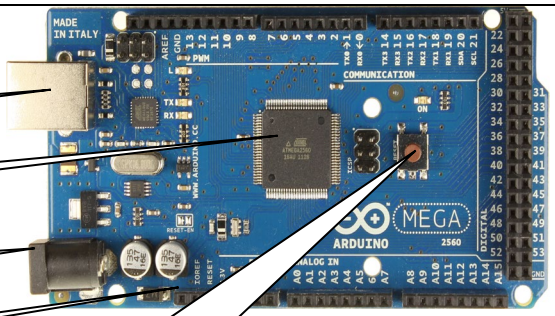
Leonardo — плата на мікроконтролері ATmega32U4 Кнопка скидання Зв'язок з комп'ютером Мікроконтролер Зовнішнє живлення від мережевого адаптера Контакти (піни)	
UNO — найбільш популярна версія базової платформи Arduino Кнопка скидання Зв'язок з комп'ютером Мікроконтролер Зовнішнє живлення від мережевого адаптера Контакти (піни)	

Продовження таблиці 5

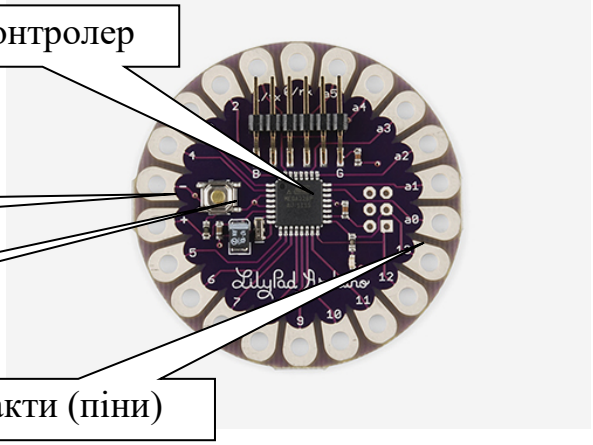
Nano — компактна платформа, що використовується як макет. Nano підключається до комп'ютера за допомогою кабеля USB Mini-B



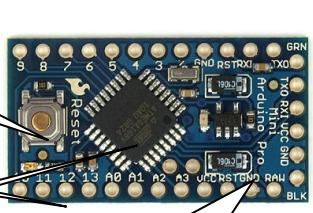
Mega 2560 — плата на основі мікроконтролера ATmega2560



LilyPad — платформа, розроблена для перенесення, може бути вшита в тканину

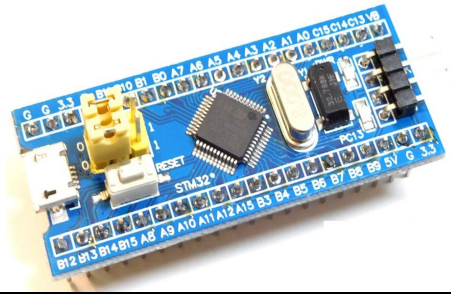
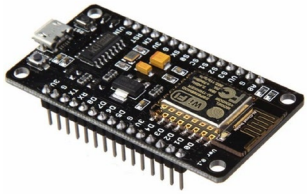
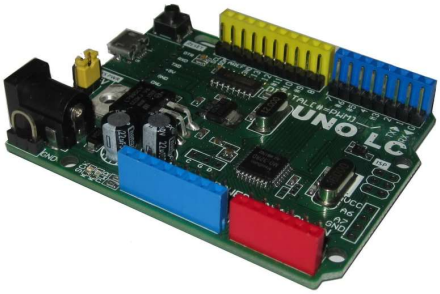


Pro Mini — як і платформа Pro, розроблена для досвідчених користувачів, яким потрібна низька ціна, менші розміри та додаткова функціональність



Завдяки величезній популярності Arduino ентузіастами було написано бібліотеки і програми, що дають змогу програмувати інші платформи засобами Arduino. Найбільш поширені подано в таблиці 6.

Таблиця 6

STM32

NodeMCU (спеціальна прошивка для мікроконтролера ESP8266)

MassDuino UNO LC


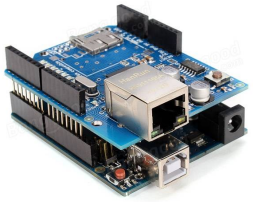
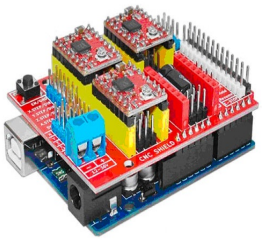
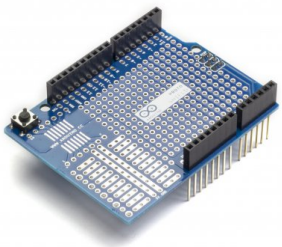
Робота з цією платою повністю аналогічна до роботи з Arduino UNO, але маємо низку додаткових можливостей. Запишемо деякі з них:

- можливість перемичкою на платі **вибирати логічні рівні:** 5 В або 3,3 В;
- два додаткові аналогові входи;
- роздільна здатність АЦП до 16 біт;
- два 8-розрядні справжні цифро-аналогові перетворювачі, а не широтно-імпульсна модуляція (ШІМ).

Заняття 2. ПЛАТИ РОЗШИРЕННЯ (ШИЛДИ SHIELD)

Для збільшення можливостей платформ Arduino розроблено велику кількість окремих плат (шилдів), які легко підключаються і мають істотний функціонал (таблиця 7) [3].

Таблиця 7

Ethernet Shield Робота в Internet	
Wi-Fi	
CNC Керування кроковими двигунами	
GSM/GPRS Мобільний зв'язок та СМС	
Proto Shield Для створення своїх пристроїв	

Цифрові та аналогові сигнали

Аналоговий сигнал змінюється **безперервно**, тобто якщо в даний момент часу на виході, наприклад, 3,2372 В, то в

наступний момент величина сигналу може змінитися на довільно мале значення (наприклад 3,23721 В).

Цифровий сигнал змінюється **дискретно**, тобто показаний цілим числом. Для платформ Arduino характерним є 10-бітний цифровий сигнал. Це означає, що весь інтервал напруг від 0 до 5 вольт (В) розбивається на 1023 інтервали (рисунок 1).

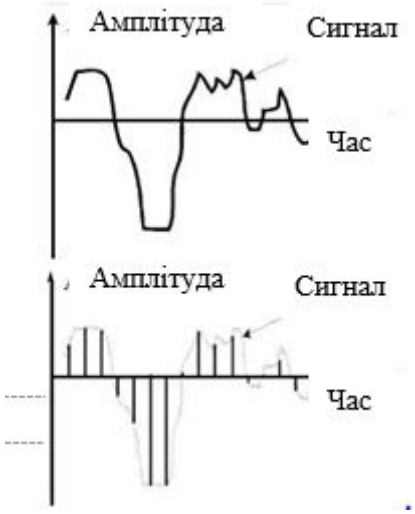


Рисунок 1

Тобто ми отримаємо, що 0 В відповідає значенню 0, а 5 В – значенню 1023. Відповідність між цифровим сигналом та будь-яким іншим значенням напруги визначається за допомогою пропорції. Наприклад, 3,2372 В згідно з пропорцією дорівнює:

$$3{,}2372\,\frac{1023}{5} = 662{,}33112\,.$$

Цифровий сигнал — це лише цілі числа, тоді 3,2372 В відповідає значенню 662. Особливу увагу необхідно звернути на те, що і напруга 3,2356 і 3,24046 В буде в цифровому вигляді відповідати числу 662. Тобто при оцифруванні ми втрачаємо точність, але при цьому збільшуємо перешкодозахищеність сигналу, який передається. Захищеність від перешкод зумовлена тим, що будь-яке число передається двійковим кодом, тобто у вигляді набору нулів та одиниць (низьким або високим рівнем сигналу).

Для розуміння суті двійкового коду (двійкової системи числення) пограємо у гру «Вгадай число від 0 до 15», відповідаючи лише «Так» або «Ні».

Нехай задумано число 11.

Можна, звичайно, запитувати: «Задумано число 0? Задумано число 1?» тощо, але найбільш нераціональний спосіб, при якому кількість запитань залежить від задуманого числа.

Покажемо алгоритм, який дає змогу відгадати число завжди за чотирма запитаннями. Для цього будемо записувати числа у два стовпчики, ставити одне й те саме запитання і залежно від відповіді записувати інші числа у два стовпчики (таблиця 8).

1-ше запитання: Число міститься в лівому стовпчику?

0, 1. 2, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, 14, 15.

Відповідь: Ні.

Отже, число міститься в правому стовпчику.

2-ге запитання: Число міститься в лівому стовпчику?

8, 9, 10, 11, 12, 13, 14, 15.

Відповідь: Так.

3-тє запитання: Число міститься в лівому стовпчику?

8, 9, 10, 11.

Відповідь: Ні.

4-те запитання: Число міститься в лівому стовпчику?

10, 11.

Відповідь: Ні.

Чотири запитання, і ми знаємо відповідь – 11.

Таблиця 8

0	**0**	*0000*
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001
10	1010	1010
11	1011	1011
12	1100	1100
13	1101	1101
14	1110	1110
15	1111	1111

Але побудова запитання залежить від попередньої відповіді. Як ставити запитання, щоб вони не залежали від попередніх відповідей?

Поставимо у відповідність кожному числу від 0 до 15 (перший стовпчик) числа, що складаються лише із нулів та одиниць (другий стовпчик), і допишемо незначущі нулі зліва (третій стовпчик). Будемо запитувати таким чином, щоб

з’ясувати послідовність нулів та одиниць для задуманого числа. Так, для нашого випадку отримаємо.

1 Перша цифра 1? *Так*.

2 Друга цифра 1? *Ні.*

3 Третя цифра 1? *Так*.

4 Четверта цифра 1? *Так*.

Відповідь: отримали 1011, тобто задумане число – 11.

Кількість інформації — це міра зменшення невизначеності. **1 біт** – кількість інформації, яку містить повідомлення, що зменшує невизначеність знань у два рази. Біт — це найменша одиниця вимірювання інформації.

1 Байт = 8 біт.

1 кБ (кілобайт) = 2^{10} Байт = 1024 Байт = 8192 біт.

1 МБ (мегабайт) = 2^{10} кБ = 1024 кБ = 8388608 біт.

1 ГБ (гігабайт) = 2^{10} МБ = 1024 МБ = 8589934592 біт.

Переведення двійкового числа в десяткове

Переведемо двійкове число 1101_2 в десяткове. Для цього перенумеруємо цифри цього числа справа наліво, починаючи з нуля (таблиця 9).

Таблиця 9

3	2	1	0
1	1	0	1

Тепер складемо наступну суму, де числа ступеня двійки взято із першого рядка таблиці, а множники перед відповідними двійками – із другого рядка, отримаємо:

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13,$$

де цифра 2 в числі 1101_2 означає, що воно записане у двійковій системі числення.

Завдання. Покажіть, що справедливі такі рівності:

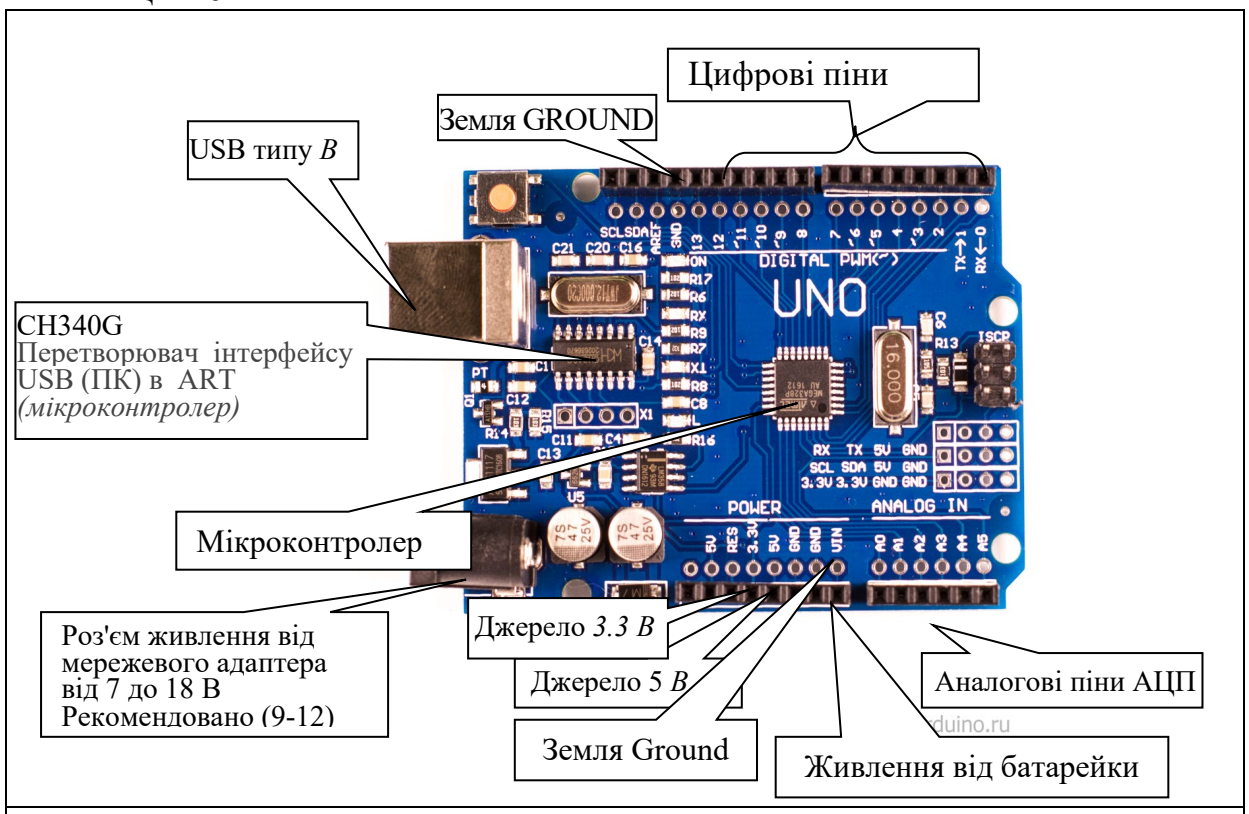
$$110100111_2 = 423.$$
$$1001100011011101_2 = 78269.$$

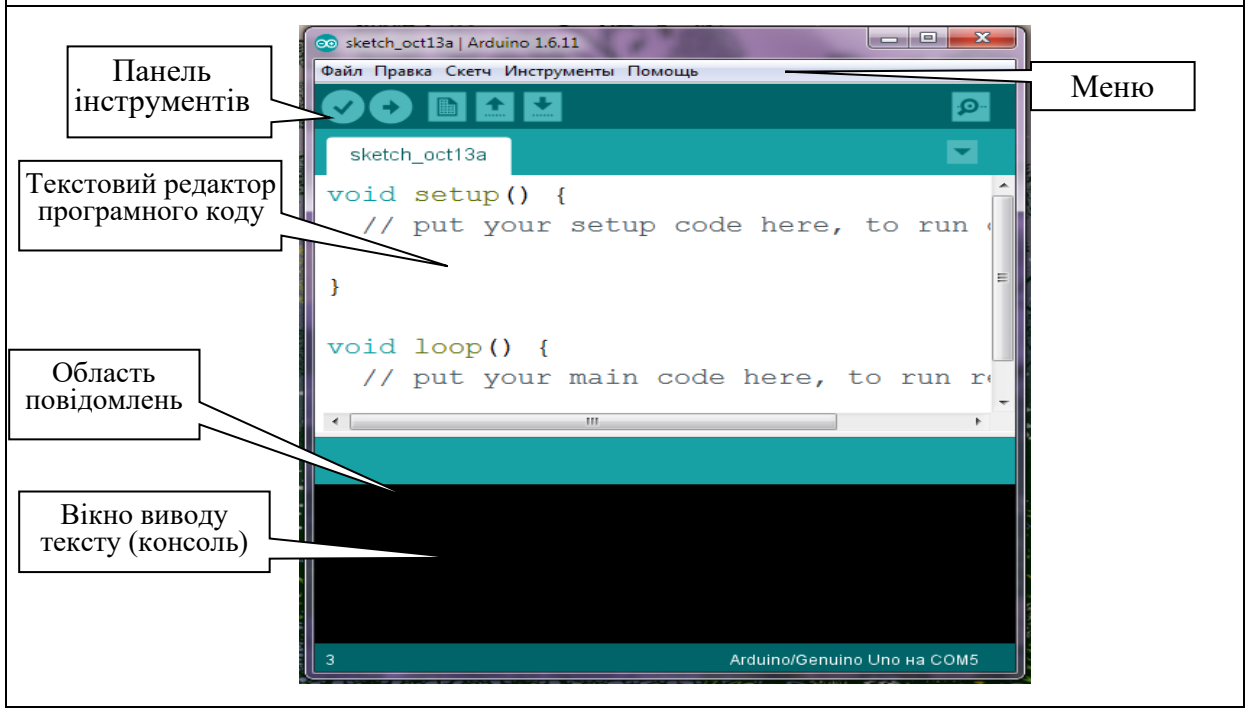
Покажіть це як вручну, так і за допомогою калькулятора програміста.

Заняття 3. ПЛАТФОРМА ARDUINO UNO R3 ТА СЕРЕДОВИЩЕ РОЗРОБКИ ARDUINO IDE

Розглянемо плату Arduino UNO R3 [1–4]. Для роботи з будь-якою платою Arduino необхідно скачати і встановити середовище розробки **Arduino IDE** (таблиця 10).

Таблиця 10





Останню версію можна знайти на офіційному сайті Arduino для різних операційних систем https://www.arduino.cc/en/Main/Software.

Середовище розробки Arduino IDE

Скетч – програма, яка написана в середовищі Arduino. Скетч пишеться в текстовому редакторі, що має інструменти вирізання/вставки, пошуку/заміни тексту (рисунок 2).

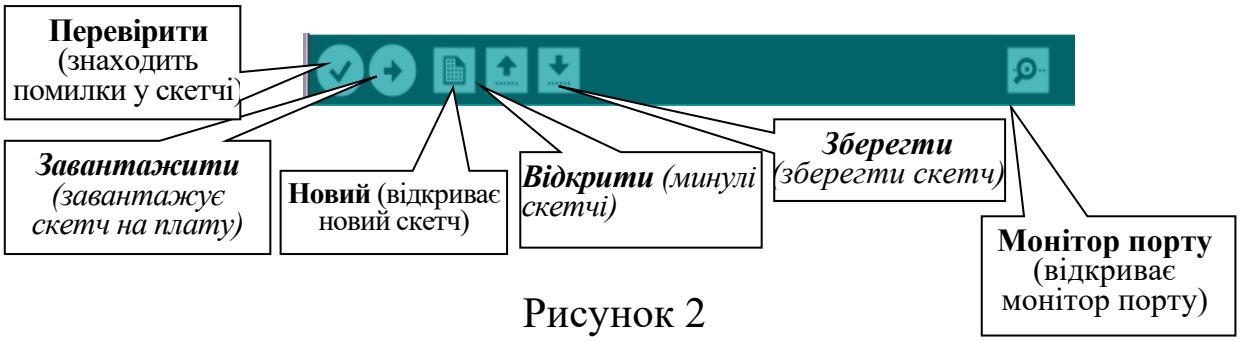


Рисунок 2

Під час збереження і завантаження проекту в області повідомлень з'являються пояснення, можуть відображатися помилки. Вікно виводу тексту (консоль) показує повідомлення Arduino, яке містить повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дають змогу перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини (монітор порту).

Панель інструментів

Для зручності роботи можна використовувати гарячі клавіші, покажемо найчастіше використовувані (рисунок 3).

- Ctrl+N відкрити новий скетч
- Ctrl+S зберегти
- Ctrl+Shift+S зберегти як
- Ctrl+C/ Ctrl+V копіювати/вставити
- Ctrl+R перевірити скетч
- Ctrl+U завантажити скетч на плату

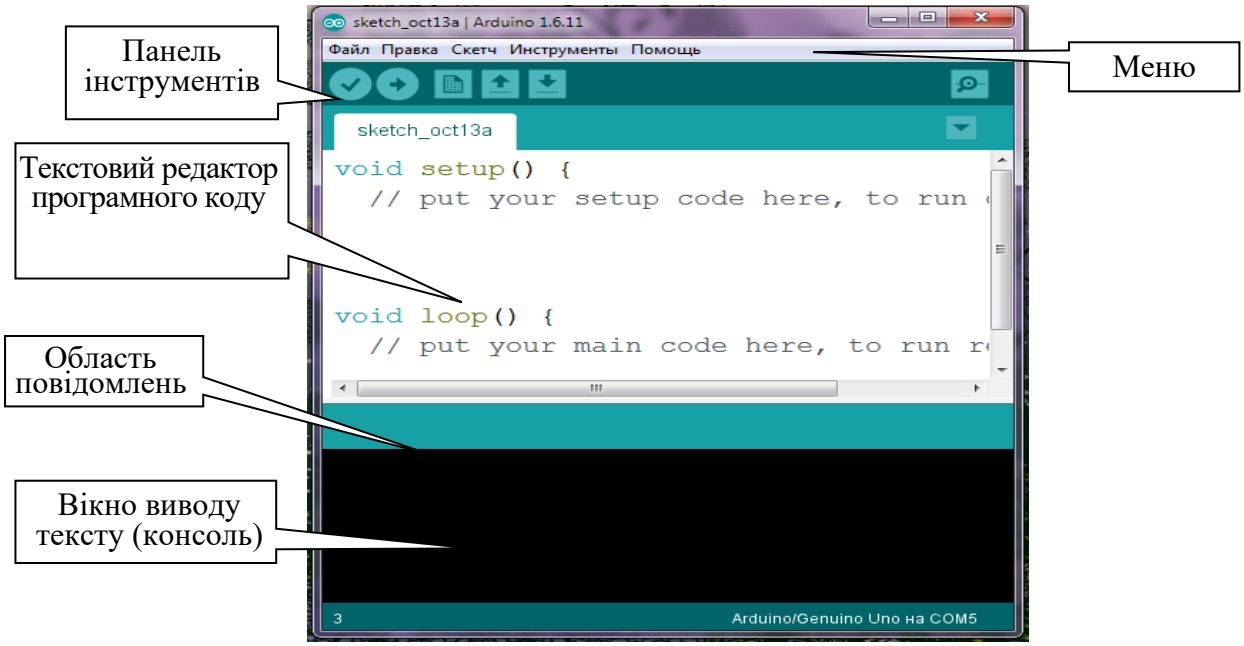


Рисунок 3

Підключення плати Arduino до комп'ютера

При **кожному підключенні** потрібно обов'язково зробити кілька важливих налаштувань:

1 Вибрати потрібну плату Arduino UNO (рисунок 4).

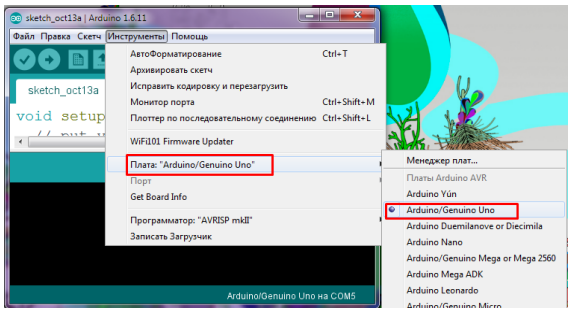


Рисунок 4

2 Вибрати потрібний порт, до якого підключена плата (рисунок 5).

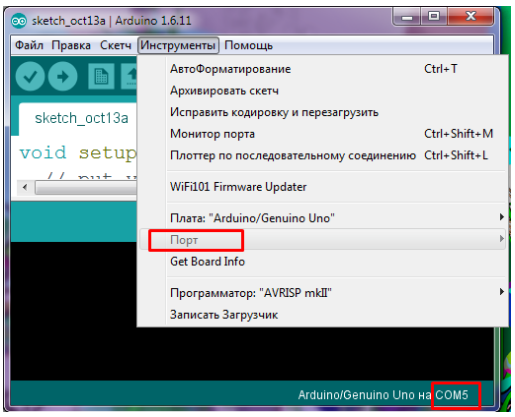


Рисунок 5

3 Вибрати програматор AVRISP mkII (рисунок 6).

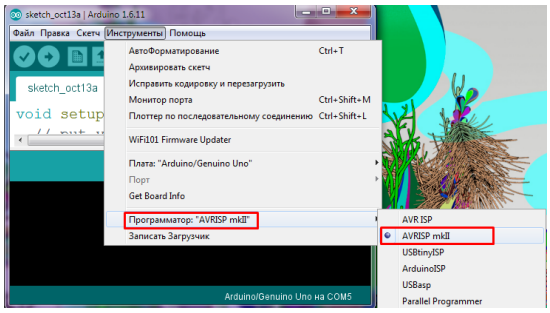


Рисунок 6

За час заняття, що залишився, можна подивитися приклад програми:

Файл –> Приклади –> 01.Basics–> Blink.

Заняття 4. ОСНОВИ ПРОГРАМУВАННЯ НА ARDUINO

Мова програмування WIRING (C/C++) Processing/Wiring — це, фактично, звичайний C++, доповнений простими та зрозумілими функціями для керування вводом-виводом на контактах.

Дуже важливо: великі та малі літери в середовищі Arduino IDE розрізняються, тобто слова **Led** і **led** компілятор сприймає як два різних слова.

Мінімальна програма, яку можна запустити на Arduino, складається лише з двох функцій:

setup() {}
loop() {}

Перша функція **setup** *– викликається лише один раз, після запуску Arduino зазвичай слугує для первинного налаштування.*

Друга функція **loop** *– викликається безліч разів за час роботи Arduino.*

Напишемо першу програму, яка змусить Arduino UNO миготіти вмонтованим світлодіодом із заданою частотою. Для початку напишемо функцію **setup,** в якій зробимо необхідне первинне налаштування:

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

Службове слово **void** означає, що ця функція не повертає жодне число.

Приклад 1. Функція $y = x^2$. Це означає, що ця функція повертає квадрат числа. Тобто якщо функція отримує 3, то повертає 9.

Приклад 2. Функція «*візьми книгу, поклади на стіл*». Говорять, що ця функція не повертає число, а виконує деяку дію.

Функція **setup()** записується з порожніми круглими дужками. Це означає, що ми не надаємо їй жодних параметрів. Далі пишемо відкриту і закриту фігурні дужки. Між цими дужками пишемо потрібні оператори. В даному випадку між фігурними дужками стоїть функція:

pinMode (pin, mode);

Вона конфігурує pin (параметр, який вказано першим у круглих дужках) у режимі «mode».

Для Arduino UNO як параметр pin вказується номер, якому відповідають ніжки мікроконтролера від 0 до 13 (для цифрових пінів), від 14 до 19 (для аналогових пінів A0, A1, A2, A3, A4, A5, розглянутих як цифрові). Параметр mode в основному виді набуває двох значень:

INPUT, **OUTPUT**

Перше значення пишуть, коли потрібно налаштувати відповідний пін на ВХІД. Тобто на цю ніжку мікроконтролера

будуть подаватися деякі дані, наприклад, від датчика, а нам необхідно ці дані отримувати і обробляти.

Друге значення пишуть, коли потрібно налаштувати відповідний пін на ВИХІД. Тобто станом цієї ніжки керує мікроконтролер, а зовнішні пристрої, наприклад лампочка, отримують ці дані і відповідно або вмикаються, або вимикаються.

У цьому випадку ми будемо використовувати 13-й пін. До цієї ніжки на платформі Arduino UNO вмонтований світлодіод. При високому рівні (+5 В) на піні світлодіод світиться, при низькому (0 В) – гасне.

У тілі функції **loop** пишемо основний код, який буде вмикати та вимикати світлодіод. Для цього використовуємо функцію

digitalWrite(pin, value);

Як і в попередньому випадку параметр pin являє собою номер ніжки мікроконтролера (в нашому випадку 13).

Параметр value може набувати двох значень:

HIGH або LOW.

Якщо параметр *value* має значення HIGH, то на відповідному піні встановлюється високий потенціал (+5 В), а якщо **LOW**, то низький (0 В або «земля»).

Для того щоб світлодіод світився (або не світився) потрібну кількість часу, будемо використовувати функцію

delay(ms),

яка на час **ms** призупиняє виконання коду, де **ms** задається в мілісекундах. Нагадаємо, що одна секунда дорівнює 1000 мілісекунд:

1 с = 1000 мс.

Отже, маємо програму:

Prog4

```
void setup() {          // початок налаштування, відкрита дужка
  pinMode(13, OUTPUT); // оголошення 13-го піна як вихід
}                       // кінець налаштувань, закрита дужка
void loop() {           // початок основної програми, відкрита дужка
```

```
    digitalWrite(13, HIGH); // подаємо на 13-й пін високий рівень
    delay(400);             // чекаємо 400 мілісекунд
    digitalWrite(13, LOW);  // подаємо на 13-й пін низький рівень
    delay(400);             // чекаємо 400 мілісекунд
  }                         // кінець основної програми, закрита дужка
```

1 Наберемо цей скетч і збережемо його.

2 Оскільки це наша перша програма, то спочатку створимо на жорсткому диску (або іншому за рекомендацією викладача) в папці Arduino власну папку під назвою свого прізвища (бажано латинськими літерами).

3 У своїй папці створимо підпапку з номером заняття, тобто **lesson4,** і в ній будемо зберігати всі файли, які необхідні нам для цього заняття.

Починаючи з наступного заняття, ці дії студенти мають виконувати самостійно на початку кожного заняття. Збережемо написаний скетч у своїй підпапці **lesson4** під ім'ям Prog4, де номер 4 означає, що дана програма написана на четвертому занятті. Після цього підключаємо Arduino UNO в USB-порт.

Перевіримо в середовищі Arduino IDE, що встановлена потрібна плата і програматор (дивись заняття 3). Виберемо потрібний **com** порт і натиснемо кнопку «перевірити скетч». Після вдалої компіляції програми (якщо немає помилок) натискаємо кнопку «завантажити».

Завдання: написати програму, яка передає азбукою Морзе сигнал SOS (СОС) — міжнародний сигнал лиха. Сигнал являє собою послідовність «три крапки — три тире — три крапки», яка передається без будь-яких інтервалів.

Азбука Морзе – це спосіб передавання літер абетки, цифр і знаків пунктуації послідовністю сигналів: довгих («тире») і коротких («крапок»). За одиницю часу приймається довжина однієї крапки. Довжина тире дорівнює трьом крапкам. Пауза між елементами одного знака — одна крапка, між знаками в слові — три крапки, між словами — сім крапок. Тобто маємо шість «крапок» і три «тире».

Запишемо програму:

Prog4a

```
    void setup() {                  // первинне налаштування, відкрита дужка
      pinMode(13, OUTPUT);  // оголошення 13-го піна як вихід
    }                                   // кінець первинного налаштування, закрита дужка
    void loop() {               // початок основної програми, відкрита дужка
      digitalWrite(13, HIGH); // початок першої «крапки», світлодіод світиться
      delay(200);                   // затримка 200 мілісекунд
      digitalWrite(13, LOW);   // світлодіод вимикається
      delay(200);                   // кінець першої «крапки»
      digitalWrite(13, HIGH);  // початок другої «крапки»
      delay(200);
      digitalWrite(13, LOW);
      delay(200);                   // кінець другої «крапки»
      digitalWrite(13, HIGH);  // початок третьої «крапки»
      delay(200);
      digitalWrite(13, LOW);
      delay(200);                   // кінець третьої «крапки»
      digitalWrite(13, HIGH);  // початок першого «тире»
      delay(600);
      digitalWrite(13, LOW);
      delay(200);                 // кінець першого «тире»
      digitalWrite(13, HIGH); // початок другого «тире»
      delay(600);
      digitalWrite(13, LOW);
      delay(200);                   // кінець другого «тире»
      digitalWrite(13, HIGH); // початок третього «тире»
      delay(600);
      digitalWrite(13, LOW);
      delay(200);                   // кінець третього «тире»
      digitalWrite(13, HIGH); // початок четвертої «крапки»
      delay(200);
      digitalWrite(13, LOW);
      delay(200);                   // кінець четвертої «крапки»
      digitalWrite(13, HIGH); // початок п’ятої «крапки»
```

```
  delay(200);
  digitalWrite(13, LOW);
  delay(200);                  // кінець п’ятої «крапки»
  digitalWrite(13, HIGH); // початок шостої «крапки»
  delay(200);
  digitalWrite(13, LOW);
  delay(200);                  // кінець шостої «крапки»
}
```

Заняття 5. МАКЕТНА ПЛАТА (BREADBOARD) ТА ОСНОВНІ ЕЛЕМЕНТИ, ПАРАЛЕЛЬНЕ ТА ПОСЛІДОВНЕ З’ЄДНАННЯ ЇХ

Основне призначення макетної плати — конструювання та налагодження прототипів різноманітних пристроїв. Складається цей пристрій з отворів-гнізд із кроком 2,54 мм (0,1 дюйма), саме з таким (або кратним йому) кроком розміщуються виходи на більшості сучасних радіодеталей (рисунок 7).

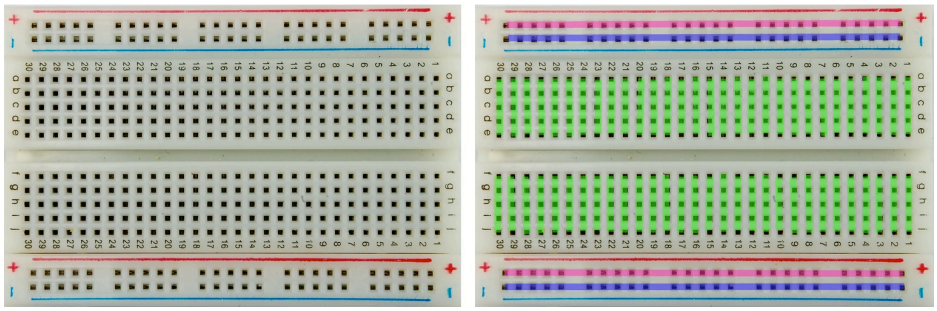


Рисунок 7

На рисунку 7 зліва показано загальний вигляд плати. На правій частині рисунка кольором позначені шини-провідники. *Синій колір* – це «мінус» схеми, *червоний* – «плюс», *зелений* – це провідники, які ви можете використати на свій розсуд для з’єднання частин електричної схеми, яка складається на макетній платі. Звернемо увагу, що центральні отвори з’єднані паралельними рядами поперек макетної плати, а не вздовж, на відміну від шин живлення, які розміщені по краях макетної плати. Як бачимо, маємо дві пари шин живлення, що дає змогу за

потреби подавати на плату дві різні напруги, наприклад 5 В і 3,3 В. Отвори макетної плати з'єднуються за допомогою радіоелементів (резисторів, діодів, кнопок, мікросхем і т. д.), а також за допомогою з'єднувальних проводів (рисунок 8).

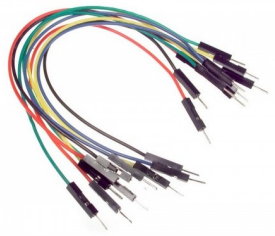


Рисунок 8

Розглянемо принципову електричну схему підключення світлодіода (рисунок 9).

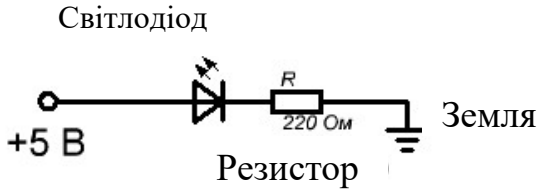


Рисунок 9

Розберемо елементи наведеної схеми (таблиця 11). Після підключення всіх елементів за схемою світлодіод має світитися.

Таблиця 11

	Клема для підключення живлення +5 В
	Світлодіод
	Резистор *R* на 220 Ом
	Земля, GROUND (Gnd), «мінус» живлення

Світлодіод являє собою напівпровідниковий пристрій, здатний випромінювати світло при пропусканні через нього електричного струму в прямому напрямку (від *анода* до *катода*).

На принциповій схемі анод позначається дротом, який виходить із сторони трикутника, а катод – із вершини трикутника та відрізка (рисунок 10). Для того щоб світлодіод світив, на анод необхідно подати «плюс», а на катод – «мінус».

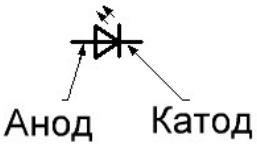
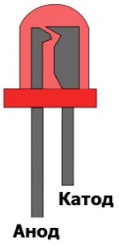


Рисунок 10

Для визначення ніжок реального світлодіода необхідно або подивитися на їхню довжину (анод більш довгий) (рисунок 11, а), або на саму колбу (бік катода зрізаний) (рисунок 11, б).

а)



б)

Рисунок 11

Для зображення монтажних схем ми будемо користуватися програмою «*Fritzing*», в якій анод світлодіода показано більш довгою ламаною лінією (рисунок 12).

Рисунок 12

При підключенні світлодіода до напруги 5 В необхідно користуватися струмообмежувальним резистором номіналом 220 Ом, який потрібно вмикати *послідовно* до світлодіода (рисунок 13).

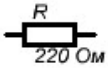


Рисунок 13

Розрахунок цього резистора буде розглянуто на наступних заняттях. Зовнішній вигляд та позначення на принципових електричних схемах подано на рисунку 14.

Рисунок 14

Зауважимо, що у резистора ніжки рівнозначні, і яким боком його встановлювати, не має значення. Звернемо увагу, що нам необхідний резистор з кольоровими смужками: червона, червона, коричнева, золота, тобто 220 Ом.

Зобразимо монтажну схему (використовуючи вільно поширений програмний продукт FRITZING) підключення світлодіода, що відповідає принциповій схемі (дивись рисунок 8).

Будемо використовувати Arduino як джерелj 5 В. Після підключення Arduino UNO до USB світлодіод vf’ засвітитися (рисунок 15).

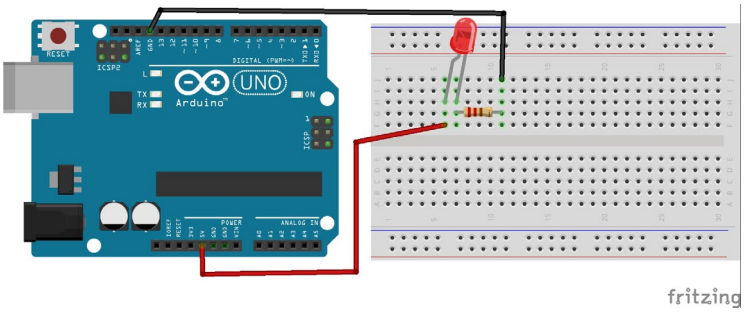


Рисунок 15

Зобразимо монтажну схему, яка буде також еквівалентна попередній, але будемо використовувати шину живлення (рисунок 16).

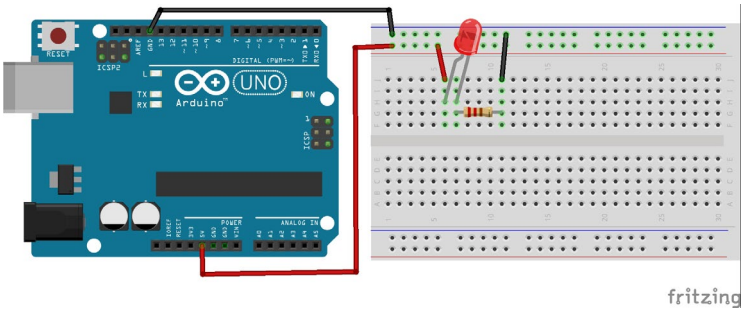


Рисунок 16

Завдання: розробити і скласти еквіваленти наведених монтажних схем, але використовуючи інші гнізда макетної плати.

Підключимо тепер три світлодіоди **паралельно**, за електричною принциповою схемою (рисунок 17).

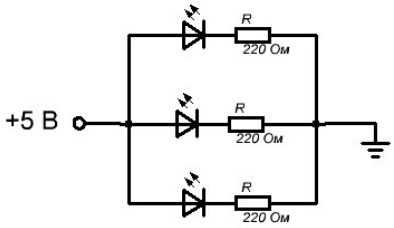


Рисунок 17

Звернемо увагу на те, що до кожного світлодіода послідовно ввімкнено резистор на 220 Ом.

Складемо монтажну схему (рисунок 18):

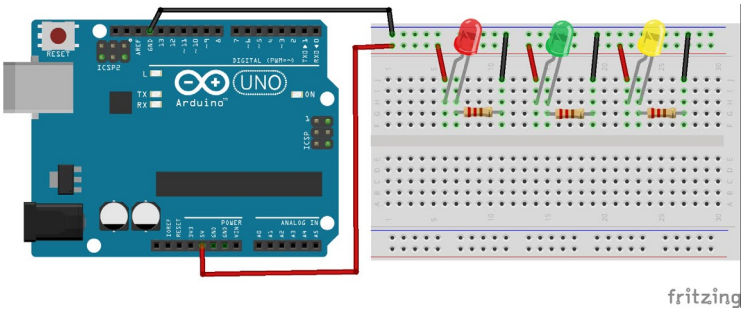


Рисунок 18

Закони паралельного з’єднання також буде розглянуто на наступних заняттях.

Підключимо три світлодіоди послідовно згідно з принциповою схемою (рисунок 19):

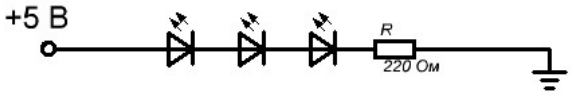


Рисунок 19

Складемо монтажну схему (рисунок 20).

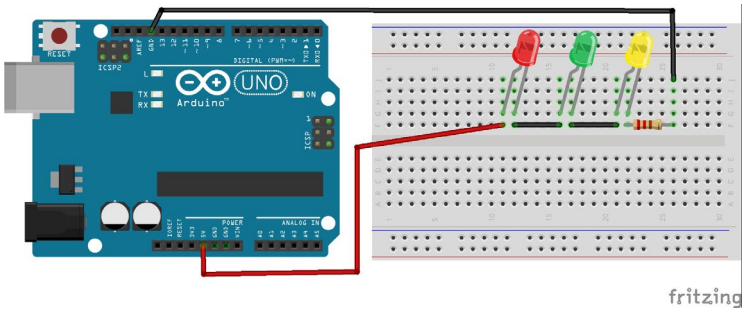


Рисунок 20

Зауважимо, що у випадку послідовного з'єднання світлодіодів достатньо одного резистора, а у випадку більшої кількості світлодіодів можна взагалі обійтися без нього.

Заняття 6. КЕРУВАННЯ ЗОВНІШНІМ СВІТЛОДІОДОМ. ПРЯМА ТА ОБЕРНЕНА ЛОГІКА РОБОТИ. СВІТЛОФОР

На занятті 5 ми підключали світлодіоди до Arduino UNO, але при цьому використовували платформу Arduino як джерела живлення (в якомусь сенсі, як «батарейку»). Фактично це означає, що на занятті 5 ми просто перевірили світлодіоди на прецездатність. При цьому не було жодного програмного керування світлодіодами.

Тепер будемо керувати зовнішнім світлодіодом подібно до того, як ми керували вбудованим світлодіодом на 13-му піні на занятті 4. Для програмного керування світлодіодом підключимо його анод на будь-який цифровий пін (наприклад четвертий) згідно з принциповою схемою (рисунок 21).

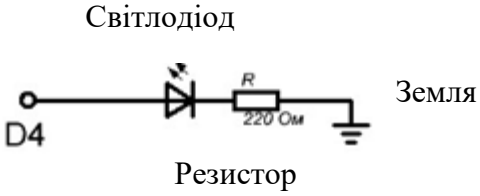


Рисунок 21

Істотна відмінність цієї схеми від схеми із заняття 5 полягає в тому, що замість живлення +5 В підключаємо анод до цифрового 4-го піна (digital 4). Зобразимо монтажну схему (рисунок 22):

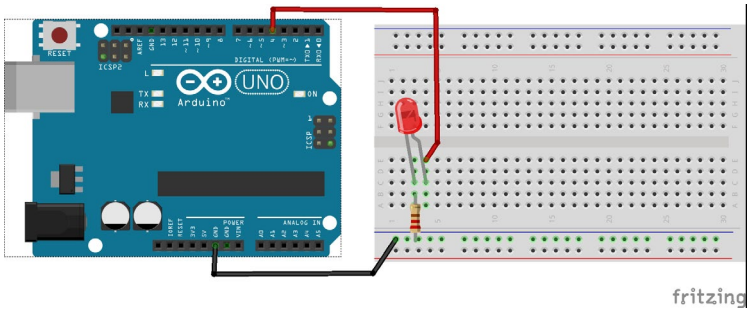


Рисунок 22

Нагадаємо, що ніжки у світлодіода розрізняються і при неправильному встановленні працювати він не буде (дивись заняття 5). Логіка роботи цієї схеми така:

- коли на 4-й пін подаємо сигнал **HIGH**, то світлодіод вмикається, оскільки це відповідає подачі +5 В на анод;

- коли на 4-й пін подаємо сигнал **LOW**, то світлодіод вимикається, оскільки це відповідає подачі напруги 0 В на анод (тобто землі або **GROUND**). Напишемо скетч, аналогічний до першої програми на занятті 4.

Prog6

```
void setup() {
  pinMode(4, OUTPUT);    // оголошення 4-го піна як вихід
}
void loop() {
  digitalWrite(4, HIGH);    // подаємо на 4-й пін високий рівень
  delay(1000);              // чекаємо 1000 мілісекунд
  digitalWrite(4, LOW);     // подаємо на 4-й пін низький рівень
  delay(200);               // чекаємо 200 мілісекунд
}
```

Завдання: згадайте, що означає кожний рядок коду.

Після завантаження наведеного скетчу в Arduino UNO світлодіод світиться 1000 мілісекунд (одну секунду) і не світиться 200 мілісекунд. Така логіка роботи світлодіода називається

прямою, оскільки рівень **HIGH** вмикає світлодіод, а рівень **LOW** його вимикає.

Іноді виникає потреба керувати світлодіодом за допомогою оберненої логіки. Для цього розглянемо принципову схему вимикання (рисунок 23).

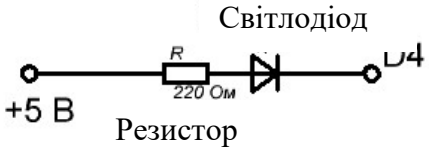


Рисунок 23

Одна з відмінностей цієї схеми від попередньої полягає у тому, що резистор і світлодіод поміняли місцями. Однак ця відмінність не принципова, тобто не впливає на роботу схеми, вона зроблена для зручності підключення.

Принциповою відмінністю є те, що тепер до 4-го цифрового піна підключено катод світлодіода. Це призводить до того, що при подачі сигналу **LOW** світлодіод вмикається, а при сигналі **HIGH** світлодіод вимикається. Змінимо монтажну схему підключення (рисунок 24):

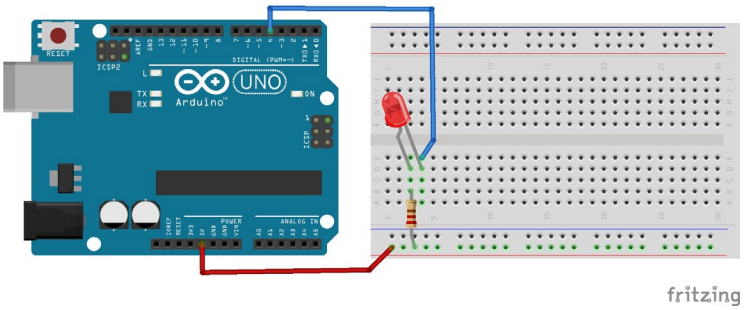


Рисунок 24

Завантажимо той самий скетч **Prog6** в Arduino UNO (або просто подамо живлення на Arduino за умови, що цей скетч там вже є). Тепер світлодіод світиться 200 мілісекунд, не світиться 1000 мілісекунд (одну секунду). Тобто рівень **HIGH** вимикає світлодіод, а рівень **LOW** його вмикає. Така логіка роботи називається оберненою.

Будемо тепер керувати одразу трьома світлодіодами. Для цього підключимо їх за поданою нижче схемою (рисунок 25).

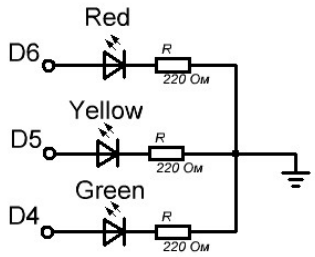


Рисунок 25

Із схеми випливає, що три світлодіоди ввімкнено згідно з прямою логікою. Напишемо тепер програму, яка буде імітувати роботу світлофора таким чином:

1) світить (3 секунди) тільки червоний світлодіод;

2) не вимикаючи червоного сигналу, ввімкнемо жовтий світлодіод на одну секунду;

3) вимикаємо червоний і жовтий, вмикаємо зелений (три секунди);

4) вимикаємо зелений сигнал, вмикаємо жовтий на одну секунду;

5) далі повторюємо.

Складемо монтажну схему (рисунок 26).

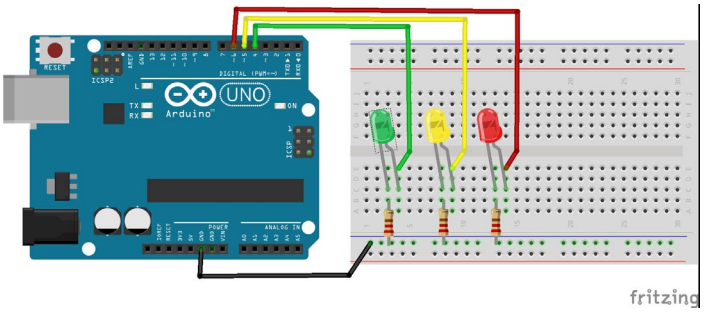


Рисунок 26

Напишемо скетч:

Prog6a

```
void setup(){
  pinMode(4, OUTPUT);  // ініціалізація Green
  pinMode(5, OUTPUT);  //  ініціалізація Yellow
  pinMode(6, OUTPUT);  // ініціалізація Red
  digitalWrite(4, LOW);   // вимикаємо Green
  digitalWrite(5, LOW);   // вимикаємо Yellow
  digitalWrite(6, LOW);   // вимикаємо Red
```

```
}
   void loop() {
   digitalWrite(6, HIGH);
        delay(3000);
   digitalWrite(5, HIGH);
        delay(1000);
   digitalWrite(6, LOW);
   digitalWrite(5, LOW);
   digitalWrite(4, HIGH);
        delay(3000);
   digitalWrite(4, LOW);
   digitalWrite(5, HIGH);
        delay(1000);
   digitalWrite(5, LOW);
}
```

Завдання 1: пояснити всі рядки коду.

Завдання 2: навіщо потрібно вимикати світлодіоди після ініціалізації пінів?

Відповідь: для визначеності, щоб ми знали, що на пінах знаходиться низький рівень, тобто світлодіоди спочатку вимкнені.

Заняття 7. ЗАКОН ОМА. ЗАКОНИ ПОСЛІДОВНОГО І ПАРАЛЕЛЬНОГО З'ЄДНАННЯ. РОЗРАХУНОК РЕЗИСТОРА

На минулих заняттях ми підключали світлодіоди до Arduino, не думаючи про фізичні закони. Тепер треба ознайомитися з основними поняттями і законами, які діють в електричних колах [5, 6].

Будь-яку ділянку або елемент електричного кола можна охарактеризувати за допомогою трьох складників: струму, напруги та опору. В міжнародній системі одиниць СІ:

напруга вимірюється у вольтах: $[U] = B$;

опір – в омах: $[R] = \text{Ом}$;

сила струму – в амперах: $[I] = A$.

Якщо використовуються кратні одиниці вимірювань цих величин (наприклад міліампер, мілівольт, мегаом, кілоом і т. д.), то їх потрібно перевести відповідно в ампери, вольти та оми. Залежність між напругою, опором і силою струму називається **законом Ома.**

Закон Ома [6]: Величина сили струму на ділянці кола прямо пропорційна напрузі, прикладеній до цієї ділянки, та обернено пропорційна її опору:

$$I = \frac{U}{R}, \tag{1}$$

де I – сила струму; U – напруга на ділянці кола; R – опір провідника.

Трикутник Ома (рисунок 27).

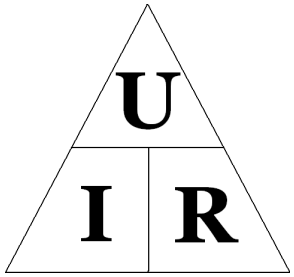


Рисунок 27

Як використовувати трикутник Ома: закриємо шукану величину – два інших символи дадуть формулу для її обчислення.

Якщо збільшиться напруга, то струм також збільшиться в таку саму кількість разів. Піднявши опір у колі, струм зменшиться в стільки ж разів, у скільки піднявся опір. Це можна легко зрозуміти на простому прикладі: беремо трубу і пропускаємо через неї потік води, чим вище один край труби (еквівалент різниці потенціалів), тим сильніше потік води (еквівалент сили струму), а якщо труба звузиться (еквівалент опору), то потік води зменшиться [6].

Закони послідовного з’єднання

Послідовне з’єднання – це з’єднання, в якому кожний окремий елемент з’єднується з іншими лише в одній точці [5] (рисунок 28).

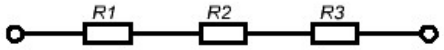


Рисунок 28

1-й закон послідовного з’єднання: *Сила струму через кожний елемент при послідовному з’єднанні дорівнює загальному струму, який проходить через усе коло* (рисунок 29).

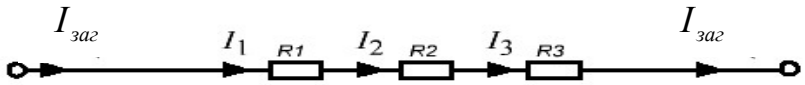


Рисунок 29

$$I_{заг} = I_1 = I_2 = I_3 . \qquad (2)$$

2-й закон послідовного з’єднання: *Загальна напруга (напруга на всьому колі) дорівнює сумі напруг на кожному елементі в послідовному з’єднанні:*

$$U_{заг} = U_1 + U_2 + U_3 . \qquad (3)$$

3-й закон послідовного з’єднання: *Загальний опір, тобто опір усього кола, дорівнює сумі опорів усіх елементів у послідовному з’єднанні:*

$$R_{заг} = R_1 + R_2 + ... + R_n . \qquad (4)$$

Закони паралельного з’єднання

Паралельне з’єднання – це з’єднання, при якому елементи з’єднуються між собою двома контактами (рисунок 30). У результаті до однієї точки (електричного вузла) може бути приєднано декілька елементів [5].

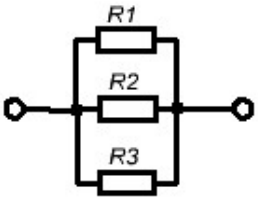


Рисунок 30

1-й закон паралельного з’єднання [6]: *Загальна сила струму (повний струм на всьому колі) дорівнює сумі струмів на кожному елементі в паралельному з’єднанні* (рисунок 31).

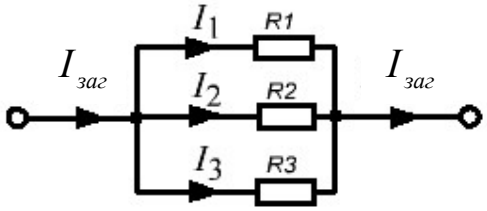


Рисунок 31

$$I_{заг} = I_1 + I_2 + I_3 . \qquad (5)$$

2-й закон паралельного з’єднання [6]: *Загальна напруга (напруга на всьому колі) дорівнює напрузі на кожному елементі в паралельному з’єднанні:*

$$U_{заг} = U_1 = U_2 = U_3 . \qquad (6)$$

3-й закон паралельного з’єднання [6]: *Загальна провідність (обернений опір), тобто провідність усього кола, дорівнює сумі провідностей усіх елементів у паралельному з’єднанні:*

$$\frac{1}{R_{заг}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} . \qquad (7)$$

Таким чином, при паралельному з'єднанні резисторів з різними опорами загальний опір буде завжди менше значення найменшого окремого резистора.

У випадку двох паралельно з'єднаних опорів із загальної формули легко отримати формулу [6]:

$$R_{заг} = \frac{R_1 \cdot R_2}{R_1 + R_2}. \qquad (8)$$

Розрахунок резистора

На підставі вивчених законів розрахуємо значення резистора для підключення світлодіода.

Однак спочатку потрібно визначити параметри наших світлодіодів. На сайті продавця або виробника ми дізнаємося, що характерна робоча напруга наших світлодіодів складає 2,2 В при струмі 20 мА (*міліампер*). Напруга на піні Arduino UNO при подачі на нього сигналу HIGH дорівнює 5 В. Тобто маємо таку задачу:

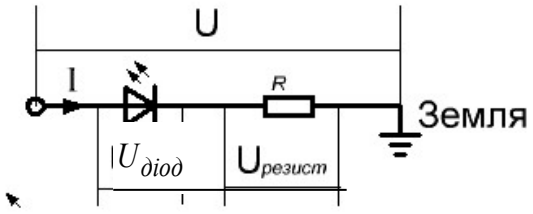
Дано

$U = 5\ В$

$I = 20\ мА = 0{,}02\ А$

$U_{діод} = 5\ В$

R -?



Розв'язок: Із схеми випливає, що резистор і світлодіод з'єднані *послідовно*.

Згідно з другим законом послідовного з'єднання, маємо:

$$U = U_{резист} + U_{діод}.$$

Знайдемо напругу на резисторі:

$$U_{резист} = U - U_{діод} = 5 - 2{,}2 = 2{,}8\ В.$$

Напишемо закон Ома для ділянки кола з резистором:

$$I = \frac{U_{резист}}{R},$$

тоді, опір резистора

$$R = \frac{U_{резист}}{I} = \frac{2{,}8}{0{,}02} = 140\ Ом.$$

Ми отримали, що розрахункове значення резистора має бути 140 Ом. Однак на практиці вибирають параметри з невеликим запасом у бік збільшення. Тобто використаний нами резистор на 220 Ом є цілком прийнятним.

Заняття 8. МАРКУВАННЯ РЕЗИСТОРІВ. ПОТУЖНІСТЬ. ГІРЛЯНДА З ШЕСТИ СВІТЛОДІОДІВ

Опір резистора – його основна характеристика. Основною одиницею електричного опору є Ом.

Зазвичай на схемах резистор позначається великою латинською літерою *R* і прямокутником. Іноді в іноземних схемах трапляється зображення резистора у вигляді «пилки» (рисунок 32).

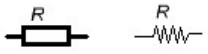
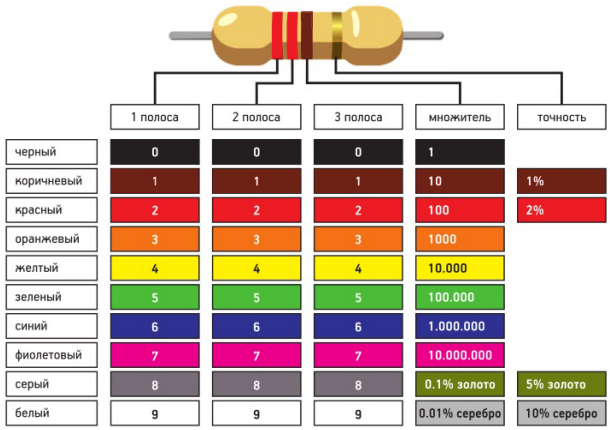


Рисунок 32

На минулих заняттях ми вже використовували резистор на 220 Ом. Однак його номінал (220 Ом) був просто озвучений, а чому саме цей резистор має таку характеристику, було нам не відомо. Тепер ми з'ясуємо, як визначати опір будь-якого резистора і чим вони відрізняються крім опору. Для визначення номіналу необхідно подивитися на кольорові смужки, нанесені на резистор (таблиця 12).

Таблиця 12



	1 полоса	2 полоса	3 полоса	множитель	точность
черный	0	0	0	1	
коричневый	1	1	1	10	1%
красный	2	2	2	100	2%
оранжевый	3	3	3	1000	
желтый	4	4	4	10.000	
зеленый	5	5	5	100.000	
синий	6	6	6	1.000.000	
фиолетовый	7	7	7	10.000.000	
серый	8	8	8	0.1% золото	5% золото
белый	9	9	9	0.01% серебро	10% серебро

Знаходимо на резисторі золоту смужку, розташовуємо його так, щоб вона була справа (рисунок 33).

Рисунок 33

Бачимо, що, окрім золотої смужки, є ще три (можливий варіант – чотири). Послідовність трьох смужок зліва направо така: *червона, червона, коричнева*.

Дивимося таблицю 12: перша червона смужка відповідає цифрі «2», друга – цифрі «2», а третя, коричнева, – множнику «10». Таким чином, маємо таку комбінацію:

$$22 \times 10,$$

яка відповідає значенню 220 Ом.

Золота смужка означає точність виготовлення резистора, тобто 5 %. Це відповідає $220 \times 0{,}05 = 11$ Ом, тобто наш резистор може бути в межах від 220 - 11 до 220 + 11 Ом, іншими словами, опір резистора лежить від 209 до 231 Ом.

Якби у нас був резистор з чотирма смужками, то третя смужка відповідала б цифрі, а не множнику. Іноді використовується такий резистор (рисунок 34).

Рисунок 34

Цей резистор, згідно з таблицею 12, має чотири лінії, які визначають його номінал: червона, червона, чорна, чорна. І одна лінія відповідає точності – коричнева. Тобто отримуємо такі цифри: «2», «2», «0», «1».

Остаточно маємо $22 \times 1 = 220$ Ом. Точність 1 %, тобто опір резистора перебуває в межах від 217,8 до 222,2 Ом.

Завдання: розрахувати номінал резистора (рисунок 35).

Рисунок 35

Коричнева лінія – цифра «1», чорна – цифра «0», помаранчева – множник «1000», разом отримуємо:

$$10 \times 1000 = 10000 \text{ Ом} = 10 \text{ кОм}.$$

Точність 5 %, тобто 500 Ом. Остаточно маємо резистор опором від 9500 Ом до 10500 Ом або від 9,5 кОм до 10,5 кОм.

Наступна, дуже важлива, характеристика – потужність резистора.

Потужність вимірюється у ватах і розраховується за однією з формул:

$$P = U \cdot I = I^2 \cdot R = \frac{U^2}{R}. \qquad (9)$$

Визначимо потужність, яка розсіюється на резисторі (заняття 7). Нам відомо, що падіння напруги на резисторі дорівнює 2,8 В, сила струму через резистор становить 20 мА, опір резистора є 140 Ом. Розрахунок за кожною із трьох наведених формул дає такий результат:

$$P = 2{,}8 \cdot 0{,}02 = 0{,}02^2 \cdot 140 = \frac{2{,}8^2}{140} = 0{,}056 \text{ Вт} = 56 \text{ мВт}.$$

Використовуваний нами резистор розрахований на потужність до 250 мВт, тобто він повністю прийнятний, оскільки завжди потрібно вибирати резистор за потужністю, більшою за ту, що йому доведеться розсіювати. Неправильно підібраний резистор за потужністю може бути причиною пожежі.

На принципових електричних схемах на потужність резисторів вказує спеціальний значок (рисунок 36).

Рисунок 36

Чим більшу розсіювальну потужність має резистор, тим він більше за розмірами (габаритами).

Гірлянда з шести світлодіодів

Наприкінці заняття складемо схему для керування шістьма світлодіодами і створимо ефект рухомої змійки.

Принципова електрична схема подібна до схеми «світлофора» (заняття 6), тому наведемо лише монтажну схему з'єднання (рисунок 37).

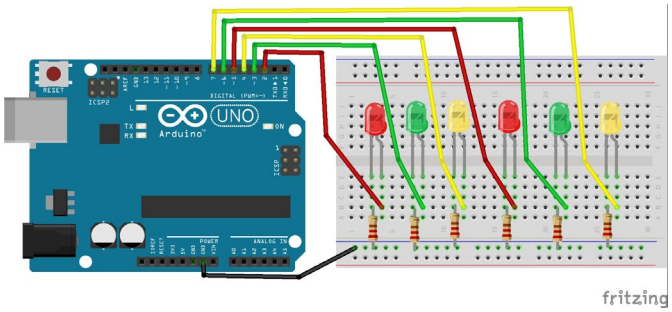


Рисунок 37

Prog8

```
void setup() {
 pinMode(2, OUTPUT); // ініціалізація 2-го піна
 pinMode(3, OUTPUT); // ініціалізація 3-го піна
 pinMode(4, OUTPUT); // ініціалізація 4-го піна
 pinMode(5, OUTPUT); // ініціалізація 5-го піна
 pinMode(6, OUTPUT); // ініціалізація 6-го піна
 pinMode(7, OUTPUT); // ініціалізація 6-го піна
 digitalWrite(2, LOW); // вимикаємо світлодіоди
 digitalWrite(3, LOW);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, LOW);
 digitalWrite(7, LOW);
 }
void loop() {
 digitalWrite(2, HIGH); // послідовно вмикаємо
 delay(200);      // всі світлодіоди
 digitalWrite(3, HIGH);
 delay(200);
 digitalWrite(4, HIGH);
 delay(200);
 digitalWrite(5, HIGH);
 delay(200);
 digitalWrite(6, HIGH);
 delay(200);
 digitalWrite(7, HIGH);
 delay(200);
 digitalWrite(2, LOW); // послідовно вимикаємо
 delay(200);       // всі світлодіоди
 digitalWrite(3, LOW);
 delay(200);
 digitalWrite(4, LOW);
 delay(200);
 digitalWrite(5, LOW);
 delay(200);
 digitalWrite(6, LOW);
```

```
 delay(200);
 digitalWrite(7, LOW);
 delay(200);
}
```

Завдання: написати програму, в якій світлодіоди світяться по черзі в один бік, а гаснуть – в інший.

Заняття 9. SERIAL MONITOR (МОНІТОР ПОСЛІДОВНОГО З'ЄДНАННЯ)

Дуже часто при налагодженні або при роботі програм виникає потреба передачі даних з Arduino в персональний комп'ютер. Для цього використовують монітор послідовного з'єднання (**Serial Monitor).**

Для роботи з **Serial Monitor** його необхідно *ініціалізувати* (встановити).

Для цього застосовується команда

Serial.begin (baud_rate);

де baud_rate – швидкість, з якою будуть спілкуватися Arduino і ПК. Ця швидкість вимірюється в

Бод (baud)

Для двійкового кодування Бод = біт / секунду.

Стандартними значеннями швидкості є: 300, 1200, 2400, 4800, **9600**, 19200, 38400, 57600, 74880, **115200**, 230400, 250000.

Для передачі даних існують команди:

Serial.print(); виводить символи на екран

Serial.println(); виводить символи на екран і переводить курсор на новий рядок

Ці команди можна використовувати як у функції **setup,** так і в **loop.**

Наберіть подані нижче програми і опишіть різницю в їхній роботі.

Prog9

```
void setup() {
 Serial.begin(9600); // підключаємо монітор порту
 Serial.println("Start"); // виводимо текст на монітор 1 раз
```

```
}
void loop() {}
```

Prog9a

```
void setup() {
 Serial.begin(9600); // підключаємо монітор порту
}
void loop() {
 Serial.print("Start"); // змінюємо рядок на Serial.println
("Start");
 delay(1000); // затримка в 1 секунду
}
```

Після того як завантажили скетч у плату Arduino, включаємо **Serial Monitor.**

Для цього використовуємо або меню, або гарячі клавіші *Ctrl* + *Shift* + *M* (рисунок 38),

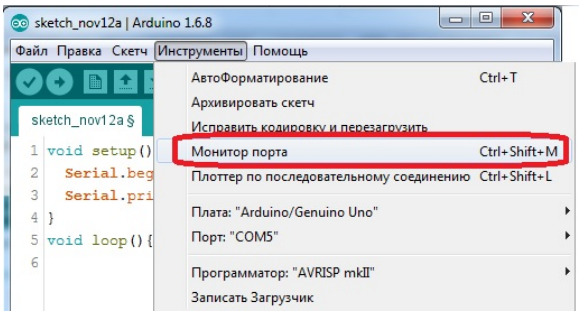


Рисунок 38

або кнопку (рисунок 39).

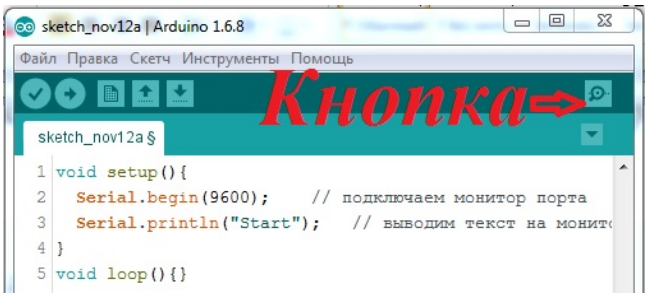


Рисунок 39

Також необхідно встановити відповідну швидкість з'єднання в моніторі порту. Виведення чисел у Serial Monitor

```
void setup() {
 Serial.begin(9600); // підключаємо монітор порту
}
void loop() {
 Serial.println(78);    // виводимо число 78 у монітор
 delay(1000);           // затримка в 1 секунду
}
```

Змінюємо п'ятий рядок на:

```
 Serial.println(78, BIN); 1001110 двійковий код
 Serial.println(78, OCT); 116 вісімковий код
 Serial.println(78, DEC); 78 десятковий код
 Serial.println(78, HEX); 4E шістнадцятковий код
```

Бачимо на екрані (можемо перевірити на калькуляторі).

Дрібні числа в **Serial Monitor**

```
void setup() {
 Serial.begin(9600);           // підключаємо монітор порту
}
void loop() {
 Serial.println(781.34567);
 delay(1000);                  // затримка в 1 секунду
}
```

Бачимо, що на екрані виводиться число **781.35.** Тобто відбувається округлення до сотих.

Вказуємо інший формат виведення (вказується число цифр після коми)

```
 Serial.println(781.34567, 0);
```

або

```
 Serial.println(781.34567, 1);
```

або

```
 Serial.println(781.34567, 2);
```

або

```
Serial.println(781.34567, 4);
```

Завдання: вивести один раз у монітор порту число 3642 у двійковій, вісімковій, десятковій і шістнадцятковій системах у вигляді, зображеному на рисунку 40.

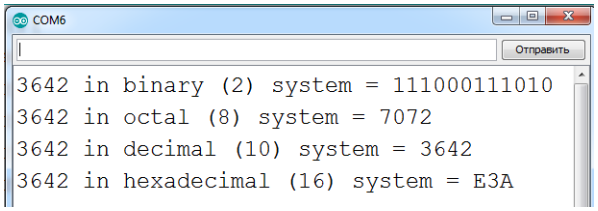


Рисунок 40

Відповідь:_____ .

```
void setup() {
 Serial.begin(9600);
 Serial.print( "3642 in binary (2) system = ");
 Serial.println(3642, BIN);
 Serial.print( "3642 in octal (8) system = ");
 Serial.println(3642, OCT);
 Serial.print( "3642 in decimal (10) system = ");
 Serial.println(3642, DEC);
 Serial.print( "3642 in hexadecimal (16) system = ");
 Serial.println(3642, HEX);
}
void loop() {}
```

Заняття 10. ЗМІННІ. ТИПИ ЗМІННИХ. ОБЛАСТЬ ВИДИМОСТІ ЗМІННИХ. ВИХІД ЗА МЕЖІ ДОПУСТИМОГО ДІАПАЗОНУ

Змінні

У попередньому прикладі нам знадобилося число 3642 увести вісім разів. Якщо тепер нам потрібно зробити те саме з іншим числом, то знову знадобиться його ввести вісім разів. Це дуже нераціонально. Для оптимізації коду використовують змінні.

Змінні бувають різних типів. Перший тип змінних, з якими ми ознайомимося, – це цілий тип змінних **int.** Кожна змінна такого типу займає в пам'яті комп'ютера 2 байта, тобто 16 біт.

Будь-яка змінна повинна мати своє унікальне ім'я, яке задає програміст. Це ім'я вказується при оголошенні змінної, а далі в потрібному місці програми звернення до змінної відбувається на її ім'я. Оголосимо змінну типу **int** з ім'ям **value** і дамо їй значення 3642:

int value = 3642;

Увага:

1) в кінці рядка ставиться крапка з комою;

2) *при оголошенні змінної необов'язково відразу привласнювати їй значення.*

Перепишемо попередню програму, але будемо використовувати змінну.

Prog10

```
int value = 3642;
void setup() {
 Serial.begin(9600);
 Serial.print(value);
 Serial.print( " in binary (2) system = ");
 Serial.println(value, BIN);

 Serial.print(value);
 Serial.print( " in octal (8) system = ");
 Serial.println(value, OCT);

 Serial.print(value);
 Serial.print( "In decimal (10) system = ");
 Serial.println(value, DEC);

 Serial.print(value);
 Serial.print( " in hexadecimal (16) system = ");
 Serial.println(value, HEX);
}
void loop() {}
```

Перевірте, який вигляд мають такі числа в різних системах: 3, 6, 7, 9, 10, 15, 16.

З’ясуємо,що таке змінна.

Змінна – це місце зберігання даних. Вона має:

ім'я, значення і тип.

Наприклад, дане оголошення (воно називається декларацією):

```
int
pin = 13;
```

створює змінну з ім'ям **pin**, значенням **13** і типом **int**.

Типи змінних

Нижче подано список основних типів даних, які використовуються в скетчах Arduino. Поруч з кожним типом даних вказано його розмір. Зверніть увагу, що змінні типу **signed** дають можливість оперувати додатними і від’ємними числами, а змінні типу **unsigned** допускають роботу тільки з невід'ємними значеннями.

boolean або **bool**

Першим ми розглянемо тип **boolean**. Змінні цього типу можуть набувати лише двох значень: **true** або **false**; **HIGH** або **LOW**; **1** або **0**. У пам'яті така змінна займає 8 біт, тобто 1 байт

byte

Наступний тип змінної, який займає також 1 байт, є **byte**. Оскільки 1 байт = 8 біт, то максимально можливе число значень, якого може набувати така змінна, є 256.

Оскільки мінімальне значення такої змінної дорівнює нулю, то діапазон набутих значень буде від 0 до 255.

char

Змінна типу **char** також займає в пам'яті 1 байт, а отже, може набувати 256 різних значень. Однак у цьому випадку інтервал значень ділиться на 128 додатних чисел (разом із нулем) і на 128 від’ємних значень. У тому числі значення змінної типу **char** лежить у діапазоні від -128 до +127. У деяких випадках компілятор буде інтерпретувати цей тип даних як символ, що може призвести до несподіваних результатів.

int

Уже знайомий нам тип змінної **int,** тобто integer. Змінна займає в пам'яті 16 біт, тобто 2 байти. Отже, може набувати

$$2^{16}=65536$$

різних значень. Ця кількість значень ділиться навпіл: 32768 додатних чисел (разом з нулем), 32768 від'ємних чисел.

Діапазон прийнятих значень для типу **int** є від -32768 до 32767.

word

Змінна типу **word** займає в пам'яті також 16 біт. Отже, має також $2^{16}=65536$ різних значень. Цей тип змінної є беззнаковим (**unsigned**), тобто не містить від'ємних чисел, отже, діапазон значень є від 0 до 65536.

long

Тип **long –** це 32-бітний знаковий тип змінних. Отже, кількість набутих значень

$$2^{32}=4294967296$$

ділиться навпіл на додатні з нулем і від'ємні. Діапазон набутих значень буде від -2147483648 до +2147483648

unsigned long

Тип **unsigned long** це 32-бітний беззнаковий тип змінних. Отже, кількість значень є також

$$2^{32}=4294967296,$$

а діапазон значень – від 0 до + 4294967295.

float

Тип **float –** це так званий дійсний тип змінної. Тобто може бути як цілим, так і нецілим числом, а також набувати додатних і від'ємних значень. У пам'яті комп'ютера займає 32 біта. Значення лежить у діапазоні від $-3.4028235\text{x}10^{38}$ до $3.4028235\text{x}10^{38}$. Такі числа називаються числами з плаваючою комою і не характерні для Arduino IDE, тому компілятор витратить значно більше часу на їх обробку. Отже, якщо швидкодія роботи програми є істотною, то цей тип змінних рекомендується не використовувати.

Операції зі змінними

Найбільш часто використовувана операція – це

= (Привласнення)

При цьому вираз зліва набуває значення від виразу справа. Наприклад:

$x = 20$

записує у змінну *x* число 20.

Стандартні та добре зрозумілі операції:

+ (Додавання)
- (Віднімання)
***** (Множення)
/ (Ділення)

Коментар щодо поділу. Якщо ми використовуємо цілі числа, то

7/3 = 2, 19/10 = 1,

тобто дрібна частина просто відкидається.

Менш очевидна, але дуже корисна операція

% (*остача від ділення*).

Приклад: 12 % від 10 дає результат 2.

Зауважимо, що операція **присвоєння** – це не математична операція **дорівнює**. Так, з точки зору математики запис $x = x* 2$ – є безглуздим, проте після застосування операції присвоєння результат буде залежати від попереднього значення змінної *x*. Так, якщо до виконання операції *x* дорівнює 3, то після виконання операції значення змінної *x* дорівнюватиме 6.

Область видимості змінних

Змінні в мові програмування C++, що використовується Arduino, мають властивість, яка називається область видимості, на відміну від мов, подібних до BASIC, PASCAL, в яких кожна змінна є глобальною. Глобальна змінна доступна для будь-якої функції в будь-якому місці програми. Локальні змінні «видно» тільки в тій функції, в якій вони оголошені. У середовищі Arduino IDE будь-яка змінна, оголошена поза функцією setup() або loop(), є глобальною змінною.

Вихід за межі допустимого діапазону

Бувають ситуації, коли в процесі роботи програми значення змінної може вийти з допустимого діапазону. Виникнення такої ситуації вкрай небажано, тому що може призвести до непередбачуваних наслідків.

Напишемо скетч, в якому оголосимо дві змінні типу **byte** і будемо виводити в монітор послідовного порту (Serial Monitor) різницю цих двох чисел.

Prog10a

```
byte x = 17;
byte y = 13;
byte z;
void setup() { // ініціалізація послідовного з'єднання
 Serial.begin(9600); // на швидкості 9600 бод
 z = x-y;
 Serial.println(Z); // виводимо результат операції один раз
}
void loop() {} // навіть невикористовувана функція loop
                // має бути оголошена
```

Відкриваючи монітор порту, бачимо відповідь: 4, тобто все, як і належить.

Але що буде, якщо ми обчислимо y – x? Змінимо рядок № 6 на

```
z = y - x;
```

і отримаємо відповідь: 252. Чому саме так, адже має вийти «-4»?

Допустимий діапазон значень для типу **byte** є 0..255. Уявімо ці числа як циферблат (рисунок 41).

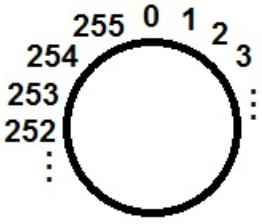


Рисунок 41

Число «-4» міститься зліва від нуля на 4-й позиції, тобто отримуємо 252.

Наведемо приклад ще однієї програми, в якій вихід з діапазону відбудеться не відразу, а в процесі роботи скетчу.

Prog10б

```
byte x = 2;
byte y = 3;
void setup() {
 Serial.begin(9600);
}
```

```
void loop() {
 y=x+y;
 Serial.println(Y);
 delay(200);
}
```

Тут ми бачимо, як в один момент отримаємо результат:

$$255 + 2 = 1,$$

що повністю відповідає міркуванням, наведеним вище.

КОНТРОЛЬНІ ПИТАННЯ

1 Яка відмінність платформи Arduino від Raspberry Pi?

2 Яка відмінність мікроконтролера від звичайної мікросхеми?

3 Навіщо потрібні шилди?

4 Яку платформу Arduino ви використовуєте?

5 Як називається середовище розробки, в якому ви пишете програми?

6 Із яких двох функцій завжди складається програма для Arduino?

7 Яка відмінність виходу від входу мікроконтролера?

8 Як називаються ніжки світлодіода?

9 Яка полярність підключення світлодіода?

10 Навіщо потрібен резистор при підключенні світлодіода?

11 Як влаштована макетна плата і навіщо вона потрібна?

12 Сформулюйте закон Ома.

13 Накресліть послідовно ввімкненими чотири резистори.

14 Як розрахувати струм, напругу і опір при послідовному з’єднанні?

15 Накресліть паралельно з’єднаними чотири резистори.

16 Як розрахувати струм, напругу і опір при паралельному з’єднанні?

ВІДПОВІДІ НА ПИТАННЯ

1 Яка відмінність платформи Arduino від Raspberry Pi?

Raspberry Pi в 40 разів швидша за Arduino. Ще більша відмінність в оперативній пам’яті: Raspberry Pi має в 128000 разів більше оперативної пам’яті, ніж Arduino. Raspberry Pi є комп'ютером, на якому може бути запущена операційна система Linux, яка підтримує багатозадачність. До USB-портів можна підключати різні прилади, наприклад для безпроводового підключення до мережі Інтернет. Arduino здатна краще, ніж Raspberry Pi, і дійсно в реальному часі зчитувати аналогові сигнали. Ця гнучкість дає змогу Arduino працювати майже з будь-яким видом датчиків або чіпів. Raspberry Pi не така гнучка, наприклад, для читання аналогових датчиків потрібні додаткові апаратні засоби. Arduino менш вибаглива до живлення. Так, рекомендоване живлення для Arduino UNO 7–12 В, напруга стабілізується до 5 В. А плата Raspberry Pi потребує строго 5 В на вході, тому для роботи з нею не обійтись без фільтра живлення зі струмом 1 А. Arduino працює з будь-яким комп’ютером і може працювати від батареї. Arduino можна вмикати і вимикати в будь-який час. Операційна система на Raspberry Pi може бути пошкоджена, якщо відключити плату без належного завершення роботи.

2 Яка відмінність мікроконтролера від звичайної мікросхеми?

Мікроконтролер — це самостійна комп'ютерна система, яка містить: процесор, допоміжні схеми і пристрої вводу-виводу даних, розміщені в загальному корпусі. Мікроконтролер ми програмуємо, а сигнали отримуємо відповідно до закладеної програми, а не конструкції, як у випадку з мікросхемою.

3 Навіщо потрібні шилди?

Для збільшення можливостей платформ Arduino.

4 Яку платформу Arduino ви використовуєте?

UNO, Nano.

5 Як називається середовище розробки, в якому ви пишете програми?

Arduino IDE.

6 Із яких двох функцій завжди складається програма для Arduino?

Setup і loop

7 Яка відмінність виходу від входу мікроконтролера?

Якщо порт налаштований як вихід, то на ньому встановлюється логічна «1», або логічний «0» за допомогою написаної програми (тобто, фактично, за допомогою мікроконтролера). Якщо порт налаштований як вхід, то на ньому встановлюється логічна «1», або логічний «0» за допомогою зовнішнього пристрою, наприклад датчика.

8 Як називаються ніжки світлодіода?

Анод і катод.

9 Яка полярність підключення світлодіода?

Анод – «плюс», а катод – «мінус».

10 Навіщо потрібен резистор при підключенні світлодіода?

Для обмеження струму. При підключенні світлодіода до 5 В напряму, без резистора, через нього проходитиме великий струм (більше 20 мА), який виведе пристрій із ладу.

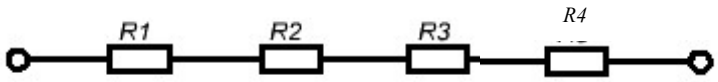
11 Як влаштована макетна плата і навіщо вона потрібна?

Дві пари шин живлення по краях і провідники. Вона потрібна для того, щоб конструювати та налагоджувати прототипи різних пристроїв без паяння.

12 Сформулюйте закон Ома.

Величина сили струму на ділянці кола прямо пропорційна напрузі, прикладеній до цієї ділянки, і обернено пропорційна його опору.

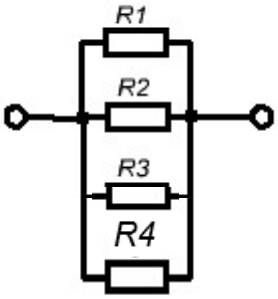
13 Накресліть послідовно з’єднаними чотири резистори.



14 Як розрахувати струм, напругу і опір при послідовному з’єднанні?

$$I_{заг} = I_1 = I_2 = I_3 = I_4;$$
$$U_{заг} = U_1 + U_2 + U_3 + U_4;$$
$$R_{заг} = R_1 + R_2 + R_3 + R_4.$$

15 Накресліть паралельно з’єднаними чотири резистори.



16 Як розрахувати струм, напругу і опір при паралельному з’єднанні?

$$I_{заг} = I_1 + I_2 + I_3 + I_4,$$
$$U_{заг} = U_1 = U_2 = U_3 = U_4,$$
$$\frac{1}{R_{заг}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}.$$

СПИСОК ЛІТЕРАТУРИ

1 Jeremy Blum. Exploring Arduino: Tools and Techniques for Engineering Wizardry, ISBN 978-1-118-54936-0. 2013 by John Wiley & Sons, Inc. P. 357.

2 Julien Bayle. C Programming for Arduino. Learn how to program and use Arduino boards with a series of engaging examples, illustrating each core concept. First published: May 2013. Production Reference: 1070513. Published by Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK. ISBN 978-1-84951-758-4. P. 512.

3 Ляшенко О., Мартинюк О. Моделювання та дослідження електронних пристроїв: навч. посіб. Луцьк: Східноєвроп. нац. ун-т ім. Лесі Українки, 2013. 217 с.

4 Соменко Д. В. Використання апаратно-обчислювальної платформи Arduino в навчальному процесі з фізики. *Наукові записки*. Серія: Проблеми методики фізико-математичної і технологічної освіти. 2013. № 9, С. 173-182.

5 Кучерук І. М., Горбачук І. Т., Луцик П. П. Загальний курс фізики. Том 2: Електрика і магнетизм / за ред. І. М. Кучерука. Київ: Техніка, 2001. 452с.

6 Котвицький А. Т., Гресь В. Ю., Котвицька К. А. Методичні вказівки до контрольних робіт з фізики № 1, 2 «Механіка. Молекулярна фізика і термодинаміка. Електростатика і постійний струм». Харків: УкрДУЗТ, 2017. 88 с.

МЕТОДИЧНІ РОЗРОБКИ

З ВИКОРИСТАННЯ ПЛАТФОРМИ ARDUINO
У НАВЧАЛЬНОМУ ПРОЦЕСІ

Відповідальний за випуск Котвицький А. Т.

Редактор Буранова Н. В.

Підписано до друку 16.03.21 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 4,5. Тираж 5. Замовлення №

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.