МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УКРАЇНСЬКИЙ ДЕРЖАВНИЙ ХІМІКО-ТЕХНОЛОГІЧНИЙ
УНІВЕРСИТЕТ»

# МЕТОДИЧНІ ВКАЗІВКИ
# ДО ПРАКТИЧНИХ ЗАНЯТЬ
# З ДИСЦИПЛІНИ «ЕЛЕКТРОННІ ПРИСТРОЇ АВТОМАТИКИ»
# ЗА ОСВІТНІМ РІВНЕМ «БАКАЛАВР» ДЛЯ СТУДЕНТІВ
# СПЕЦІАЛЬНОСТІ «151 АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ»

Затверджено на засіданні кафедри
Комп'ютерно-інтегрованих технологій
та автоматизації.
Протокол № 2 від 22.11.18

Дніпро ДВНЗ УДХТУ 2019

Методичні вказівки до практичних занять з дисципліни «Електронні пристрої автоматики» за освітнім рівнем «Бакалавр» для студентів спеціальності «151 Автоматизація та комп'ютерно-інтегровані технології» / Укл. О.П. Мисов, М.О. Савченко – Дніпро: ДВНЗ УДХТУ, 2019. – 25 с.

Укладачі: О.П. Мисов, канд. техн. наук
М.О. Савченко, канд. техн. наук

Відповідальний за випуск О.П. Мисов, канд. техн. наук

Навчальне видання
Методичні вказівки
до практичних занять з дисципліни «Електронні пристрої автоматики» за освітнім рівнем «Бакалавр» для студентів спеціальності «151 Автоматизація та комп'ютерно-інтегровані технології»

Укладачі: МИСОВ Олег Петрович
САВЧЕНКО Марія Олегівна

Комп'ютерна верстка Т.М. Кіжло

Підписано до друку 27.03.19. Формат 60×84/16. Папір ксерокс. Друк різограф. Умов. друк. арк. 1,04 Обл.-вид. арк. 1,11. Тираж 100 прим. Зам. № 195. Свідоцтво ДК № 5026 від 16.12.2015

ДВНЗ УДХТУ, просп. Гагаріна, 8, м. Дніпро, 49005

Редакційно-видавничий відділ

**ЗМІСТ**

# ЗАГАЛЬНІ ВКАЗІВКИ

Ці вказівки рекомендуються в якості методичного матеріалу при роботі на практичних заняттях за курсом «Електронні пристрої автоматики». В доповнення до вказівок припускається використання спеціальної літератури, перелік якої додається.

Тематика практичних занять охоплює наступні розділи програми курсу «Електронні пристрої автоматики»:

1 Робота з цифровими виходами і входами Arduino. Тест – програма Arduino

2 Використання широтно-імпульсної модуляції Arduino. Робота з COM-портом.

3 Побудова практичних вимірювальних схем на базі Arduino Uno

4 Застосування мови Python в практичних схемах автоматизації с використанням Arduino

Завдання на практичні заняття складаються таким чином, щоб найбільш повно уявити перераховані вище розділи в рамках одного заняття і завдяки цьому поширити практичні навички студентів при розрахунку і проектуванні електронних схем. Роботу на практичному занятті слід починати з ознайомлення із завданням. Після цього зробити підбір, вивчення, огляд і аналіз аналогічних схем та вибір структурної схеми приладу, накреслити принципову електричну схему приладу, виконати розрахунки, підбір елементів і скласти їх перелік.

Виконане завдання містить: завдання на проектування, структурну схему пристрою (при необхідності), принципову електричну схему і її обґрунтування, електричні, графічні, графоаналітичні розрахунки (вибір типів приладів, призначення і розрахунок всіх елементів схеми, розрахунок параметрів, що визначають роботу схеми, тощо), перелік елементів.

При оформленні розрахункового завдання слід наводити розрахункові формули, а після цього підставляти в них цифрові значення вхідних параметрів. Отримані при розрахунках значення величин опору резисторів і ємностей конденсаторів необхідно округлити до номінальних згідно зі шкалою стандартних значень цих елементів. При складанні переліку елементів необхідно вибрати по довіднику типи резисторів та ємностей.

Схеми виконуються відповідно до ГОСТ 2.730-73, 2.721-68, 2.723-68, 2.728-74, 2.755-84, 2.759-82 та ін.

# 1. ВСТУП В ARDUINO

Будь-який електронний продукт, який потребує обчислення або взаємодії з іншими комп'ютерами, спочатку вимагає швидкої перевірки концепції з використанням простих інструментів. Arduino – це платформа для прототипів з відкритим вихідним кодом, розроблена на основі популярного сімейства мікроконтролерів, і включає в себе просте середовище розробки програмного забезпечення. Крім перевірки, ви також можливо використовувати Arduino для розробки власних проектів (DIY). Arduino з'єднує обчислювальний світ з фізичним світом, дозволяючи просто підключати датчики і приводи до комп'ютера. На загал, ви можете написати код для моніторингу та управління різними електронними компонентами в повсякденному житті за допомогою входів/виходів контактів Arduino і мікроконтролера. Прикладами цих компонентів можуть бути двигуни, термостати, світлові пристрої, перемикачі та багато іншого.

## 1.2. Історія

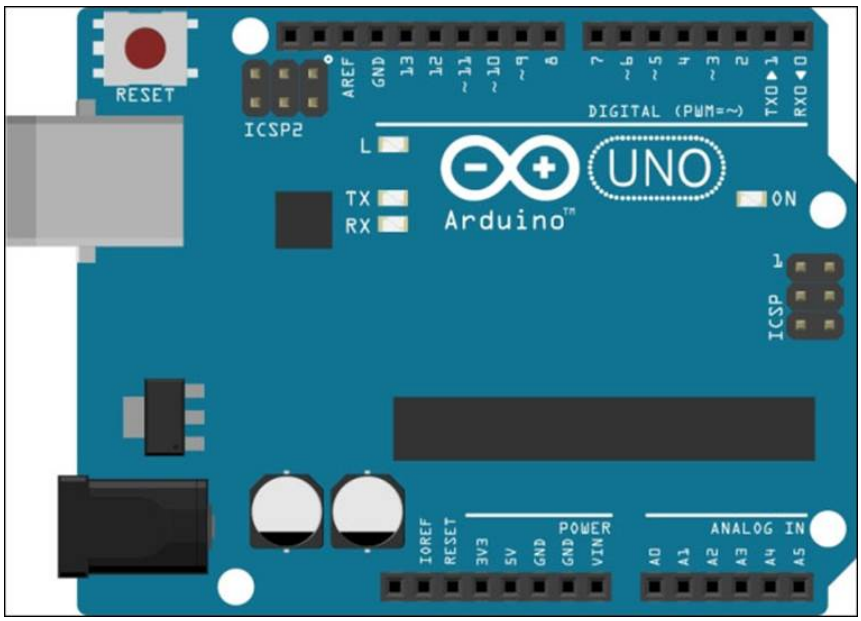
У 2005 році Массімо Банзі, італійський співзасновник Arduino, розробив технологію для своїх студентів в Інституті дизайну взаємодії Ivrea (IDII). З того часу Arduino перетворилася в одну з найбільших апаратних платформ з відкритим вихідним кодом. Всі програмні компоненти і схеми дизайну Arduino мають відкритий вихідний код, і ви можете купити обладнання за дуже низькою ціною, або навіть можете зробити їх самостійно.

## 1.3. Переваги Arduino

Основний напрямок спільноти Arduino – постійно вдосконалювати платформу Arduino з огляду на таке:

- платформа Arduino повинна бути доступною;
- вона повинна бути проста у використанні і легко кодуватися;
- відкрита вихідна і розширювана програмна платформа;
- відкрита вихідна і розширювана апаратна платформа;
- вона повинна мати проекти DIY, підтримувані спільнотою.

Ці прості, але потужні напрямки зробили Arduino популярної і широко використовуваною платформою для створення прототипів. Arduino використовує мікроконтролери Atmel серії ATmega, засновані на популярній апаратної архітектурі AVR. Величезна підтримка, яка доступна для архітектури AVR, також робить Arduino кращою апаратною платформою. На наступному малюнку показана базова версія плати Arduino, яка називається Arduino Uno (Uno означає один італійською мовою):




### 1.4. Варіанти Arduino

Як і будь-який інший проект, вимоги до обладнання визначаються специфікаціями проекту. Якщо ви розробляєте проект, який вимагає від вас взаємодії з великою кількістю зовнішніх компонентів, вам потрібна платформа прототипування, у якій є достатня кількість **входів/виходів (I/O)** для сполучення. Якщо ви працюєте над проектом, який повинен виконувати величезну кількість складних обчислень, вам потрібна платформа з великими обчислювальними можливостями.

На щастя, плата Arduino існує в 16 різних офіційних версіях, і кожна версія Arduino відрізняється від інших по формфактору, обчислювальній потужності, виводів введення-виведення і іншим вбудованим функціям. Arduino Uno – це основна і найпопулярніша версія, якої достатньо для простих проектів DIY. Для більшості вправ в цьому посібнику ми будемо використовувати плату Arduino Uno. Ви також можете використовувати інший популярний варіант під назвою Arduino Mega, який представляє більшу плату з додатковими виводами і потужним мікроконтролером. У наступній таблиці показано порівняння деяких найбільш популярних і активних варіантів плати Arduino:

| Назва | Процесор | Частота процесора | Кількість цифрових I/O | Кількість цифровихI/OзPWM | Кількість аналогових I/O |
|---|---|---|---|---|---|
| LilyPad Arduino | ATmega168v чи ATmega328v | 8 MHz | 14 | 6 | 6 |
| Arduino Uno | ATmega328 | 16 MHz | 14 | 6 | 6 |
| Arduino Leonardo | ATmega32u4 | 16 MHz | 14 | 6 | 12 |
| Arduino Mega | ATmega2560 | 16 MHz | 54 | 14 | 16 |
| Arduino Nano | ATmega328 | 16 MHz | 14 | 6 | 8 |
| Arduino Due | AT91SAM3X8E | 84 MHz | 54 | 12 | 12 |

Будь-який з цих варіантів може бути запрограмований з використанням загального інтегрованого середовища розробки під назвою Arduino IDE, яка описана в наступному розділі. Ви можете вибрати будь-яку з цих плат Arduino відповідно до ваших проектних вимог, а в Arduino IDE повинна бути можливість компілювати і завантажувати програму на плату.

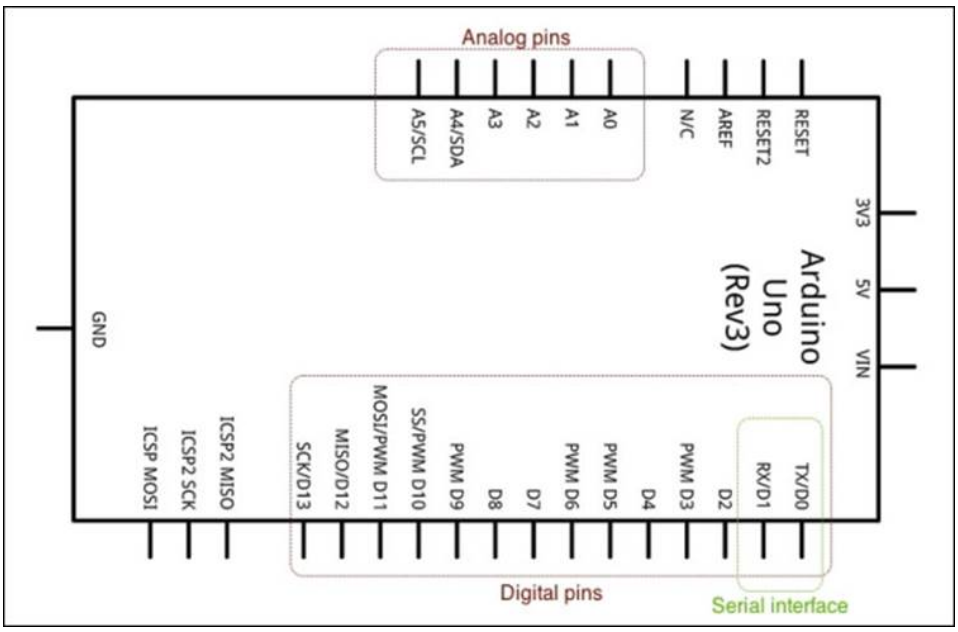
### 1.5.Описание контроллера Arduino

Остання версія плати Uno заснована на мікроконтролері Atmel ATmega328. Плата розширює виводи I/O мікроконтролера до периферійного пристрою, які потім можуть використовуватися для підключення компонентів з використанням проводів.

Плата має в цілому 20 контактів для інтерфейсу, з яких 14 є цифровими контактами введення/виведення і 6 є аналоговими вхідними контактами. З 14 цифрових контактів введення/виведення 6 контактів також підтримують широтно-імпульсну модуляцію (PWM) для підтримки контрольованої подачі живлення підключеним компонентам.

Плата працює на напрузі 5 В. Максимальний номінальний струм для цифрових входів/виходів становить 40 мА, що досить для приводу більшості електронних компонентів DIY, за винятком двигунів з високими вимогами до струму.

Наступна діаграма  та Мал.1. описує контакти на платі Uno. Як ви можете бачити, цифрові контакти розташовані на одній стороні плати, в той час як аналогові контакти знаходяться на протилежному боці. На платі також є пара силових контактів, які можуть використовуватися для забезпечення 5 В і 3,3 В потужності для зовнішніх компонентів. Плата також містить виводи заземлення з обох сторін плати. Ми будемо в основному використовувати контакти 5 В для наших проектів. Цифрові контакти **D0** і **D1** підтримують послідовний інтерфейс через інтерфейси **Tx (передача)** і **Rx (приймач)** відповідно. Порт USB на платі можна використовувати для підключення Arduino до комп'ютера.



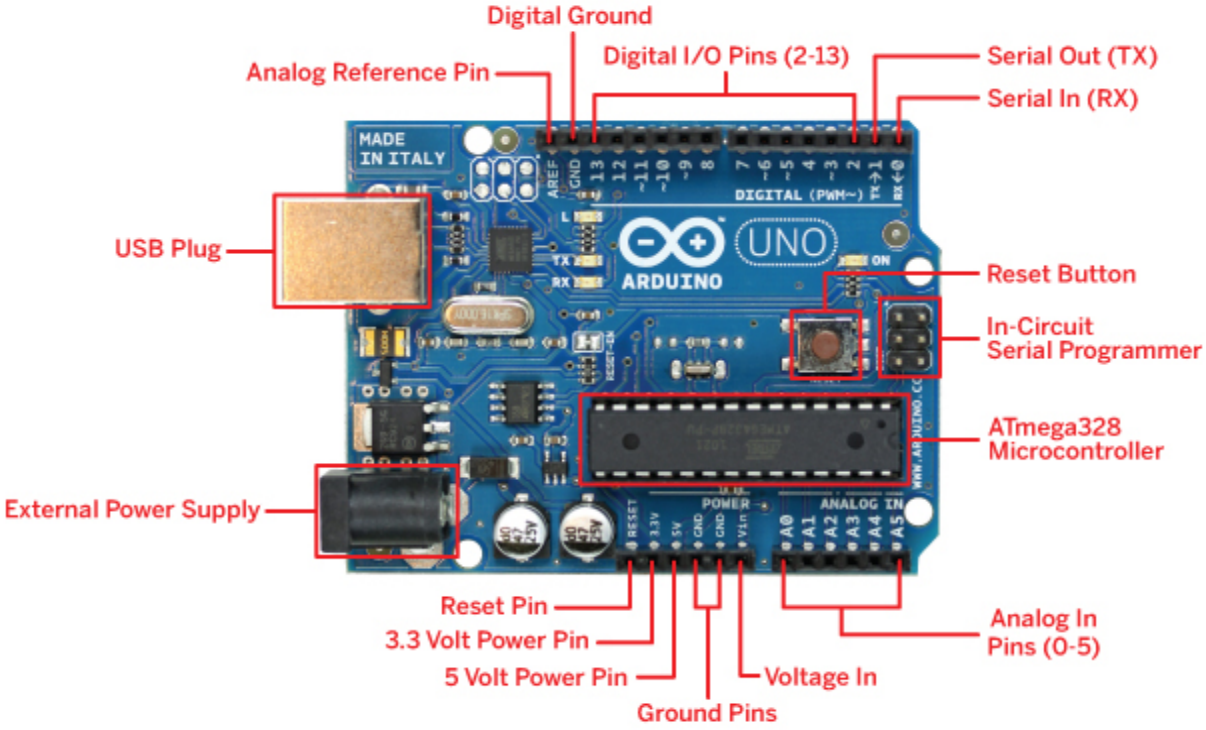




Рис. 1 – Зовнішній вигляд і роз'єми контролера ArduinoUno

### 1.6. ПАРАМЕТРИ КОНТРОЛЛЕРА ARDUINO UNO

Микроконтроллер

- ATmega328;

Живлення

- Від USB комп'ютера (+5 В, USB Plug) або зовнішнього джерела (+ 7 ... 12 В, ExternalPowerSupply)

Цифрові входи / виходи

- 14 штук - D0 ... D13, кожен з яких може видавати рівень напруги 0 В або 5 В або зчитувати їх. 6 з них (D3, D5, D6, D9, D10, і D11, зазвичай позначені на платі) можуть використовуватися як виходи регульованого рівня напруги в діапазоні 0 ... 5 В

Аналогові входи

- 6 штук - A0 ... A5. Вимірюють значення напруги на відповідному піне Arduino в діапазоні 0 ... 5 В. Можуть використовуватися як цифрові входи / виходи (D14 ... D18)

Максимальний струм

- 40 мА (досить, щоб живити світлодіод, але недостатньо, щоб живити електромотор). При перевищенні струму контролер може вийти з ладу

Флешпам `ять

- 32 Кб, при цьому 2 КБ використовуються для завантажувача, а 30 - для зберігання написаної програми для контролера

ОЗУ

- 2 КБ

Індикатори на платі

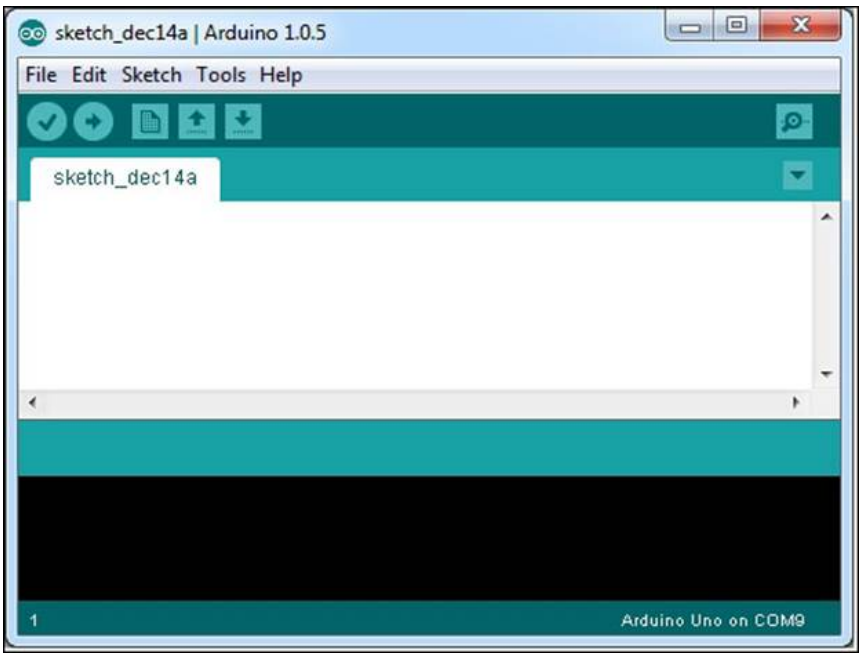
- світлодіод ON, загоряється при підключенні контролера до живлення
- світлодіоди RX, TX, миготливі в процесі прошивки контролера, а також при передачі / прийому інформації з комп'ютера
- світлодіод L, з'єднаний з цифровим контактом D13

### 1.7. Установка Arduino IDE

Завантажте інсталяційний файл з http://arduino.cc/en/Main/Software. Виберіть останню версію IDE Arduino, тобто 1.0.x або новішу версію.Переконайтеся, що ви завантажуєте версію IDE Arduino відповідно до вашої операційної системи, тобто 32-розрядної або 64-розрядної. Встановіть IDE в розташування за замовчуванням, як зазначено в майстрі установки. Після установки ви можете відкрити середу IDE, перейшовши в меню **Пуск | Програми**.

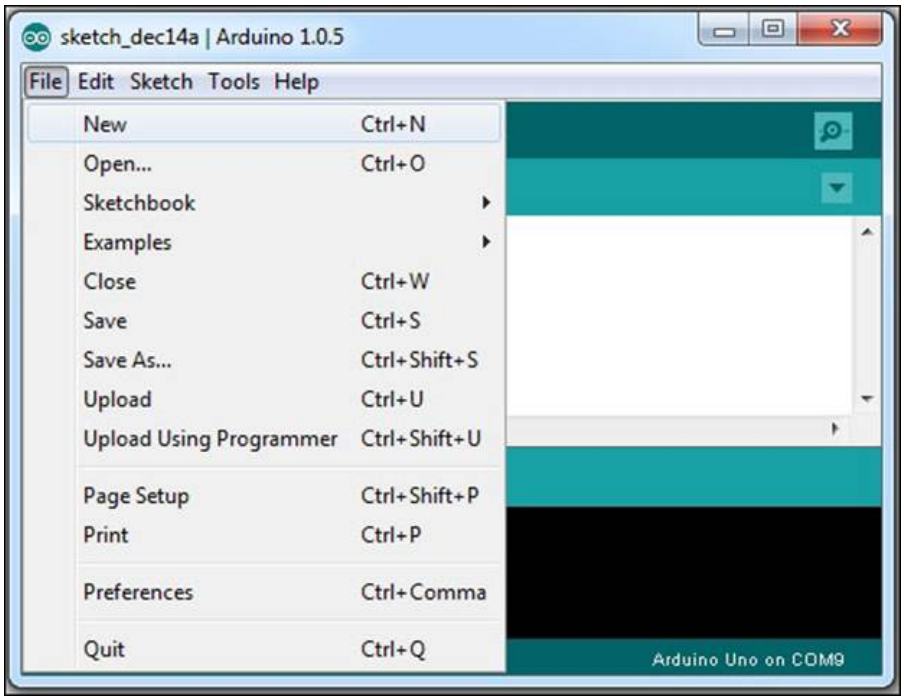
### 1.8. Початок роботи з інтегрованим середовищем розробки Arduino

Arduino IDE являє собою крос-платформний додаток, розроблений на Java, який може використовуватися для розробки, компіляції та завантаження програм на плату Arduino. При запуску IDE Arduino ви знайдете інтерфейс, аналогічний інтерфейсу, показаному на наступному скріншоті. IDE містить текстовий редактор для кодування, панель меню для доступу до компонентів IDE, панель інструментів для доступу до найбільш поширених функцій і текстову консоль для перевірки виходів компілятора. Рядок стану внизу показує обрану плату Arduino і ім'я порту, до якого вона підключена, як показано нижче:




Програма Arduino, розроблена з використанням IDE, називається скетчем. Скетчі кодуються на мові Arduino, який заснований на користувальницькій версії C/C++. Після написання коду у вбудованому текстовому редакторі, його можна зберегти за допомогою розширення. Коли ви зберігаєте ці файли скетчу, середовище IDE автоматично створює папку для їх зберігання. Якщо ви використовуєте будь-які інші підтримуючі файли для скетчу, такі як файли заголовків або файли бібліотек, всі вони зберігаються в цьому місці (яке також називається скетчем).

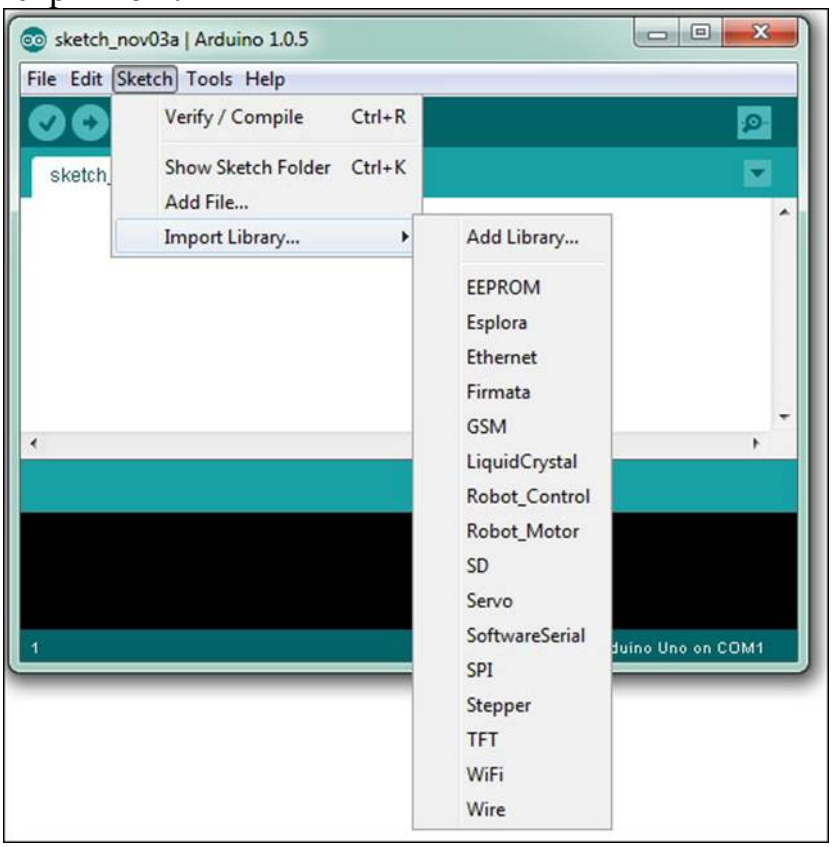
Щоб відкрити новий альбом, відкрийте Arduino IDE і виберіть «Створити» в меню «Файл», як показано на наступному скріншоті:



Вам буде запропоновано ввести порожній текстовий редактор. Текстовий редактор підтримує стандартні функції (тобто копіювати/вставляти, вибирати, знаходити/замінювати і т. д.). Перш ніж перейти до програми Arduino, давайте розглянемо інші інструменти, що надаються середовищем IDE.

### 1.9. Робота з бібліотеками

У середовищі IDE Arduino використовуються бібліотеки для розширення функціональних можливостей існуючих скетчів. Бібліотеки є набором функцій, об'єднаних для виконання завдань навколо конкретного компонента або концепції. Більшість вбудованих бібліотек Arduino надають методи для початку роботи із зовнішніми апаратними компонентами. Ви можете імпортувати будь-яку бібліотеку, перейшовши в **Sketch | Імпорт бібліотеки...,** як показано на наступному скріншоті:




Ви також можете використовувати бібліотеку для свого скетчу, просто вказавши її за допомогою оператора **#include** на початку скетчу, тобто, **#include<Wire.h>.**

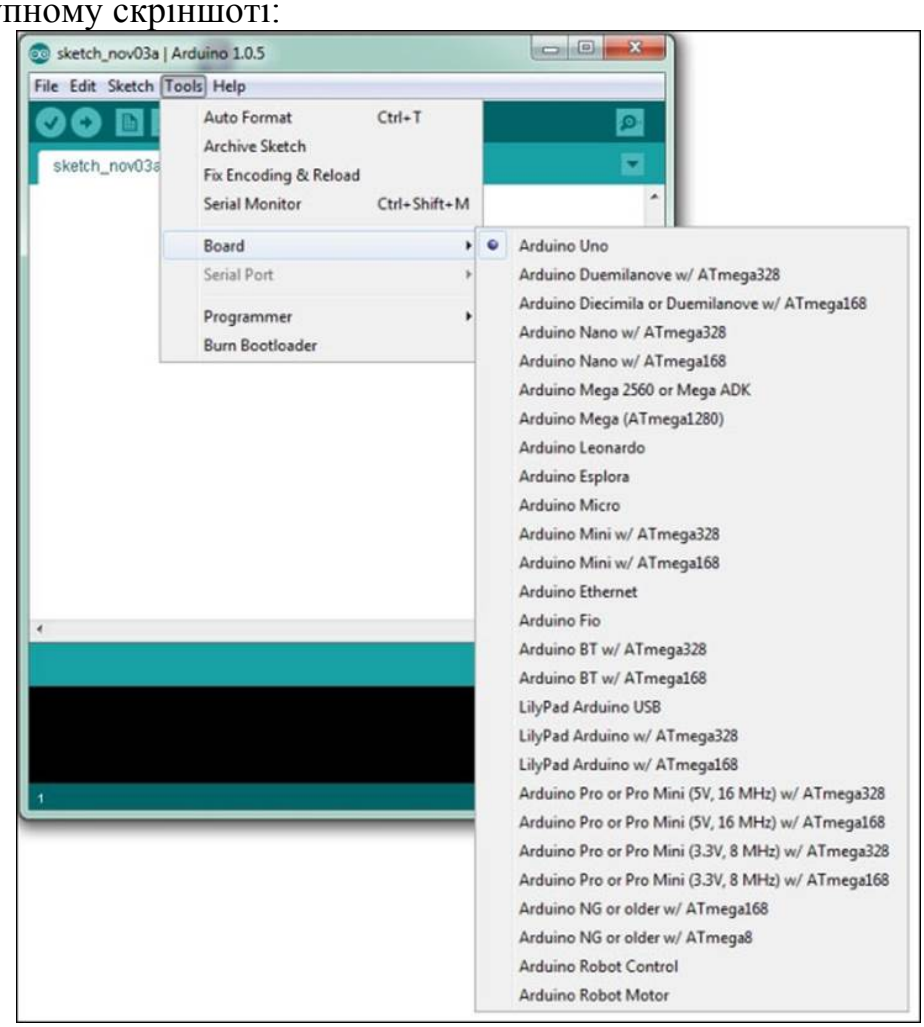
IDE Arduino також надає можливість додавання зовнішньої бібліотеки, яка підтримує конкретне обладнання або надає додаткові функції.

Детальніше про вбудовані бібліотеки Arduino ви можете дізнатися з http://arduino.cc/en/Reference/Libraries.

### 1.10. Компіляція і завантаження скетчів

Коли ви відкриєте свій код в середовищі IDE, перше, що вам потрібно зробити, це вибрати тип плати Arduino, на якій ви збираєтеся завантажити свій скетч. IDE Arduino повинен знати тип плати, щоб скомпілювати програму для відповідного мікроконтролера, оскільки різні плати Arduino можуть мати різні мікроконтролери Atmel. Тому вам необхідно виконати цей крок, перш ніж продовжити компіляцію або завантаження програми на плату.

Вибрати плату Arduino, перейшовши в **Інструменти|Плата**:, як показано на наступному скріншоті:




Виберіть Arduino Uno зі списку пристроїв, якщо ви не використовуєте іншу плату Arduino. Після того, як плату вибрана, ви можете продовжити і скомпілювати скетч. Ви можете скомпілювати скетч, перейшовши в **Скетч|Перевірити/компілювати** з рядка меню або за допомогою поєднання клавіш Ctrl+R. Якщо все налаштовано правильно, ви зможете скомпілювати код без будь-яких помилок.

Після успішної компіляції скетчу, необхідно завантажити скомпільований код в плату Arduino. Для цього вам необхідно переконатися, що ваш Arduino правильно підключений до комп'ютера. Якщо він ще не підключений, підключити плату Arduino до комп'ютера за допомогою USB-порту. Тепер прийшов час повідомити вашій IDE про серійний порт, до якого підключена

плата. Перейдіть в розділ **Інструменти|Порт** і виберіть відповідний послідовний порт.

Тепер ви можете завантажити скомпільований скетч на свою плату Arduino, перейшовши в **Скетч|Загрузка.** Цей процес буде використовувати послідовне з'єднання для запису компільованої прошивки в мікроконтролер. Почекайте деякий час або поки світлодіоди (світлодіоди Tx і Rx) на платі не перестануть блимати. Тепер у вас є готова плата Arduino з вашим першим скетчем. Ви можете спостерігати за роботою миготливого світлодіода поруч з цифровим виводом 13.

### 1.11. ВИКОРИСТАННЯ SERIALMONITOR

У попередньому прикладі ми використовували кабель універсальної послідовної шини (**UniversalSerialBus, USB**) для підключення плати Arduino до USB-порту вашого комп'ютера. USB-порт є промисловим стандартом для підключення різних електронних компонентів до комп'ютера з використанням послідовного інтерфейсу. Коли ви підключаєте плату Arduino за допомогою USB, комп'ютер фактично підключає його як послідовний периферійний пристрій. Скрізь будуть посилання на з'єднання, які зроблені з використанням USB в якості послідовних з'єднань. **SerialMonitor**- це вбудована утиліта IDE Arduino. Доступ до вікна SerialMonitor можна отримати, перейшовши в меню **Інструменти | Монітор порту** або за допомогою поєднання клавіш Ctrl+Shift+M. Він може бути налаштований для спостереження за даними, які відправляються або приймаються на послідовному порту, який використовується для підключення плати Arduino до комп'ютера. Ви також можете встановити швидкість передачі для послідовного зв'язку за допомогою випадаючого меню. Ця утиліта буде дуже корисною при тестуванні ваших прототипів і їх характеристик.

# 2. ВСТУП В ПРОГРАМУВАННЯ ARDUINO

Платформа Arduino була представлена для спрощення прототипів електронного обладнання для широкого кола користувачів. З цієї причини програмування Arduino призначалося для того, щоб легко навчатися непрограмістам, таким як дизайнери, художники і студенти. Мова Arduino реалізована на C/C++, в той час як основи скетчу й структур програм отримані з мови програмування з відкритим вихідним кодом під назвою **Processing**і відкритого електронного прототипування під назвою **Wiring.**

## 2.1. Коментарі

Arduino перейняв формат коментарів, який прийнятий в мові C, і він схожий на мови більш високого рівня. Однак він відрізняється від формату коментарів Python. Існують різні методи коментування, які полягають в наступному:

- **Blockcomment**: This is done by covering the commented text between /* and */:

**/ * Thisis a comment.**

***Arduino willignoreanytexttillitfindsuntiltheendingcommentsyntax, whichis, */**

- **Single-line orinlinecomment**: Thisisdonebyusing // beforetheline:

**// Thissyntaxonlyappliestooneline.**

**// Youhavetouseitagainforeachnextlineofcomment.**

**intpin = 13; // Selectedpin 13**

Зазвичай коментар блоку на початку скетчу в основному використовується для опису програми в цілому. Однорядкові коментарі використовуються для опису конкретних функцій або завдань, наприклад наступного:

**// TODO: explainvariablesnext.**

## 2.2. Змінні

Як і в будь-якій інша мові високого рівня, змінна використовується для зберігання даних з трьома компонентами: іменем, значенням і типом. Наприклад, розглянемо наступний оператор:

**intpin = 10;**

Тут **pin** – це ім'я змінної, яке визначено з типом **int** і містить значення **10.** Пізніше в коді все значення змінної **pin** будуть вилучатись як дані. Ви можете використовувати будь-яку комбінацію букв і цифр для вибору імені змінної, але перший символ НЕ повинен бути числом.

## 2.3. Константи

Мовою Arduino константи є зумовленими змінними, які використовуються для спрощення програми:

- **HIGH,LOW**: при роботі з цифровими **pin** на платі Arduino на цих виводах можливі тільки два окремих ступеня напруги. Якщо для отримання вхідного сигналу використовується контакт, будь-яка величина вище 3 В вважається станом **HIGH.** Якщо ви використовуєте **pin** для виведення, тоді стан **HIGH** встановиться напругою на виводі 5 В. Протилежні рівні напруги вважаються низькими (**LOW**) станами.

- **False, true**: вони використовуються для подання логічного істинного і помилкового рівнів. **False** визначається як 0, а **true** в основному визначається як 1.

- **INPUT,OUTPUT**: Ці константи використовуються для визначення ролей контактів Arduino. Якщо ви встановите режим виведення Arduino як **INPUT,** програма Arduino підготує **pin** для зчитування датчиків. Аналогічно, настройка **OUTPUT** підготує контакти для виходу сигналу.

## 2.4. Типи даних

Оголошення кожної змінної вимагає, щоб користувач вказував тип даних, пов'язаний зі змінною. Мова Arduino використовує стандартний набір типів даних, які використовуються на мові С.
Список цих типів даних і їх опис:

- **void**: використовується в оголошенні функції, щоб вказати, що функція не повертає ніякого значення:

```
voidsetup () {
// actions
}
```

- **boolean**: змінні, визначені за допомогою типу даних **boolean,** можуть містити тільки одне з двох значень: **true** або **false**:

```
booleanledState = false;
```

- **byte**: використовується для зберігання 8-бітного беззнакового числа, яке є в основному будь-яким числом від 0 до 255:

```
byte b = 0xFF;
```

- **Int**: це скорочення для цілих чисел. Він зберігає 16-розрядні (Arduino Uno) або 32-розрядні (Arduino Due) номера і є одним з основних типів даних зберігання даних для мови Arduino. Мова Arduino також

має **long**і **short** типи даних для особливих випадків:

**intvarInt = 2147483647**;
**longvarLong = varInt**;
**shortvarShort = -32768**;

- **float**: цей тип даних використовується для чисел з десятковими точками. Вони також відомі як числа з плаваючою комою. **float-** один з найбільш широко використовуваних типів даних, а також **int** для подання чисел на мові Arduino :

**floatvarFloat = 1.111**;

- **Char**: цей тип даних зберігає символьне значення і займає 1 байт пам'яті. При наданні значень типам **char** символьні літерали позначаються одинарними лапками:

**charmyCharacater = 'P';**

- **array**: масив зберігає набір змінних, доступний за номером індексу. Якщо ви знайомі з масивами в мові C/C++, вам буде простіше почати роботу, оскільки мова Arduino використовує ті ж масиви C/C++. Нижче перераховані деякі методи для ініціалізації масиву:

**intmyIntArray [] = {1, 2, 3, 4, 5};**
**inttempValues [5] = {32, 55, 72, 75};**
**charmsgArray [10] = "hello!";**

Доступ до масиву можна отримати за допомогою номера індексу (де індекс починається з номера 0):

**myIntArray [0] = = 1**
**msgArray [2] = = 'e'**

### 2.5. Перетворення

Функції перетворення використовуються для перетворення будь-якого значення типу даних в надані типи даних. Мова Arduino реалізує наступні функції перетворення, які можуть використовуватися під час програмування:

- **char():** перетворює значення будь-якого типу даних в тип символьних даних;
- **byte()**: перетворює значення будь-якого типу даних в тип даних байту;
- **int():** перетворює значення будь-якого типу даних в цілочисельний тип даних;
- **float():** перетворює значення будь-якого типу даних в тип даних з плаваючою комою.

В якості демонстрації використання цих функцій ознайомтеся з наступним прикладом:

**intmyInt = 10;**
**floatmyfloat = float (myInt);**

Реалізація попереднього коду створить змінну з плаваючою комою **myFloat**зі значенням 10.0, використовуючи цілочисельне значення, ініційоване змінною **myInt.**

## 2.6. ФУНКЦІЇ ТА ТВЕРДЖЕННЯ

Функції, звані також підпрограмами або процедурами, представляють собою частину коду, реалізованого для виконання конкретних завдань. Мова Arduino має деякі зумовлені функції, і користувач може також писати призначені для користувача функції для реалізації певної логіки програми. Потім ці призначені для користувача функції можна викликати з будь-якої частини скетчу для виконання конкретного завдання. Функції допомагають програмістам спростити настройку, зменшити ймовірність помилок і організувати концепції кодування:

```
voidblinkLED () {
// action A;
// action B;
}
```

Мова Arduino має набір бібліотечних функцій для спрощення програмування. Хоча не всі ці бібліотечні функції потрібні скетчам Arduino, **setup ()** і **loop ()** є обов'язковими функціями, і вони необхідні для успішної компіляції скетчу.

### *2.6.1. ФУНКЦІЯ SETUP ()*

Коли Arduino запускає скетч, він спочатку шукає функцію **setup().** Функція **setup()** використовується як для виконання важливих підпрограм, так і іншої частини програми, таких як оголошення констант, настройка **pin,** ініціалізація послідовного зв'язку або ініціалізація зовнішніх бібліотек. Коли Arduino запускає програму, ***вона виконує функції setup() тільки один раз***. Якщо ви подивіться скетч **Blink,** який ми використовували в попередньому розділі, ви можете побачити ініціалізацію функції **setup(),** як показано в наступному фрагменті коду:

```
voidsetup () {
// initializethedigitalpinasanoutput.
pinMode (led, OUTPUT);
}
```

Як можна бачити в прикладі, використовувалась функція **pinMode (),** щоб призначити роль світлодіодного виведення в функції **setup ().**

### *2.6.2. ФУНКЦІЯ LOOP () //ЦИКЛ*

Як тільки Arduino виконає функцію **setup (),*він почне безперервну ітерацію*** функції **loop ().** Хоча **setup ()** містить параметри ініціалізації, **loop ()** містить логічні параметри вашої програми:

```
voidloop () {
digitalWrite (led, HIGH);
delay (1000);
digitalWrite (led, LOW);
delay (1000);
}
```

Як ви бачите в попередньому фрагменті коду з скетчу **Blink,** функція **loop()** виконує основний код, який блимає світлодіод і повторює процес ітеративно.

*2.6.3. Функція pinMode ()*

Функція **pinMode ()** використовується для налаштування поведінки Arduino. Як ми бачили в функції **setup ()** скетчу **Blink,** функція **pinMode ()** налаштовує світлодіодне виведення для **OUTPUT**:

**pinMode (led, OUTPUT)**

Тут **led** змінна призначається цифрового виходу 13, режим якого буде змінений функцією **pinMode ().**

## 2.7. Робота з контактами

Після закінчення настройки контактів, які будуть використовуватися програмою, також знадобиться допомога в читанні виходів з них або для відправки їм сигналів. Arduino надає кілька конкретних функцій для обробки цих сценаріїв:

- **digitalWrite ():** Це було розроблено для цифрових контактів введення/виведення. Ця функція встановлює вихід **HIGH** (5V) або **LOW** (0V), які вже налаштовані як **OUTPUT** за допомогою **pinMode ().** Наприклад, наступний рядок коду встановлює цифровий вихід 13 в **HIGH**:

**digitalWrite (13, HIGH)** ;

- **digitalRead ():** ця функція, аналогічна функції **digitalWrite (),** дозволяє зчитувати стан цифрового виходу, налаштованого як **INPUT**:

**value = digitalRead (13)** ;

- **analogRead ():** ця функція зчитує значення з певного аналогового виходу. Значення лінійно відображається між цілим значенням 0 і 1023 для подання напруги від 0 В до 5 В:

**value = analogRead (0);**

- **analogWrite ()** : ця функція використовується для виведення результатів аналогового виходу на цифровий вихід. Цей метод називається **PWM,** і важливо відзначити, що ця функція не призначена для всіх цифрових контактів, але призначена тільки для контактів, позначених як **pin PWM.**

# 3. ПРАКТИЧНІ ЗАНЯТТЯ

**3.1. Практичне заняття 1.** Ознайомлення з контролером и середовищем програмування Arduino Uno

**3.2. Практичне заняття 2.** Використання прикладів Arduino

*Порядок роботи:*

1. Запустить середу програмування ArduinoIDE (ярлик Arduino на робочому столі комп'ютера)
2. Відкрийте приклад **Blink,** перейшовши в **Файл|Приклади|01.Basics|Blink.** IDE відкриє нове вікно, що містить код, схожий на код в наступній програмі:

```
/*
 Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
   digitalWrite(led, LOW);    // turn the LED off by making the voltage
LOW
  delay(1000);               // wait for a second
}
```

Цей скетч Arduino дозволяє блимати світлодіодом на цифровому виводі 13, оскільки плата Arduino Uno оснащена вбудованим світлодіодом, який підключений до цифрового виходу 13.

3. Підключіть Arduino до USB порту комп'ютера
4. Натисніть кнопку *Завантажити* Ü, переконайтеся, що в нижній частині вікна з'явився напис Компілювання, яка через деякий час зміниться написом *Завантаження* (на Arduino замигають світлодіоди RXTX) і з'явиться напис *Завантаження виконане*. Переконайтесь, що світлодіод L на платі почав блимати з періодом в 1 секунду

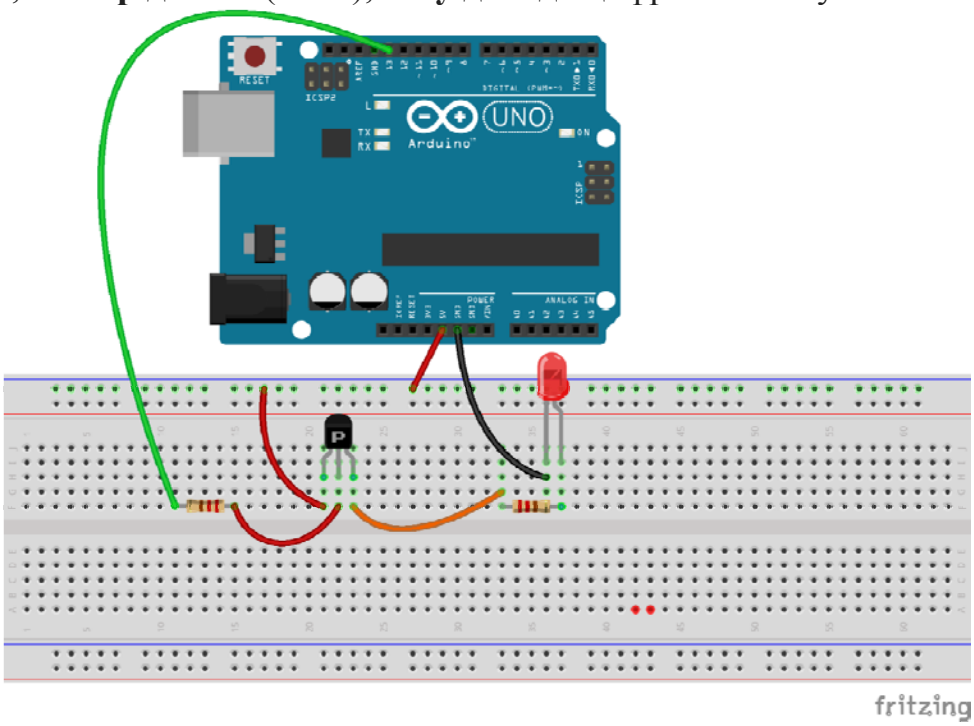
5. Змініть тайм миготіння світлодіода. У програмі періоди задані командою delay (затримка, очікування), при виконанні якої контролер чекає час в мілісекундах, заданий в дужках (1000 мілісекунд = 1 секунда).Видаліть один нуль в обох строках *delay(1000);*, щоб вони обидва мали вид *delay(100);*
6. Завантажте отриману програму в контролер (Ü) і переконайтеся, що світлодіод L на платі став блимати набагато швидше - з періодом 100 мілісекунд = 0,1 секунда, тобто 10 раз в секунду
7. Спробуйте задати різні значення періодів затримки delay і завантажити програму з ними

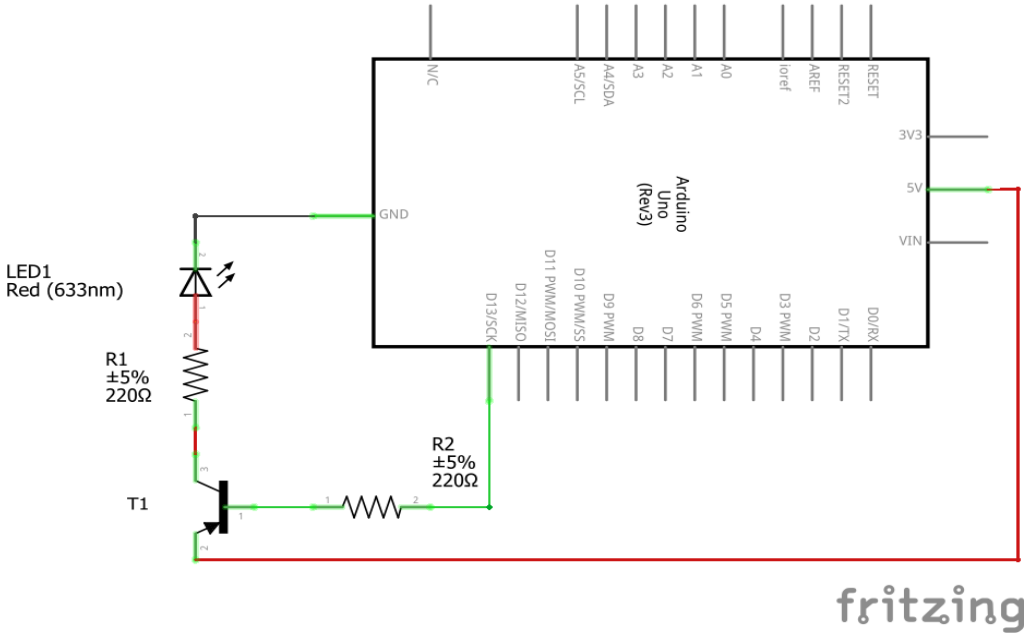
### 3.3. Практичне заняття 3. ПІДКЛЮЧЕННЯ ПОТУЖНОГО НАВАНТАЖЕННЯ ЧЕРЕЗ ДРАЙВЕР

Безпосередньо до цифрових контактах Arduino можна підключати тільки пристрої, які споживають невелику потужність, наприклад, світлодіод або спікер. Якщо струм навантаження буде набагато більше, ніж допустимий струм через цифровий контакт Arduino, рівний 40 мА., Arduino в кращому разі не буде працювати, в гіршому - контролер вийде з ладу.

Щоб цього не сталося, для управління контролером потужними приладами використовуються спеціальні пристрої, які називаються драйверами. У занятті розглядається використання в якості драйвера транзисторного ключа. Порядок роботи:

1. Запустіть середу програмування Arduino IDE (ярлик Arduino на робочому столі комп'ютера)
2. Відкрийте приклад **Blink**, перейшовши в **Файл | приклад | 01.Basics | Blink.**
3. Підключіть транзистор до контролера відповідно до схеми: **колектор** до **Gnd**, **емітер** до **Vcc** (+ 5V), **базу** до – до цифрового піну 13



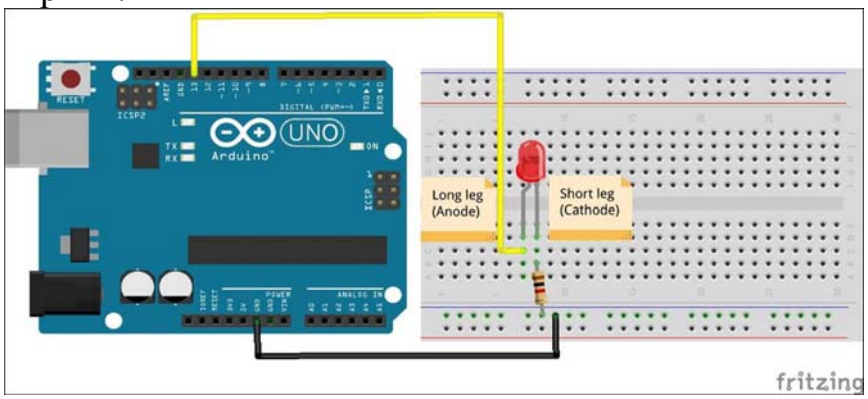



### 3.4. Практичне заняття 4. Тестування протоколу Firmata

У попередньому розділі використовувався вбудований світлодіод на виводі 13 для перевірки програми Blink. На цей раз буде використовуватися зовнішній світлодіод, щоб почати збірку апаратних компонентів за допомогою плати Arduino. Оскільки всі майбутні вправи і проекти будуть вимагати сполучення апаратних компонентів, таких як датчики і виконавчі механізми, з платою Arduino, буде корисно отримати практичний дослід підключення цих компонентів.

Світлодіод повинен мати дві ноги: коротку і довгу. Коротке плече пов'язане з катодом світлодіоду, і його потрібно підключити до землі через резистор. Як можна бачити на наступному рисунку, використовується резистор 220 Ом для заземлення катода світлодіоду. Довга нога, яка підключена до анода, повинна підключатися до одного з цифрових виводів плати Arduino.

Як показано на наступному рисунку, анод підключений до цифрового виходу номер 13. Подивіться на малюнок і переконайтеся, що плата Arduino відключена від хост-комп'ютера, щоб уникнути будь-якого пошкодження від статичної електрики.

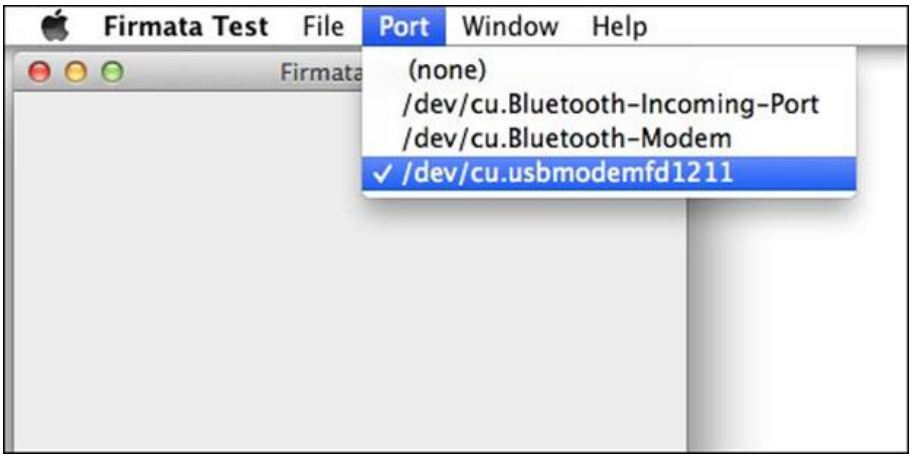


У цьому прикладі будемо використовувати світлодіод для перевірки деяких основних функцій протоколу Firmata. Після завантаження коду Firmata на плату Arduino, можна керувати світлодіодом з головного комп'ютера.

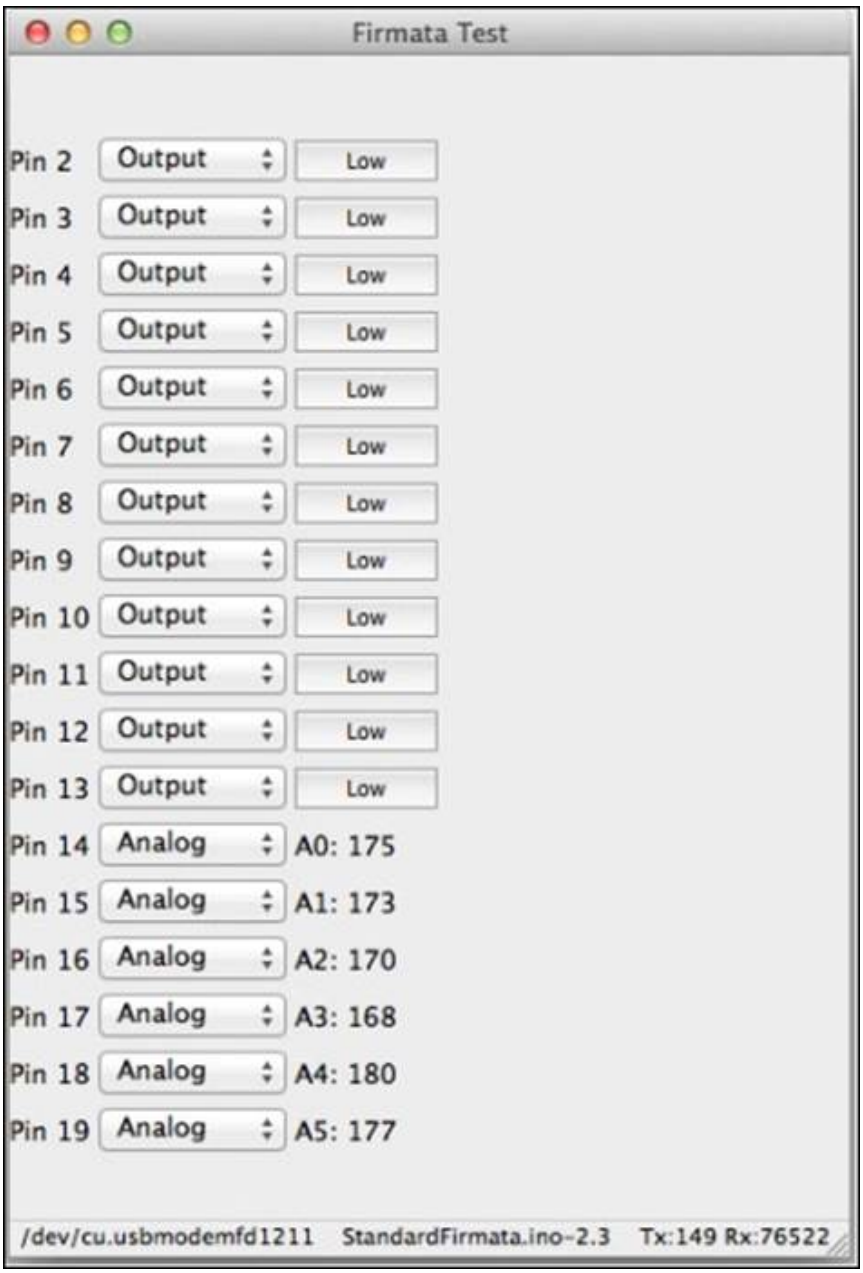
Є кілька способів спілкування з платою Arduino з головного комп'ютера за допомогою Firmata. Офіційний веб-сайт Firmata, **http://www.firmata.org,** надає інструменти для тестування, які можна завантажити з розділу **FirmataTestProgram** на головній сторінці. На веб-сайті є інший варіант інструменту під назвою **firmata_test** для різних операційних систем. Використовуючи такі кроки, можна протестувати реалізацію протоколу Firmata:

*Порядок роботи:*

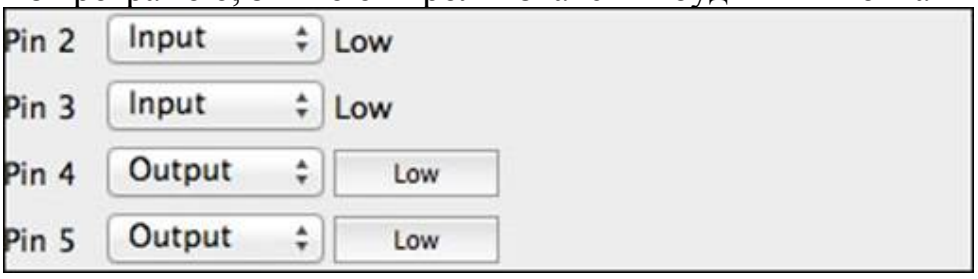
1. Завантажте відповідну версію програми **firmata_test.exe** на свій комп'ютер.
2. З'єднайте свою плату Arduino з світлодіодом на головному комп'ютері за допомогою USB-кабелю і запустіть завантажену програму **firmata_test.exe.** Можна побачити пусте вікно при успішному виконанні програми.
3. Як показано на наступному скріншоті, виберіть відповідний порт в спадному меню. Переконайтеся, що вибраний той самий порт, який використовувався для завантаження Arduino скетчу **(Файл | Приклади | Firmata | StandartFirmata).**



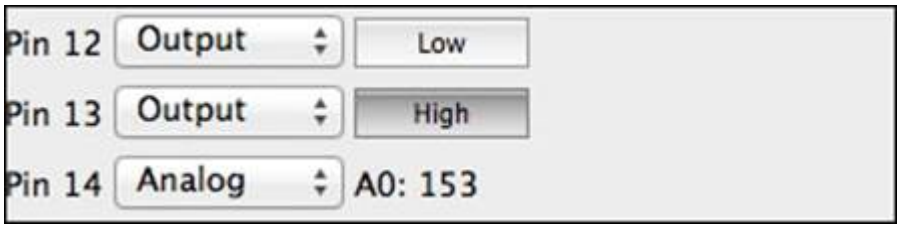

4. Після вибору послідовного порту Arduino програма завантажить кілька випадаючих полів і кнопок з мітками, які містять номер виходу. На наступному скріншоті можна бачити, що у програмі завантажено 12 цифрових виходів (від виходу 2 до контакту 13) і шість аналогових виходів (від виходу 14 до виходу 19). Оскільки використовується плата Arduino Uno, тестова програма завантажує тільки контакти, які є частиною Arduino Uno.




5. Як видно на скріншоті, є три стовпці. Перший стовпець позначає номер **pin.** Другий стовпець в програмі дозволяє вибрати роль для відповідних контактів. Можна вказати роль цифрових контактів (в разі Arduino Uno, від 2 до 13) в якості введення або виведення. Третій стовпець показує рівень сигналу. Наприклад, як показано на наступному скріншоті, **Low** в третьому стовпці, якщо обрані **pin** 2 і 3 в якості вхідних виходів. Це правильно, оскільки немає ніякого вхідного сигналу, підключеного до цих контактів. Можна працювати з програмою, змінюючи ролі і значення будь-яких контактів.



6. Якщо вибрати контакт 13 в якості вихідного контакту і натиснути кнопку на **Low,** то це змінить ярлик кнопки на **High,** і можна побачити, що світлодіод включений. Виконуючи цю дію, логіка цифрового виводу 13 змінилася на 1, тобто **High,** що відповідає +5 В на виведення. Цей потенціал буде достатнім для світіння світлодіоду. Можна змінити рівень **pin** 13 назад на 0, знову натиснувши на кнопку і перевівши її в положення **Low.** Це змінить потенціал назад на 0 В.




## ТЕМИ ДЛЯ САМОСТІЙНОЇ РОБОТИ

Назва теми та види самостійної роботи студента:

1) Підготовка до аудиторних занять
- проробка лекційного матеріалу;
- підготовка до лабораторних робіт;
- підготовка до практичних занять;

2) Проробка розділів, які не викладаються на лекціях:
- Пасивні компоненти електронних пристроїв.
- Тунельні, високочастотні та імпульсні діоди.
- Світловоди і світловодні системи. Електронно-променеві індикатори. Газорозрядні індикатори.
- Технологія виготовлення інтегральних мікросхем.
- Багатокаскадні підсилювачі.
- Вибіркові підсилювачі.
- Імпульсні підсилювачі.
- Стабілізація частоти автогенераторів.
- Інвертори. Конвертори.
- Модулятори і демодулятори.
- Множники напруги.
- Блокінг-генератори.
- Ознайомлення з варіантами контролерів Arduino

3) Виконання та захист:
Курсовий проект "Електронні пристрої автоматики"

4) Підготовка та складання підсумкового контролю знань (МКР, залік, іспит).

# СПИСОК ЛІТЕРАТУРИ

1. Колонтаєвський Ю.П., Сосков А.Г. Промислова електроніка та мікросхемотехніка: теорія і практикум / За ред. А.Г. Соскова. – К.: Каравела, 2003. – 368 с.

2. Основи схемотехніки електронних систем: Підручник / В.І. Бойко, А.М. Гуржій, В.Я. Жуйков та ін. – К.: Вища шк., 2004. – 527 с.

3. Основы промышленной электроники / Герасимов В.Г., Князьков О.М., Краснопольский А.Е. и др. // Под ред. В.Г. Герасимова. – 3-е изд., перераб. и доп. – М.: Высш. шк., 1989. – 336 с.

4. Методичні вказівки до виконання курсового проекту з дисципліни „ Електронні пристрої автоматики” для студентів III курсу усіх форм навчання за спеціальністю: 151 Автоматизація та комп’ютерно-інтегровані технології / Укл.: О.П. Мисов, Л.Д. Чумаков – Дніпро: ДВНЗ УДХТУ, 2019, – 46 с.

**Електронні ресурси:**

1. http://freepdf-books.com/python-programming-for-arduino/
2. https://www.pdfdrive.net/python-programming-for-arduino-e25954218.html
3. https://doc.arduino.ua/ru/prog/
4. http://fritzing.org/download/
5. http://arduino.net.ua/file_archive/Arduino%20Library/Arduino%20Library/