

Іванов С.А.



Запоріжжя

2019

УДК
ББК

Іванов С.А.

Програмно-апаратна платформа Arduino на STEM-уроках / С.А. Іванов. — Видавництво “Освіта”, 2019. — 150 с.: іл.

ISSN

Посібник призначено насамперед для вчителів інформатики, природничих дисциплін та трудового навчання, які прагнуть до використання програмно-апаратної платформи Arduino на STEM-уроках. У посібнику розглянуто базові відомості щодо архітектури мікроконтролеру Arduino, його розширених плат multifunctional shield та joystick shield, візуальної мови програмування Scratch у програмному середовищі mblock та методи роботи зі створення навчальних Arduino-проектів з елементами електроніки та робототехніки та розробки простіших комп'ютерних ігор та анімації. Посібник містить теоретичний матеріал, практичні роботи, тести і завдання для самостійного виконання. Головною метою посібника є сформувати мотивацію учнів навчитися програмуванню, ознайомити з елементами електроніки, розвинути творче мислення, створюючи власні ігри, використовуючи управління спрайтами за допомогою платформи Arduino.

Електронний архив на сайті

ЗМІСТ

Вступ	
Розділ 1. Програмно-апаратна платформа Arduino	
1.1. Концепція та архітектура програмно-апаратної платформи Arduino	
1.2. Система вводу-виводу (I/O)	
1.3. Установка програмного забезпечення	
1.3. Інтерфейс інтегрованого середовища розробки IDE	
Розділ 2. Основи програмування Arduino	
2.1. Структура програми Arduino	
2.2. Підключення бібліотек	
Розділ 3. Багатофункціональна плата (Multifuntional Shield)	

Вступ

Сучасний світ містить безліч вбудованих систем, які оточують нас всюди. Їх можна зустріти і в дитячих іграшках, в сучасних приладах і системах управління, зокрема обладнання “розумного дому”. Основною частиною таких систем є мініатюрні комп'ютери, які відомі під назвою мікроконтролерів. Мікроконтролер - це повноцінна комп'ютерна система, збудована на одному чіпі, яка є потужним інструментом для спілкування системи управління з оточуючим середовищем. Якби не були вбудовані системи (чи то іграшки, чи телефон, чи пральна машина, чи елементи системи “розумного дому”) всі вони побудовані на однакових принципах – во всіх в пам'яті мікроконтролерів міститься комп'ютерна програма. Вона працює у нескінченному циклі виконання і за допомогою тих чи інших датчиків чекає на будь-яку подію контролює параметри цієї системи. Розрізняються мікроконтролери один від одного внутрішньою структурою і виконуваними функціями.

Сукупність апаратної частини мікроконтролеру та його програмного забезпечення будемо називати програмно-апаратною платформою. На сьогодні таких платформ існує велика кількість. Серед них можливо найбільше розповсюдження (особливо в сфері освіти та технічної творчості школярів) набула програмно-апаратна платформа Arduino. Історія її створення і розповсюдження нагадує історію компютера **Apple, яка починалася у гаражі на засадах нової концепції, розробленої засновниками Стивом Джобсом та Стивом Возняком**. Нова мікроконтролерна плата була створена під керівництвом італійського співзасновника проекту у сфері електроніки Массимо Банці (Massimo Banzi) і побачила світ у 2005 році. Творча команда М. Банці поійменувала її Ардуино (Arduino), сидячи у барі з такою назвою, яку, у свою чергу, було отримано на честь колишнього короля.

За короткий термін нова мікроконтролерна плата викликала справжню світову революцію для мільйонів творчих людей, які отримали можливість розробляти власні “розумні” пристрої. Ця революція відбулася завдяки трьом чинникам:

- невелика вартість плати Arduino (сьогодні — це вже 100-200 грн. В залежності від типу;
- відкритість програмно-апаратної платформи, що створило умови для розробки численних клонів Arduino;
- наявність достатньої кількості цифрових та аналогових контактів для зв'язку з оточуючим середовищем з метою управління різноманітними електронними пристроями чи механізмами.

Названі чинники платформи Arduino очікувано створили умови для використання її у сфері освіти. Практично во всіх країнах світу оголошуються національні проекти, спрямовані на упровадження мікроконтролерної плати Arduino у навчальний процес. В Інтернеті наростає кількість джерел, які містять відповідні навчальні матеріали, починаючи з програмного середовища у вигляді блогів сайтів, посібників, , мільйони людей різного віку створюють саморозробки із використанням платформи Arduino.

Водночас досвід використання платформи Arduino виявив певні перешкоди на шляху масового її упровадження, зокрема, на STEM-уроках. По-перше, це необхідність оволодіння мовою програмування мікроконтролеру (Wiring, Python, C++ або будь-яка інша). Справа у тому, що на думку багатьох вчителів інформатики, лише на уроках важко сформувати в учнів алгоритмічне мислення, вони не мають мотивації на вивчення будь-якої мови програмування.

Їм не вистачає терпіння на пошук і виправлення помилок. По-друге, - збірка схеми проекту за допомогою Arduino вимагає певного часу, що входить в протиріччя з обмеженістю часу уроку.

З метою подолання цих перешкод можна запропонувати такі підходи. Щодо труднощів програмування альтернативою будь-якої класичної або об'єктно-орієнтованої мови є програмні середовища, які використовують візуальні засоби програмування. Враховуючи те, що учні 5-7 класів вже мають певний досвід роботи з візуальною мовою Scratch, у цьому посібнику використовується програмне середовище mblock, яке дозволяє не тільки програмувати будь-які події зі спрайтами, але й має належні інструменти для програмування плати Arduino. Більше того, це середовище створює умови для управління спрайтами за допомогою плати Arduino та тих чи інших електронних компонентів, зокрема, джойстика та кнопок, що суттєво розсовує рамки творчого простору учнів. Щодо скорочення часу на збірку навчальних проектів, можна використовувати плати розширення Arduino, зокрема багатофункціональну плату Multi-function Shield for Arduino, яка надає можливість створити простішу цифрову лабораторію на уроках природничих дисциплін та інформатики. Водночас це не виключає можливості створення робототехнічних проектів, використовуючи такі компоненти як крокові двигуни, сервомотори, інші типи латчиків.

Оволодіння учнями базовими знаннями та певним досвідом роботи з програмно-апаратною платформою Arduino, може стати для учнів першим кроком на шляху формування мотивації до самоосвіти у напрямку створення власних Arduino-проектів в різних галузях техніки. Водночас, й це можливо найважливіше, в учнів виховується інтелектуальна компетентність. Зокрема, розвивається алгоритмічне мислення, тому що учень, створюючи гру у середовищі mblock, навчає персонажі (спрайти) виконувати ту чи іншу послідовність команд. Також розвивається творче мислення, тому що учень самостійно може придумувати сюжети гри (квесту), створювати сценарії подій, створювати персонажі. У процесі роботи над проектом формуються навички розв'язання проблем, подолання труднощів, що виникають. Врешті решт, учень усвідомлює важливість самонавчання. Внаслідок того, щоробота над проектом виконується, як правило, у групі, учні розвивають соціальні компетентності, зокрема, вміння співпрацювати з іншими, вміння дискутувати та відстоювати власну точку зору. Нарешті, самоосвіта і саморозвиток у STEM-дисциплінах вимагає вивчати, запозичувати та використовувати іноземний досвід, що вимагає вдосконалення, щонайперше, знання англійської мови.

РОЗДІЛ 1 ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА ARDUINO

1.1. Концепція та архітектура програмно-апаратної платформи Arduino

Платформа Arduino являє собою відкриту апаратну обчислювальну платформу, основними компонентами якої є проста плата вводу/виводу та середовище розробки на мові програмування Processing/Wiring. Відмінними рисами програмно-апаратної платформи Arduino є:

- 1) багатоплатформеність (здатна працювати по ОС Windows, Macintosh, Linux);
- 2) спрощеність оболонки програмного середовища і мови, доступними для навчання учнів;
- 3) Usb-подібний інтерфейс для організації програмування, сполучення із зовнішнім світом і живлення;
- 4) відкрите апаратне і програмне забезпечення;
- 5) освітній характер платформи.

Arduino дозволяє учню зосередитися на розробці проектів, а не на вивченні пристрою та принципів функціонування окремих елементів. Розробник може використовувати готові плати розширення або просто підключити до Arduino необхідні елементи. Всі останні зусилля розробник спрямовує на розробку керуючої програми на мові високого рівня. Таким чином доступ до розробок мікропроцесорних приладів отримали не тільки професіонали, но і просто аматори, починаючи з шкільного віку. Наявність готових модулів і бібліотек програм дозволяє користувачам створювати готові прилади для розробки власних задач. Напрями та типи пристроїв на базі Arduino обмежені лише можливостями конкретного мікроконтролера та творчістю розробника.

До складу програмно-апаратної платформи Arduino входить апаратне і програмне забезпечення. Плата Arduino складається з мікроконтролера Atmel AVR (ATmega328) та елементної обв'язки (аналогові та цифрові контакти вводу-виводу) для програмування та інтеграції з іншими схемами. До складу платформи також входять цілий спектр так званих шильд-плат, які можуть бути зістиковані з материнською платою, що розширює її функціональні можливості.

Сьогодні існує багато різновидів Arduino (клонів), але найбільш популярною вважається плата Arduino Uno, зовнішній вигляд якої представлений на Рисунку 1:



Рис. 1.1. Зовнішній вигляд Arduino Uno

Розробники Arduino розмістили на цій платі всі компоненти, необхідні для нормальної роботи та зв'язку цього мікроконтролера із зовнішнім світом. Плата може бути запитана від USB-порту комп'ютера, більшості USB-зарядних пристроїв або від АС-адаптера (рекомендується напругою 9 вольт, роз'єм 2,1 мм, плюс в центрі). Якщо в роз'єм живлення не підключено джерело, плата отримує живлення від USB-роз'єму, але як тільки ви підключите джерело живлення, вона автоматично переключиться на нього. Плата містить 20 роз'ємів, призначення яких і спосіб використання стануть зрозумілі в процесі вивчення даного посібника. Зараз ми тільки перерахуємо їх:

1) 14 контактів цифрового вводу-виводу (контакти 0-13). Вони можуть бути як входами, так і виходами, що визначається програмою.

2) 6 контактів аналогового входу (контакти 0-5). Ці окремі контакти для аналогового входу отримують аналогові значення (наприклад, величину напруги в датчику) і перетворюють їх у цифри від 0 до 1023.

3) 6 контактів аналогового виходу (контакти 3, 5, 6, 9, 10 і 11) - цифрові контакти подвійного призначення, які можуть бути запрограмовані на аналоговий вихід за допомогою програми.

Технічні характеристики мікроконтролера Arduino Uno:

Тип мікроконтролера: ATmega328P;

Напруга живлення мікроконтролера: 5 В;

Рекомендоване напруга живлення плати: 7 - 12 В;

Гранично допустима напруга живлення плати: 6 - 20 В;

Цифрові входи-виходи: 14 (з них 6 підтримують ШІМ);

Виходи ШІМ модуляції: 6;

Аналогові входи: 6;

Допустимий струм цифрових виходів: 20 мА

Допустимий струм виходу: 50 мА

Обсяг флеш пам'яті (FLASH): 32 кБ (з яких 0,5 кБ використовується завантажувачем)

Об'єм оперативної пам'яті (SRAM): 2 кБ

Обсяг енергонезалежної пам'яті (EEPROM): 1 кБ

Частота тактирования: 16 мГц

Довжина плати: 68,6 мм

Ширина плати: 53,4 мм

Вага: 25 г

Розташування (розпиновку) виводів плати Arduino UNO представлено на Рис. 1.2.

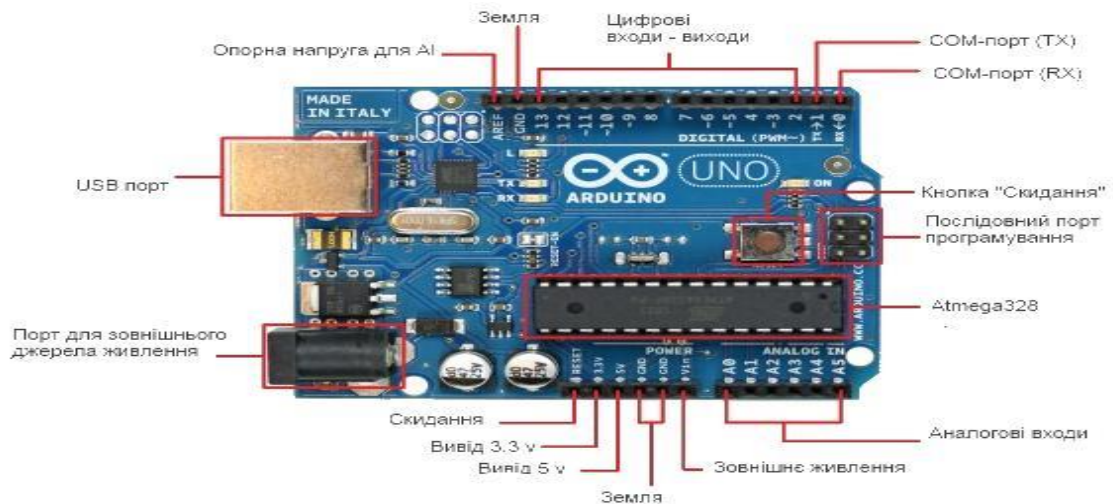


Рис.1.2. Розпиновка виводів плати Arduino UNO

1.2. Система вводу-виводу

1.2.1. Цифрові та аналогові сигнали

Програмно-апаратна платформа Arduino сприймає та передає будь-які дані від зовнішніх приладів чи пристроїв з метою їх візуалізації або керування. Ці дані фізично являють собою певні сигнали, які мають два основних типи: аналогові та цифрові. Аналогові сигнали є природними (світлові чи електричні), їх можна зафіксувати за допомогою тих чи інших видів датчиків. Наприклад, датчиками навколишнього середовища (температура, тиск, вологість) або механічними датчиками (швидкість, прискорення). Аналогові сигнали в математиці описуються безперервними функціями. Цифрові сигнали є штучними, тобто їх можна отримати тільки шляхом перетворення аналогового електричного сигналу (Рис. 1.3.).

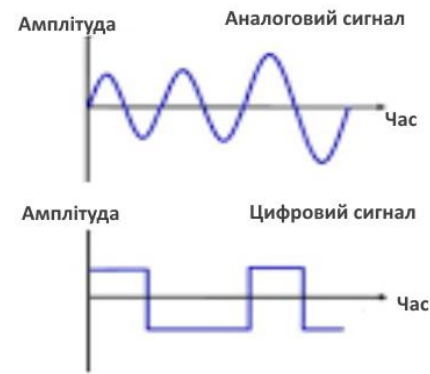


Рис. 1.3. Аналоговий та цифровий сигнал

Аналоговий сигнал поширюється безперервно, а цифровий сигнал - дискретно (уривчасто), тобто амплітуда коливань приймає певні значення в одиницю часу. Процес послідовного перетворення безперервного аналогового сигналу називається дискретизацією. Дискретизація буває двох видів: а) за часом, б) за амплітудою.

Процес перетворення вхідної фізичної величини (аналогового сигналу) в її числовий (цифровий) вигляд має назву аналого-цифрове перетворення. Аналого-цифровий перетворювач (АЦП) - пристрій, що виконує таке перетворення. Формально, вхідною величиною АЦП може бути будь-яка фізична величина - напруга, струм, опір, ємність, частота проходження імпульсів, кут повороту вала і т.п. Однак, для визначеності, в подальшому під АЦП ми будемо розуміти виключно перетворювачі напруга-код.

Поняття аналого-цифрового перетворення тісно пов'язане з поняттям вимірювання. Під вимірюванням розуміється процес порівняння вимірюваної величини з певним еталоном, при аналого-цифровому перетворенні відбувається порівняння вхідної величини з деякою опорною величиною (як правило, з опорною напругою). З метою уявлення роботи аналого-цифрового перетворення розглянемо рисунок 1.4. Через рівні проміжки часу (вісь X) відбувається зчитування значення напруги на вході АЦП (вісь Y). Внаслідок того, що АЦП має обмежену роздільну здатність, з'являється дискретність (дроблення) значень.



Рис. 1.4. Перетворення аналогового сигналу в цифровий

1.2.2. Цифрові вводи/виводи

Плата Arduino Uno містить 14 контактів для прийому або передачі цифрових сигналів.



Рис.1.5. Цифрові контакти 2-13

Цифрові виводи платформи Arduino можна використовувати як вхідні або як вихідні. Вони поділяються на два типи, на контактах без знака \sim можуть мати тільки два значення напруги (0 або 5 вольт), напруга на контактах зі знаком \sim може мінятися від 0 до 5 вольт з кроком 0.0196 вольт. У подальшому терміни *порт* та *контакт* (а також *pin*) будемо вважати рівнозначними.

Варто зазначити, якщо до того чи іншого порту у режимі вводу (високоімпедансний стан) нічого не підключено, то на ньому наводяться випадкові величини внаслідок або електричних перешкод або ємнісного взаємозв'язку з сусіднім виводом. Якщо до цифрового контакту, програмно сконфігурованого як вхідний (про що буде далі), підключені будь-які електричні або електронні елементи, то цей порт мало впливає на роботу цієї схеми. Тому що внутрішній опір цифрових портів має занадто високе значення (майже 100 Мом). Іншими словами, по вказаному опорі тече дуже маленький електричний струм. Внаслідок цього для переведення порту вводу з одного стану в інший (від 0 до 5 вольт) потрібно невелике значення струму.

Інша справа, якщо порт програмно сконфігурований як вихідний (тобто знаходяться в нізкоімпедансному стані), він може пропускати через себе досить великий струм (до 40 мА для інших пристроїв). Такого значення струму досить щоб підключити світлодіод (обов'язковий послідовно включений резистор), датчики, але недостатньо для більшості реле, соленоїдів і двигунів.

Важливо! Короткі замикання виводів Arduino або спроби підключити енергоємні пристрої можуть пошкодити вихідні транзистори виводу або весь мікроконтролер Atmega. У більшості випадків дані дії приведуть до відключення виводу на мікроконтролері, але інша частина схеми буде працювати згідно з програмою. Рекомендується до виходів платформи

підключати пристрої через резистори 220-470 Ом або 1 кОм, якщо пристрою не потрібен більший струм для роботи.

1.2.3. Аналогові вводи/виводи

Плата Arduino Uno містить 6 портів (контактів) для прийому або передачі аналогових сигналів.



Рис. 1.6. Аналогові контакти (A1:A6)

Мікроконтролери Atmega, використовувані в Arduino Uno, містять шестиканальний аналого-цифровий перетворювач (АЦП). Роздільна здатність перетворювача складає 10 біт, що означає, що напругу, яка подається на аналоговий вхід від 0 до 5 вольт може мінятися з кроком 0.0049 вольт (у відносних одиницях від 0 до 1023, це +1024, що дорівнює 2^{10}). Основним застосуванням аналогових входів більшості платформ Arduino є зчитування даних з аналогових датчиків, але в той же час вони мають функціональність вводів/виводів широкого застосування.

Важливо знати, що цифрові виводи Arduino, які відмічені тильдою (~), здатні моделювати аналоговий сигнал, тобто на них можна подавати або зчитувати напругу між 0 та 5 вольтами. Операція отримання аналогових значень напруги шляхом використання цифрових виводів має назву широтно-імпульсна модуляція (ШИМ). Широтно-імпульсний модульований сигнал - це імпульсний сигнал постійної частоти, але змінної шпаруватості (співвідношення тривалості імпульсу і періоду його проходження) (Рис. 1.7.).

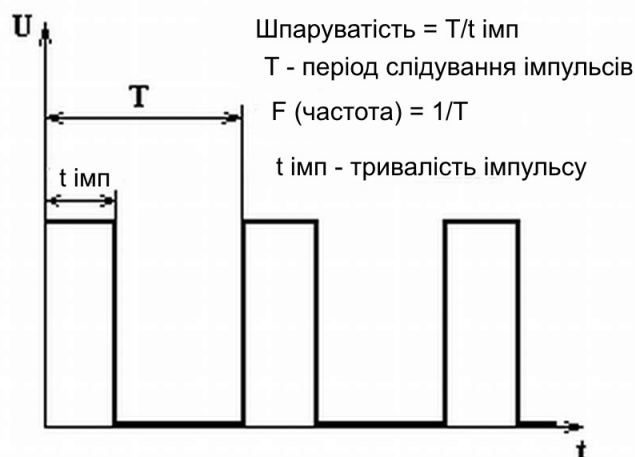


Рис.1.7. Шпаруватість прямокутного імпульсу

Через те, що більшість фізичних процесів в природі мають інерцію, то різкі перепади напруги від 1 до 0 будуть згладжуватися, приймаючи деяке середнє значення. За допомогою

завдання шпаруватості можна міняти середню напругу на виході ШІМ. Якщо шпаруватість дорівнює 100%, то весь час на цифровому виході Arduino буде напруга логічна "1" (тобто 5 вольт). Якщо задати шпаруватість 50%, то половину часу на виході буде логічна "1", а половину - логічний "0", тобто середня напруга буде дорівнювати 2,5 вольт (Рис. 1.8.).

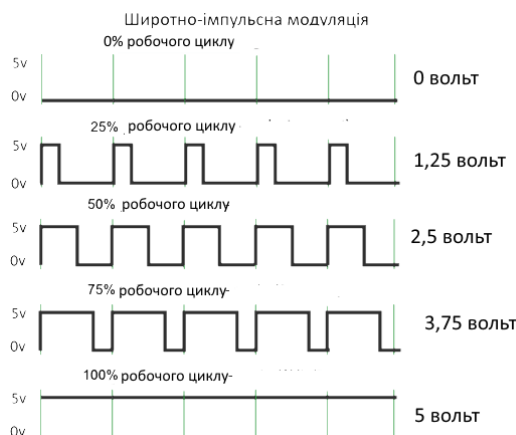


Рис. 1.8. Використання широтно-імпульсної модуляції сигналу

Платформа Arduino встановлює на всіх виводах ШІМ частоту зміни сигналів 488,28 Гц та розділення 8 розрядів ($2^8 = 255$). Це означає, по-перше, що зміна сигнала відбувається біля 2 мілісекунд, що для людського ока непомітно. По-друге, напругу на цифрових ШІМ-виходах можна виставляти з кроком 0,196 вольт (5 вольт/255).

1.3. Установка програмного забезпечення

Перш ніж підключати апаратну частину Arduino до персонального комп'ютера, необхідно встановити на нього інтегроване середовище розробки плати IDE (Integrated Development Environment). Порядок установки IDE викладений у багатьох Інтернет-джерелах, але у зв'язку з тим, що випускаються все нові і нові релізи, цей порядок змінюється. Тому, слід вибрати той порядок, який відповідає обраному типу плати і відповідної версії програмного забезпечення. Найкраще слід звернутися до офіційного сайту <https://www.arduino.cc> та слідувати цьому порядку. Між іншим сьогодні існує можливість роботи з Arduino в он-лайн режимі.

Порядок завантаження здійснюється таким чином.

а) Знайдіть останню версію на сторінці скачування: <https://www.arduino.cc/en/Main/Software>). Після закінчення завантаження розпакуйте скачаний файл. Переконайтеся, що не порушена структура папок. Відкрийте папку подвійним кліком на ній. У ній мають бути кілька файлів і підкаталогів.

б) Підключіть плату: Arduino Uno отримує живлення автоматично від будь-якого USB-підключення до комп'ютера або іншого джерела живлення. Має засвітитися зелений світлодіод живлення, позначений PWR.

в) Встановіть драйвера:

Установка драйверів для Arduino Uno на Windows 7 виконується таким чином:

- Натисніть на кнопку ПУСК і відкрийте Панель керування.
- На панелі керування перейдіть на вкладку **Система і безпека** (System and Security).

Потім виберіть **Система**, після чого виберіть **Диспетчер пристроїв** (Device Manager).

- Зверніть увагу на порти (COM і LPT). Ви побачите відкритий порт під назвою «Arduino UNO (COMxx)».

- Клацніть на назві «Arduino UNO (COMxx)» правою кнопкою мишки і виберіть опцію «Оновити драйвер» (Update Driver Software).

- Клікніть "Browse my computer for Driver software".

- Для завершення виберіть завантажений драйвер для Uno - «ArduinoUNO.inf», розташований в папці Drivers програмного забезпечення для Arduino (не в підкаталозі «FTDI USB Drivers»).

При підключенні плати Arduino до комп'ютера під керуванням Windows 10, треба упевнитися, що в системі встановлений драйвер. Драйвер налаштовує Arduino як віртуальний COM-порт, який можна побачити в диспетчері пристроїв. Відкрийте вікно Device Manager (Диспетчер пристроїв), клацнувши правою кнопкою миші кнопку запуску Windows 10 в лівому нижньому кутку екрану, а потім виберіть Диспетчер пристроїв в меню.

У диспетчері пристроїв розгорніть порти (COM і LPT), і можна побачити COM-порт, який буде вашим Arduino, як показано на Рис.1.3.

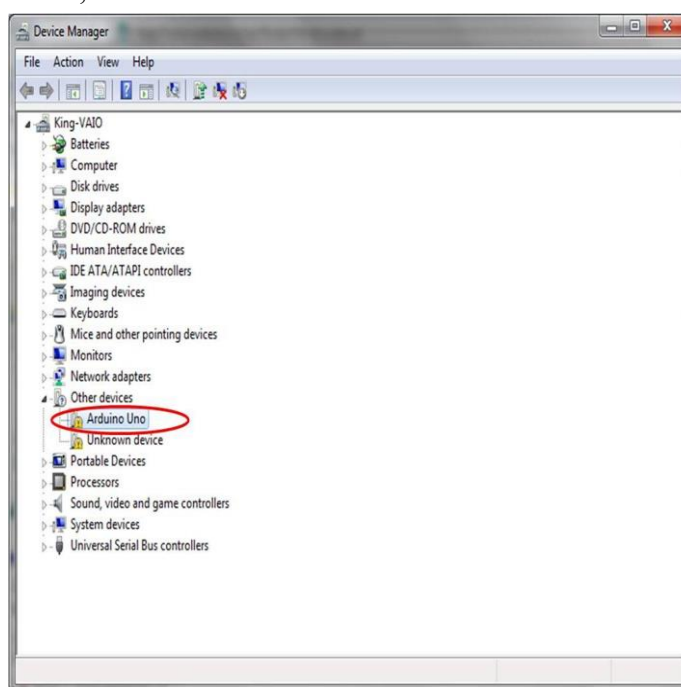


Рис. 1.3. COM-порт

для Arduino в Диспетчері пристроїв

1.3. Інтерфейс інтегрованого середовища розробки IDE

IDE (від англ. Integrated Development Environment - інтегроване середовище розробки) - це додаток або група додатків, призначених для створення, налаштування, тестування і обслуговування програмного забезпечення. Інтегроване середовище розробки включає до

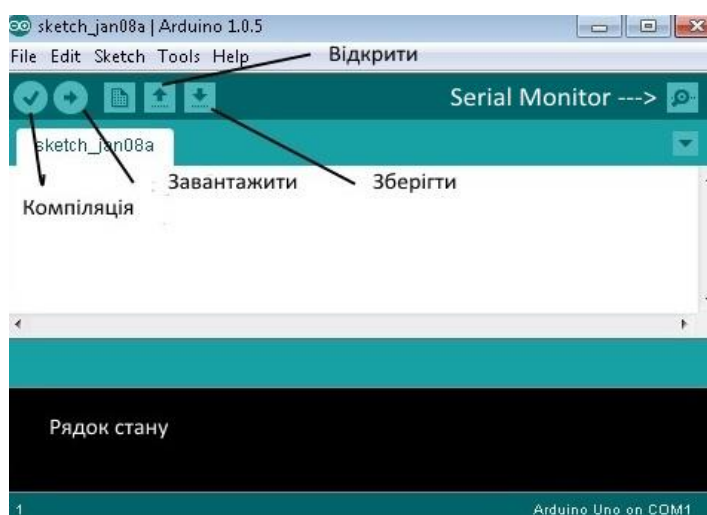
себе редагування і компіляція вихідного коду, створення програмних ресурсів, створення баз даних і т.д.

Інтерфейс середовища розробки Arduino містить такі основні елементи: текстовий редактор для написання коду, область для виведення повідомлень, текстова консоль, панель інструментів із традиційними кнопками і головне меню. Дане середовище дозволяє комп'ютеру взаємодіяти з Arduino як для передачі даних, так і для прошивки коду в контролер.

Програми, що створюються в середовищі розробки Arduino (скетчи), пишуться в текстовому редакторі і зберігаються в файлах з розширенням .ino. Вбудований текстовий редактор має стандартні інструменти копіювання, вставки, пошуку і заміни тексту. Область повідомлень у вікні програми є, свого роду, зворотним зв'язком для користувача, що інформує його про події (в тому числі і про помилки), що виникають в процесі запису або експорту написаного коду. Консоль відображає у вигляді тексту потік вихідних даних середовища Arduino, включаючи всі повідомлення про помилки та ін. У нижньому правому куті вікна програми показується модель поточної плати і послідовний порт, до якого вона підключена. Кнопки на панелі інструментів призначені для створення, відкриття, збереження і прошивки програм в плату Arduino. Окрема кнопка запускає програму SerialMonitor. Структура Інтерфейсу середовищі IDE містить такі пункти (рис. 1.4):

Рис. 1.4. Інтерфейс середовища IDE

Додаткові в меню: File, Edit, U цих меню тільки ті пункти, застосувати до або фрагменту деякі з них, з зустрічається проектів. 3 будемо



команди знаходяться Sketch, Tools і Help. завжди активні які можна поточного елементу коду. Відзначимо якими найчастіше розробник Arduino-ншими опціями знайомитись

пізніше. Найбільш важливим елементом опції «Інструменти» є можливість вибору відповідної плати, у нашому випадку Arduino Uno. Взагалі у списку знаходяться всі офіційні версії Arduino. В меню «Інструменти» також можна встановити послідовний COM-порт, до якого підключена плата Arduino. Найчастіше пакет Arduino IDE сам визначає COM-порт, але іноді потрібно вручну встановити номер порту в налаштуваннях.

Дуже корисною особливістю програми є вбудований набір прикладів програм, який можна побачити в меню в опції **File**. Це насправді зручно, тому що приклади програм можна відразу перевірити, завантаживши їх в мікроконтролер. При необхідності можна зберегти приклад і змінити його відповідно до потреб користувача. Внаслідок набуття величезного досвіду використання Arduino багатьма авторами були створені проблемно-орієнтовані бібліотеки, які можна підключати до власного проекту. Підключення бібліотек виконується в меню в опції **Sketch**. Більше докладно щодо структури середовища Arduino IDE надано в [https://doc.arduino.ua/ru/guide/Environment].

РОЗДІЛ 2 ОСНОВИ ПРОГРАМУВАННЯ ARDUINO

2.1. Структура програми Arduino

Як було показано вище, програмування плати Arduino відбувається в середовищі IDE Arduino. Програма, що написана в цьому середовищі, має назву *скетч* (Sketch). Кожний скетч містить щонайменше два оператора `setup()` та `loop()` :

```
void setup() { послідовність інструкцій; }  
void loop() { послідовність інструкцій; }
```

На початку програми, перед функцією `setup()`, зазвичай оголошуються змінні, які будуть використовуватися у програмі. Після включення живлення плати першим виконується оператор `setup()`, який використовується для ініціалізації змінних та визначення режимів роботи портів (ввод або вивід). Цей оператор також використовується для запуску потрібних бібліотек, що містять велику кількість скетчів для управління тими чи іншими елементами електронних пристрів. Оператор `setup()` виконується лише один раз після кожної подачі живлення чи скидання плати Arduino.

Після завершення роботи оператора `setup()`, починає виконуватися код (текст програми), написаний в тілі оператора `loop()`. Важливо уявити, що оператор `loop()` виконується у нескінченному циклі, тобто знову і знову, чекаючи сигнали на виходах або опитуючи виходи елементів пристрів та виконуючи різні обчислення.

Написання тексту скетчу має відбуватися згідно синтаксису програми, тобто певних вимог до форматування, які є дуже простими.

`//` - (однострочковий коментар). Часто використовується для пояснення змісту кожного рядка програми. Все, що розміщено після подвійний риси і до кінця рядка буде ігноруватися компілятором, спеціальною програмою, яка створює виконуваний файл з програми, що написана мовою програмування.

`{ }` - (фігурні дужки). Використовуються для визначення початку і кінця блоку команд (використовуються у функціях і циклах).

`/* */` - (багаторядковий коментар). Ця структура використовується для розширених коментарів, що займають більше одного рядка. Все, що знаходиться між цими символами буде ігноруватися компілятором.

`;` - (крапка з комою). Кожна команда повинна закінчуватися цим символом, відсутність якої - найбільш поширена помилка, що приводить до неможливості компіляції!).

Будь-яка програма використовує дані (напруга, стан тих чи інших параметрів тощо). У мові Wiring існують різні типи даних, серед яких найчастіше використовуються такі типи даних (повний перелік типів даних наданий за адресою http://arduino.net.ua/Arduino_articles/):

int - цілочисельний тип даних, який займає 2 байта і може приймати значення від -32 768 до 32 767;

boolean - може приймати одне з двох станів - ***true*** або ***false*** (хоча і займає цілий байт ОЗУ);

float - тип даних для зберігання чисел з плаваючою точкою. Числа з плаваючою точкою мають набагато більшу роздільну здатність ніж цілочисельні змінні;

char - даний тип даних займає 1 байт ОЗУ і зберігає символ.

У тексті програми використовуються велика кількість стандартних операторів і функцій. Зокрема, в операторі `setup()` найчастіше використовуються функції `pinMode(pin, OUTPUT)` та `pinMode(pin, INPUT)`, які визначають режим цифрових та аналогових портів (чи вони працюють або як `OUTPUT` (на цей порт подаються ті чи інші дані), або як `INPUT` (з цього порту зчитуються ті чи інші дані)).

В операторі `loop()` використовуються стандартні функції, наприклад `digitalWrite(pin, value)` або `digitalRead(pin, value)`, що означає, що на цифровий контакт `pin` (від 2 до 13) подається (або зчитується) потенціал `value`: або `HIGH` (5 В) або `LOW` (0 В). Аналогічні функції `analogWrite(pin, value)` або `analogRead(pin, value)` стосуються цифрових портів з позначкою ~ (ІІМ хвиля), яка видає аналогову величину (або вона зчитується) на порт входу/виходу.

Стандартні функції не охоплюють всі варіанти розробки Arduino-проектів. Велика кількість користувачів плати Arduino прагнуть розширити можливості плати, створюючи власні функції за допомогою додаткових бібліотек. Бібліотека - це набір функцій, призначених для того, щоб максимально спростити роботу з різними датчиками, приладами чи пристроями: ЖК-екранами, сервомоторами, кроковими двигунами, іншими модулями. Наприклад, вбудована бібліотека `LiquidCrystal` дозволяє легко взаємодіяти з символьними LCD-екранами. Знайти вбудовану бібліотеку можна у списку, який відкривається через опції “Скетч” > “Підключити бібліотеку”.

Існують два способи підключення бібліотек. По-перше, бібліотеку можна додати безпосередньо у середовищі Arduino IDE з офіційного репозиторію Arduino (репозиторій - це місце, де зберігаються будь-які дані). Для цього треба зайти в меню Arduino IDE опції “Скетч” > “Підключити бібліотеку” > “Управляти бібліотеками”(Рис. 2.1.).

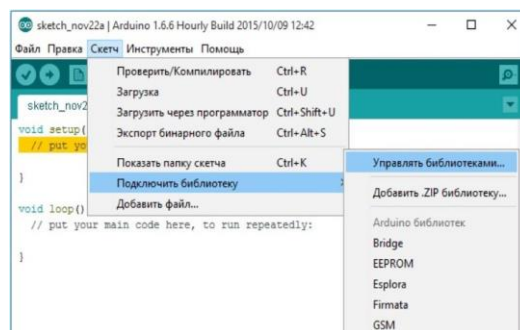


Рис. 2.1. Підключення бібліотеки з репозиторію

Після цього відкривається вікно пошуку (менеджер бібліотек) зі списком бібліотек, які або вже встановлені (позначка (INSTALLED)), або які можна встановити (через **info**). Для спрощення передбачено вказати тип та тему пошуку в якому можна знайти назвати тип, пошуку, тему пошуку або відфільтрувати запитання (Рис.2.2.). Якщо це зроблено, статус бібліотеки змінюється на INSTALLED, її можна підключити через розділи меню “Скетч” > “Підключити бібліотеку” та знайти її у списку.

Існує інший спосіб підключення бібліотеки у вигляді у вигляді упакованої у zip-архів папки, яка є ім'ям бібліотеки. Після завантаження бібліотеки у вигляді zip-архива, його не треба розпаковувати. Далі потрібно вказати шлях до zip-файлу бібліотеки. Цей zip-файл буде розпакований і поміщений в папку `Libraries` в директорію зі скетчами Arduino. Після установки бібліотека стає доступною через меню “Скетч” > “Підключити бібліотеку”. . Важливо

відзначити, що більшість бібліотек містять відповідні приклади використання, але для того, щоб вони стали доступними в меню “Файл” → “Зразки”, необхідно перезапустити оболонку Arduino IDE.

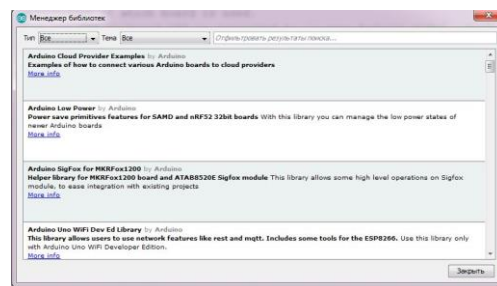


Рис. 2.2. Пошук бібліотеки в репозитарію

2.2. Основні функції та оператори мови програмування у середовищі IDE

Функції задання режиму роботи портів та зчитування (передача) даних.

З основними операторами мови Wiring середовища IDE краще ознайомитися на прикладі реальних проектів. В якості такого прикладу наведемо найпростіший скетч, який програмує миготіння, тобто включення та виключення світлодіоду. Нагадаємо, що світлодіод являє собою схемотехнічний елемент, який пропускає електричний струм лише в одному напрямку і при цьому спалахує (Рис. 2.3.)

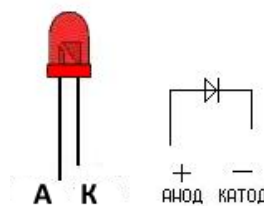


Рис. 2.3. Світлодіод та його електрична схема

Тут "К" позначає катод (негативний вивід, або більш коротка ніжка; "А" позначає анод (позитивний вивід), довга ніжка. Треба уважно слідкувати за підключенням світлодіоду:, якщо він підключений неправильно, він не буде горіти.

Теорія.

Світлодіод - далі "СВД" (світловипромінювальний діод), являє собою напівпровідниковий прилад, який за певних умов починає світитися. Ніякого зв'язку з дідовою лампою розжарювання він не має, в ньому немає нитки розжарювання, немає вакуумної колби. СВД влаштований так: два притиснутих один до одного кристала з певними добавками, до них прироблені дві контактні ніжки і все це залито оргстеклом. При подачі напруги на ніжки, частинки з двох кристалів спрямовуються один до одного і при зіткненні виділяють фотони, тобто виділяють світло. У сучасних яскравих СВД місце навколо кристала покривають люмінесцентною речовиною, яке теж додає яскравості. СВД, перш за все, є діодом (головна властивість діода - пропускати струм тільки в одному напрямку), тому запалюється він тільки при правильному підключенні полярності.

Підключимо світлодіод до цифрового контакту 13 плати Arduino. Зауважимо, що світлодіоди треба підключати до цифрових виводів через резистор, номіналом 220-330 Ом для обмеження струму, інакше світлодіод перегорить. Але вивід 13 має внутрішній резистор для обмеження струму, тому на рис. 2.4. резистор відсутній.

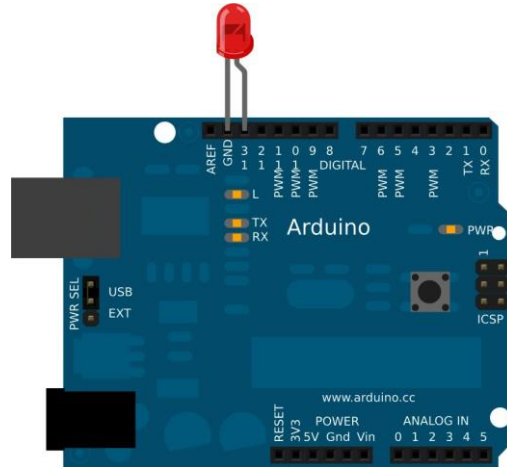


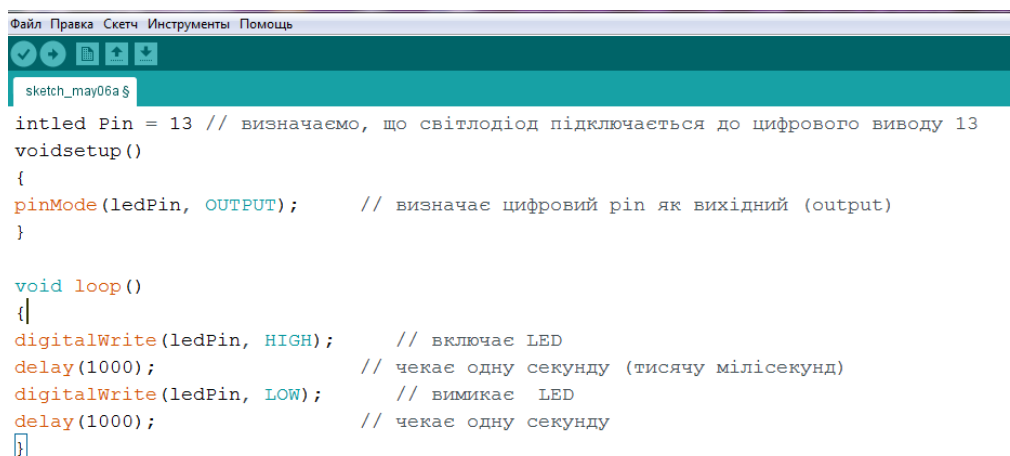
Рис.2.4. Підключення світлодіоду до плати Arduino

Тепер вже можна навести текст скетчу, який програмує виконання миготіння світлодіоду.

```
/* Завдання проекту: включити на одну секунду світлодіод, розташований на цифровому контакті 13 плати Arduino, потім вимкнути його також на одну секунду */
void setup()
{
    // Ініціалізуємо цифровий вхід/вихід у режимі виходу.
    pinMode(13, OUTPUT); // pinMode – це функція, яка задає режим OUTPUT
}
// починає працювати нескінченна функція loop()
void loop() {
    digitalWrite(13, HIGH);
    // функція digitalWrite() включає світлодіод (HIGH – це подача напруги 5 в)
    delay(1000);          // чекаємо одну секунду (1000 мілісекунд)
    digitalWrite(13, LOW);
    /* тепер функція digitalWrite() виключає світлодіод (LOW – це зняття напруги, тобто подача на контакт 13 нульової напруги) */
    delay(1000);          // чекаємо одну секунду.
}
```

Для виконання цього проекту треба двічі клацнути на позначку Arduino та обрати в меню IDE опцію "File">"New". Після треба назвати ім'я папки зі скетчами: це місце, де буде зберігатися скетч Arduino. Після треба ввести ім'я скетчу Blinking_LED (або якимось іншим) і натиснути "OK". Потім треба набрати підготовлений текст скетчу у головному вікні IDE Arduino або скопіювати відповідний текст, представлений вище, та вставити його у тому ж

вікні. Текст скетчу також можна завантажити в опції “File”> “Примеры”> “01. Basic”> “Blink”. Тоді вкно редагування має такий вигляд (Рис.2.5):



```
Файл Правка Скетч Инструменты Помощь
sketch_may06a $
int ledPin = 13 // визначаємо, що світлодіод підключається до цифрового виводу 13
void setup()
{
  pinMode(ledPin, OUTPUT); // визначає цифровий pin як вихідний (output)
}

void loop()
{
  digitalWrite(ledPin, HIGH); // включає LED
  delay(1000); // чекає одну секунду (тисячу мілісекунд)
  digitalWrite(ledPin, LOW); // вимикає LED
  delay(1000); // чекає одну секунду
}
```

Рис. 2.5. Середовище IDE розробки Arduino зі скетчем

Тепер треба перевірити правильність написання тексту скетчу, тобто відповідність вимогам синтаксису. Для цього треба натиснути кнопку "Проверить": якщо помилок немає, внизу віконця буде повідомлення "Компиляция завершена". Це повідомлення говорить про те, що IDE Arduino відтранслявав ваш скетч у виконувану програму, яка може бути запущена на платі, майже як .exe-файли в Windows.

Тепер можна вивантажити скетч на плату, для чого треба натиснути кнопку "Загрузка" (завантажити у плату Arduino). Відбудеться перезапуск плати, який змушує плату зупинити виконання коду і слухати інструкції по порту USB. IDE Arduino відправляє поточний скетч на плату, яка зберігає його в своїй пам'яті і в кінці кінців виконує його. Можна побачити кілька повідомлень в чорній області внизу екрану IDE, і прямо над цією областю виникає повідомлення "Done uploading". Це означає, що процес вивантаження успішно завершений.

На платі встановлено два світлодіоди, що позначені «RX» і «TX»; вони блимають щоразу при відправленні або отриманні байта даних платою. Під час завантаження вони мерехтять. Якщо світлодіоди не мерехтять, або у повідомленні йдеться про помилку замість "Компиляция завершена", значить, існує проблема зв'язку між вашим комп'ютером і Arduino. Спочатку треба переконайтеся, що обраний вірний COM-порт в меню "Инструменты"> "Последовательные порты". Також слід перевірити пункт меню "Инструменты"> "плата" - в ньому повинна бути обрана вірна модель Arduino.

Після того, як код був завантажений в плату Arduino, він буде залишатися в ній до тих пір, поки не буде завантажений інший скетч. Скетч залишиться на місці, якщо плату буде перезапущено або вимкнено, майже як на жорсткому диску комп'ютера. Якщо скетч був завантажений успішно, тоді світлодіод, підключений до 13 цифрового контакту включається на одну секунду, а потім на той же час вимикається. Одночасно буде блимати внутрішній світлодіод "L", розташований на платі.

Те, що тільки-що написано і запущено, і є "комп'ютерна програма". Таким чином, як визначалося раніше, плата Arduino, - це невеликий комп'ютер, до якого можна завантажити

будь-яку програму та примусити плату робити те, що хоче автор. Іншими словами, програмування відбувається за допомогою написання послідовності інструкцій мовою програмування в середовищі розробки Arduino IDE, як перетворює цю послідовність в виконуваний код для плати Arduino.

У наведеному прикладі використовувались лише дві функції: `pinMode()` та `digitalWrite()` зі своїми аргументами. Зрозуміло, що у мові програмування Wiring, яка насправді є діалектом мови C/C++, є багато стандартних функцій, які призначені для різних операцій з даними. З повним переліком стандартних функцій платформи Arduino можна ознайомитися за адресою <https://doc.arduino.ua/ru/prog/>. У тексті скетчів Arduino присутні такі стандартні оператори як оператори циклу (`for` та `while`), умовні оператори (`if ... else`), логічні оператори, оператори порівняння, математичні оператори та інші, які ми розглянемо у наступних прикладах.

Зі скетчу видно, що світлодіод у нас блимає щосекунди на протязі хвилини (для кожного разу – 2 сек, всього 30, тобто одна хвилина), після чого півхвилини відпочиває і знову блимає хвилину.

Миготіння світлодіоду у циклі *for*

Схема підключення світлодіоду залишається (Рис.2.4), але трохи складніше: світлодіод має блимати щосекунди на протязі хвилини (для кожного разу – 2 сек, всього 30, тобто одна хвилина), після чого чверть хвилини не горить. Наведемо текст відповідного скетчу.

```
void setup ()
{
  pinMode (13, OUTPUT); // призначаємо порт 13 виходом як вихідний
}

void loop () // основний цикл програми
{
  / * створюємо цикл for, де змінній k (параметр циклу) типу int (ціле значення) присвоєно
  початкове значення 0 і створено правило перевірки виходу з циклу, яке свідчить, що цикл
  закінчиться, як тільки k дорівнюватиме 30. При цьому у циклі параметру k збільшує своє
  значення на 1 за допомогою лічильника (k++) при кожному зверненні до нього. * /

  for (int k = 0; k < 30; k ++)
  {
    digitalWrite (13, HIGH); // Включаємо світлодіод, подаючи на 13 вивід + 5 вольт
    delay (1000); // Чекаємо 1000 мілісекунд або 1 секунду
    digitalWrite (13, LOW); // Вимикаємо світлодіод, подаючи на 13 вивід 0 вольт
    delay (1000); // Чекаємо 1000 мілісекунд або 1 секунду
  }
  delay (15000); // чекаємо чверть вилини - так помітніше.
```

```
}
```

Миготіння світлодіоду у циклі *while*

Цей проект практично такий як попередній, але, за допомогою циклу *while*.

```
int k = 0; // Оголошуємо змінну k типу int та присвоюємо їй початкове значення 0
void setup ()
{
  pinMode (13, OUTPUT);
}
void loop ()
{
  while (!k == 30)
  /*
цією фразою створюється цикл, де є правило перевірки виходу з циклу: цикл завершується,
коли k дорівнюватиме 30.
*/
  {
    digitalWrite (13, HIGH); // Включаємо світлодіод, подаючи на 13 порт 5 вольт
    delay (1000); // Чекаємо +1000 мілісекунд або 1 секунду
    digitalWrite (13, LOW); // Вимикаємо світлодіод, подаючи на 13 порт 0 вольт
    delay (1000); // Чекаємо +1000 мілісекунд або 1 секунду
    k ++; // створюємо лічильник, тобто при кожному зверненні до нього k збільшується на 1.
  }
  delay (15000); // чекаємо чверть хвилини.
}
```

РОЗДІЛ 3. БАГАТОФУНКЦІОНАЛЬНА ПЛАТА (MULTIFUNCTION SHIELD)

Концепція програмно-апаратної платформи Arduino передбачає відкритість системи, що зумовило розробку значної кількості плат доповнення (shield), які функціонально розширюють можливості головної плати Arduino. Плата розширення Ардуіно - це закінчений пристрій, який призначений для виконання певних функцій, він підключається до основної плати Arduino за допомогою стандартних роз'ємів. На платі розширення встановлені всі необхідні електронні компоненти, а взаємодія з мікроконтролером і іншими елементами основної плати відбуваються через стандартні контакти (порти) Arduino. Найчастіше живлення плати доповнення теж подається з основної плати Arduino, хоча в багатьох випадках є можливість живлення з інших джерел. У будь-якому у платі залишається кілька вільних контактів, які можна використовувати на свій розсуд, підключивши до них будь-які інші компоненти.

Плати доповнення не тільки розширяють можливості Arduino, вони значно скорочують час зборки будь-якого проекту, одночасно знижуючи ризики її зипсування внаслідок будь-яких помилок. Зрозуміло, що плати доповнення слід обирати із врахуванням типу мікроконтролеру. Для підключення плати потрібно просто акуратно «надіти» її на основну плату. Зазвичай контакти плати типу гребінки (тато) легко вставляються в роз'єми плати Arduino. У деяких випадках потрібно акуратно підправити штиркі, якщо сама плата спаяна неакуратно. Тут головне діяти акуратно і не додаватися зайвої сили.

Сьогодні існує безліч шілд (плат доповнення), більше того, маючи певні знання та вміння самостійно створювати друковані плати з метою розроблення власного унікального проекту. Найбільш популярним прикладами таких плат є плати розширення для роботи з датчиками, двигунами, LCD-екранами, SD-картами, мережеві і GPS-плата, плата з вбудованими реле для підключення до навантаження та багато інших.

У даному посібнику використовується плата розширення **multi-fubction shield** - багатофункціональна плата (БПФ) (Рис. 4.1.).



Рис. 3.1. Багатофункціональна плата (multi-fubction shield)

Багатофункціональна плата доповнення (БФП) призначена для ознайомлення з платформою Arduino та створення Arduino-проектів для управління периферією. Плата дозволяє отримати практичний досвід щодо управління семисегментним індикатором, аналоговими і цифровими датчиками температури, кнопками, bluetooth-модулями, і т.д. Плату можна використовувати у навчальному процесі на уроках природничих дисциплін, інформатики, на STEM-уроках, але можна використовувати й у закладах позашкільної освіти для розробки власних проектів. Схема плати проста у використанні, скорочує час зборки проекту і зводить до мінімуму ризики виходу з ладу елементів при їх неправильному підключенні. Плата розширення розташовується на основній платі (Рис. 3.2.).

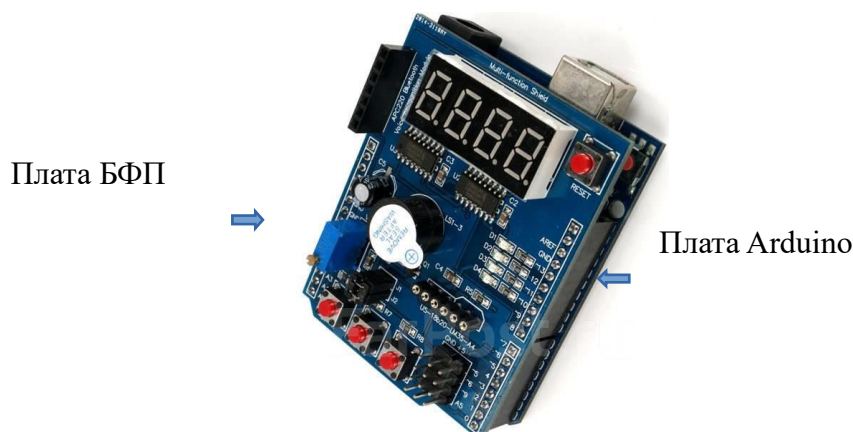


Рис.3.2. Збірка модуля “багатофункціональна плата 0 Arduino”

Характеристика багатофункціональної плати:

Напруга живлення: 5В;

Пристрої вводу: тактові кнопки (A1, A2, A3);

Інтерфейс введення:

Цифровий: DS18B20 (A4) (з відключається резистором);

Аналогові: LM35 (A4);

Аналогові: змінний резистор (A0);

Інтерфейс виводу:

Звуковий випромінювач (D3) (з додатковим транзистором);

Світлодіоди (D10, D11, D12, D13);

Семисегментного світлодіодна матриця (SPI);

Bluetooth роз'єм APC220;

Роз'єм для підключення голосового модуля APC220;

Додаткові інтерфейси:

Додатковий роз'єм живлення (Vcc, Gnd);

Роз'єми для підключення сервоприводів (D5, D6, D9);

Роз'єм для підключення аналогових датчиків (A5);

Роз'єм для підключення інфрачервоного датчика (D2);

Для роботи з Multi-function Shield необхідно встановити наступні бібліотеки: TimerOne.h, Wire.h та MultiFuncShield.h. Процес установки бібліотек викладено у підрозділі 2.2.

Розглянемо декілька базових, навчальних проектів, які використовують багатофункціональну плату. Процес підготовки плати Arduino до експериментів (налаштування типу плати та послідовного порту зв'язку плати Arduino з комп'ютером) та завантаження скетчів викладено раніше у Розділі 1.

а). Миготіння всіх чотирьох світлодіодів

Почнемо з найпростішого проекту — миготіння всіх чотирьох світлодіодів плати, які підключені до контактів D10, D11, D12, D13 (на платі БФП позначені як D1,D2,D3,D4). Нагадаємо, що ці контакти плати електрично з'єднані з аналогічними контактами плати Arduino, як це видно з Рис. 3.2.

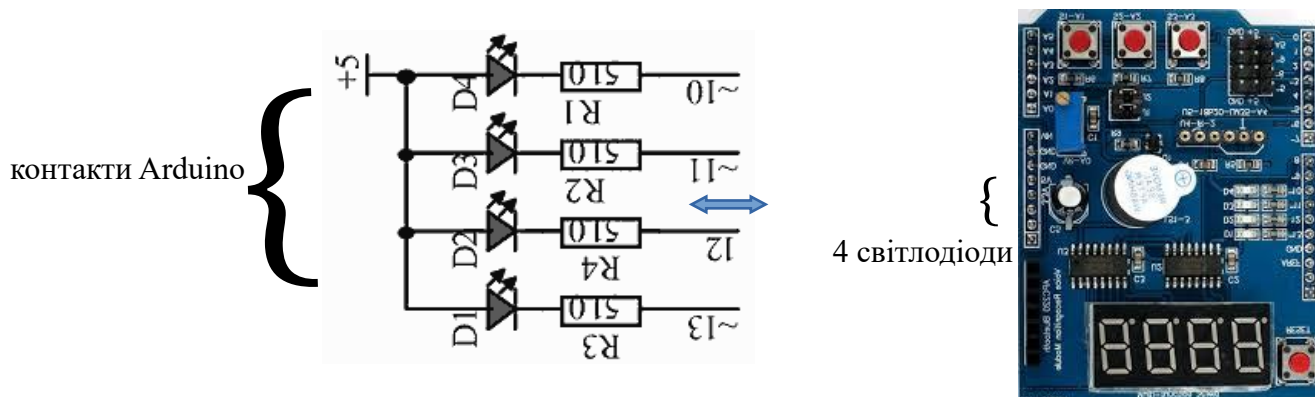


Рис.3.3. Чотири світлодіоди багатофункціональної плати та їх електрична схема

Ніже представлений текст скетчу проекту миготіння світлодіодів БФП. Звернемо увагу на те, що у даному проекті не використовуються підключені бібліотеки, тому, що у цьому немає потреби.

```
int led1 = 13;
int led2 = 12;
int led3 = 11;
int led4 = 10;

void setup()
{
  // цифрові контакти ініціалізуємо як вихідні (OUTPUT)
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
}
```



```

void loop()
{
digitalWrite(led1, HIGH);
digitalWrite(led2, HIGH);
digitalWrite(led3, HIGH);
digitalWrite(led4, HIGH);
delay(1000);
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
delay(1000);
}

```

У цьому скетчі використовуються ті ж самі оператори void setup() і loop() та функції, які використовувались у прикладі підрозділу 2.1: pinMode(), digitalWrite() із відповідними аргументами та функція затримки delay(1000). Світлодіоди включаються одночасно на 1 сек., потім гаснуть також на 1 сек. І поділ у нескінченному циклі loop().

б) Миготіння світлодіодів по черзі

Однак якщо підключити бібліотеки, які створені для цієї плати: TimerOne.h, Wire.h та MultiFuncShield.h, тоді можна легко використовувати інші комбінації перемикання світлодіодів. Розберемо скетч такого проекту.

```

#include <TimerOne.h>           // Підключити бібліотеку TimerOne
#include <Wire.h>               // Підключити бібліотеку Wire
#include <MultiFuncShield.h>    // Підключити бібліотеку Multifunction shield

void setup()
{
  Timer1.initialize();          // ініціювати Таймер 1
  MFS.initialize(&Timer1);      // ініціювати бібліотеку БФП
}

void loop()
{
  MFS.writeLeds(LED_ALL, ON);   // підключити всі світлодіоди
  delay(1000);                  // затримка 1 сек.
  MFS.blinkLeds(LED_1, ON);     // підключити світлодіод 1
  delay(1000);                  // затримка 1 сек.
  MFS.blinkLeds(LED_2, ON);     // підключити світлодіод 2
}

```

```

delay(1000);                                // затримка de 1 сек.
MFS.blinkLeds(LED_1 | LED_2, OFF);          // погасити світлодіоди 1 та 2
MFS.blinkLeds(LED_3 | LED_4, ON);           // погасити світлодіоди 3 та 4
delay(1000);                                // затримка 1 сек.
MFS.blinkLeds(LED_ALL, ON);                 // підключити світлодіоди
delay(1000);                                // затримка 1 сек.
MFS.blinkLeds(LED_ALL, OFF);                // вимкніть миготливі світлодіоди
MFS.writeLeds(LED_ALL, OFF);                // вимкнути світлодіоди
delay(1000);                                // затримка 1 сек.
}

```

ВТОЧНИТИ blinkLeds та writeLeds

в) Перемикання світлодіодів кнопками

Перемикання на платі відбувається тактовими кнопками S1, S2, S3 (Рис.3.4.а), які з'єднані з відповідними контактами плати Arduino A1,A2,A3. Схему підключення кнопок представлено на Рис.3.4.б. Всі кнопки підключені через 10K підтягуючий резистор, тобто рівень напруги на контактах буде 5 В (HIGH). При натисканні кнопки рівень напруги буде нульовим (LOW). Кнопка RESET використовується для перезапуску програми в Arduino. Кнопки S1, S2 та S3 підключені до аналогових вхідних портів A1, A2 та A3 відповідно.

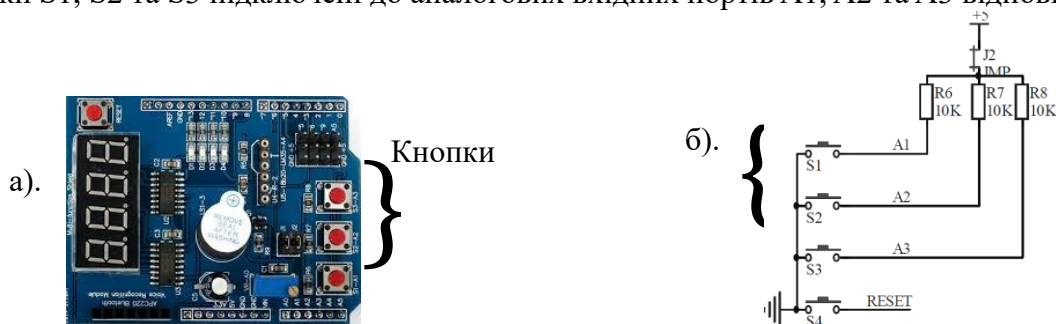


Рис.3.4. Тактові кнопки (а) та схема їх підключення (б)

Нижче представлено відповідний скетч, який вмикає та вимикає світлодіоди

```

int ledpin1 = 10; //створюємо змінну ledpin1 і пов'язуємо її з вивідом 10
int ledpin2 = 11; //створюємо змінну ledpin2 і пов'язуємо її з вивідом 11
int ledpin3 = 12; //створюємо змінну ledpin3 і пов'язуємо її з вивідом 12
int inpin1=A1; //створюємо змінну inpin1 і пов'язуємо її з вивідом A1
int inpin2=A2; //створюємо змінну inpin2 і пов'язуємо її з вивідом A2
int inpin3=A3; //створюємо змінну inpin3 і пов'язуємо її з вивідом A3

int val;      //створюємо змінну val

void setup(){
pinMode(ledpin1,OUTPUT);    //встановлюємо вивід 10 як вихід
pinMode(ledpin2,OUTPUT);    //встановлюємо вивід 11 як вихід
pinMode(ledpin3,OUTPUT);    //встановлюємо вивід 12 як вихід

```

```

pinMode(inpin1,INPUT);           // встановлюємо вивід A1 як вихід
pinMode(inpin2,INPUT);           // встановлюємо вивід A2 як вихід
pinMode(inpin3,INPUT);           // встановлюємо вивід A3 як вихід
}

void loop(){
val=digitalRead(inpin1);         //зчитуємо дані порту A1 (змінна inpin1) в змінну val
if(val==LOW)                     // виконуємо умову, якщо в змінну записано 0, тоді гасимо світлодіод
{ digitalWrite(ledpin1,LOW);}
else // інакше вмикаємо
{ digitalWrite(ledpin1,HIGH);}

val=digitalRead(inpin2);         //зчитуємо дані порту A2 (змінна inpin2) в змінну val
if(val==LOW)                     //виконуємо умову, якщо в змінну записано 0, тоді гасимо світлодіод
{ digitalWrite(ledpin2,LOW);}
else // інакше вмикаємо
{ digitalWrite(ledpin2,HIGH);}

val=digitalRead(inpin3);         //зчитуємо дані порту A3 (змінна inpin3) в змінну val
if(val==LOW)                     //виконуємо умову, якщо в змінну записано 0, тоді гасимо світлодіод
{ digitalWrite(ledpin3,LOW);}
else // інакше вмикаємо
{ digitalWrite(ledpin3,HIGH);}
}

```

У цьому скетчу зустрічаємо деякі елементи мови програмування. По-перше, у скетчу оголошуються змінні `int1`, `int2`, `int3`, які мають цілочисельний тип. Крім цього типу у мові існують інші типи, зокрема `boolean`, `byte`, `char`, `float` та інші. Звернемо увагу на функцію `pinMode(inpin1,INPUT)`. Тут вперше зустрічаємо оголошення статусу порту A1 як `INPUT`; це означає, що вивід A1 налаштовується на зчитування з цього аналогового порту даних, а саме: чи натиснуто кнопку S1, яка розташована на БПФ. Якщо кнопку натиснуто, тоді на вході порту A1 з'явиться 5 В (див. Рис. 3.4. б), якщо кнопка вільна, то вхід порту A1 з'єднується із землею (0 В). Також у цьому скетчу використовується змінна `val`, тому що після зчитування стану кнопки з виводу A1 отримане значення цього статусу треба зафіксувати з метою подальшого використання. Ось, далі у фрагменті `if(val==LOW)` й використовується ця змінна: чи вона дорівнює `LOW`, тобто чи вона є вільною.

г). Генерація звуку через вбудований зумер

Цей скетч дозволяє використовувати кнопку користувача для генерації звуку через вбудований зумер (Рис.3.5 а.) .

Теорія.

Зумер являє собою невеликий пристрій, побудований на основі п'єзоелектричного кристала. Фізична дія його заснована на зворотному п'єзоелектричному ефекті, який полягає у виникненні поляризації, яке супроводжується механічними деформаціями. Тому, якщо на металеві обкладки, укріплені на кристалі, подати електричну напругу, то кристал під дією поля поляризується, деформується й видає звук. На БПФ встановлено зумер досить гучний, він підключений до

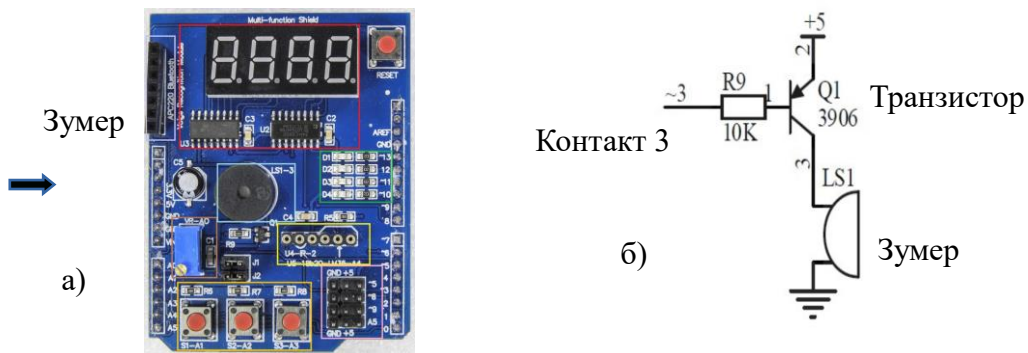


Рис. 3.5. Схема підключення зумера

Управління роботою зумера надається у такому скетчі.

```
#define ON LOW           //задаємо включення зумера
#define OFF HIGH        //задаємо виключення зумера
#define BUZZER 3        //підключаємо зумер до цифрового входу (контакту) 3
#define KEY1 A1 //      підключаємо кнопку до аналогового входу A1

void setup(){

  /* конфігуруємо виводи*/
  pinMode(KEY1, INPUT);      //кнопка A1 як вход
  pinMode(BUZZER, OUTPUT);   //зумер 3 як вихід
  digitalWrite(BUZZER, OFF); //вимикаємо звук зумера
}

void loop(){

  if( digitalRead(KEY1)==ON ) //тиснемо кнопку KEY1
  {
    digitalWrite(BUZZER, ON); //зумер включений
  }
  else{
    digitalWrite(BUZZER, OFF); //зумер виключений
  }
}
```

```

}
}

```

У цьому скетчі використовуються такі ж самі оператори і функції як і раніше за виключенням директиви `#define`. За допомогою цієї директиви можна ввести будь-яку константу (яка не змінюється впродовж виконання скетчу) та дати їй ім'я. У даному випадку директива `#define ON LOW` означає, що константі `.ON` присвоюється ім'я `LOW`. Константи, по-перше, спрощують написання програм, а, по-друге, не вимагають пам'яті комп'ютера.

д). Використання потенціометру

У наступному проєкті будемо використовувати один з найбільш поширених елементів електроніки — потенціометр (Рис. 3.6 а.). Потенціометр — це регульований дільник електричної напруги, тобто змінний резистор. Являє собою, як правило, резистор з рухомих контактом (двигком). Потенціометри використовуються в якості регуляторів параметрів (гучності звуку, потужності, вихідної напруги і т. д.). На багатофункціональній платі використовується змінний резистор опором 10 кОм, середньою ніжкою підключений до аналогового входу A0. З метою зменшення будь-яких перешкод, що виникають з багатьох чинників (наприклад, від сусідніх елементів), на платі розташований згладжуючий конденсатор C1 (Рис. 3.6 б). Регулювання потенціометру відбувається за допомогою відповідної відвертки.

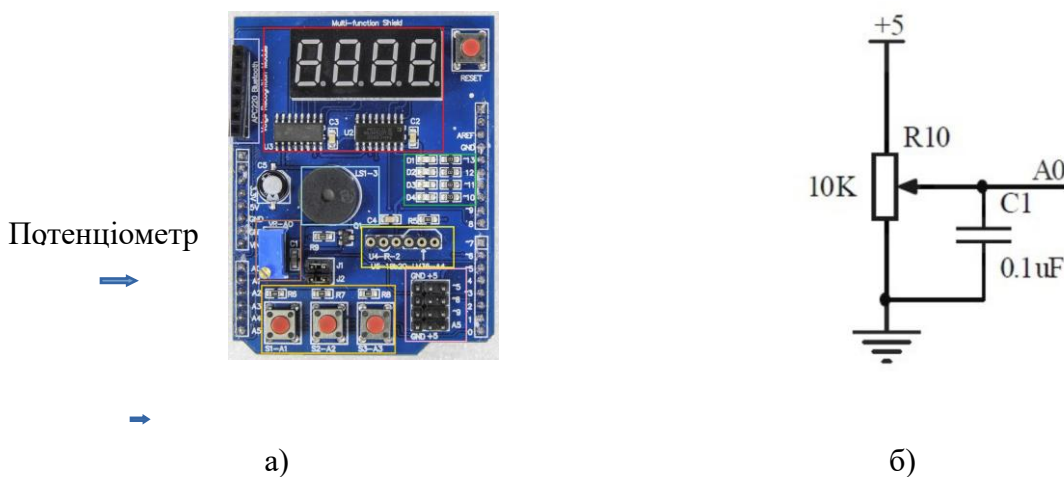


Рис. 3.6. Розташування потенціометру на платі (а) та його принципова схема (б)

Використання потенціометру для управління включенням світлодіодів в залежності від рівня напруги представлено у наступному скетчі.

```

#define Pot1 A0
int I1 = 13;
int I2 = 12;
int I3 = 11;
int I4 = 10;
void setup()
{
  Serial.begin(9600);
  // initialize the digital pin as an output.
  /* Set each pin to outputs */

```

```

pinMode(I1, OUTPUT);
pinMode(I2, OUTPUT);
pinMode(I3, OUTPUT);
pinMode(I4, OUTPUT);
}
/* Main Program */
void loop()
{
int PotValue;
//Serial.print("Potentiometer reading: ");
PotValue = analogRead(Pot1);
/* Чекати 0.5 сек. Перед новим зчитуванням*/
if(PotValue < 400)
{
digitalWrite(I1, LOW);
digitalWrite(I2, LOW);
digitalWrite(I3, LOW);
digitalWrite(I4, LOW);
Serial.print("Потенціометр: ");
Serial.println(PotValue);
}
else
{
digitalWrite(I1, HIGH);
digitalWrite(I2, HIGH);
digitalWrite(I3, HIGH);
digitalWrite(I4, HIGH);
Serial.print("Потенціометр: ");
Serial.println(PotValue);
}
delay(500);
}

```

Розглянемо докладно новації цього скетчу. Почнемо з функції `Serial.begin(9600)`, яка ініціює послідовне з'єднання комп'ютера з платою Arduino зі швидкістю 9600 біт на секунду через послідовний порт. Взагалі, набір функцій `Serial` служить для зв'язку пристрою Ардуіно з комп'ютером або іншими пристроями, що підтримують послідовний інтерфейс обміну даними. Обидві сторони послідовного з'єднання (тобто Arduino та ваш комп'ютер) повинні бути налаштовані на використання однакового швидкого послідовного з'єднання, щоб отримати будь-які зрозумілі дані. Якщо встановити неправильну швидкість, то замість даних отримаємо "сміття" - дані, які не можна обробити. Швидкість обміну може бути змінено, якщо це потрібно для проекту. Всі плати Arduino мають хоча б один послідовний порт (UART). Для обміну даними `Serial` використовують USB порт, а також цифрові порти вводу/виводу 0 (RX) і 1 (TX).

Середовище розробки Arduino IDE має вбудований монітор послідовного інтерфейсу (`Serial monitor`) (Рис. 3.7). Для початку обміну даними необхідно запустити монітор

натисканням кнопки Serial monitor і виставити ту ж швидкість зв'язку (baud rate), з якої викликана функція begin (). У вікні, що відкривається можна бачити дані, або відправляти їх

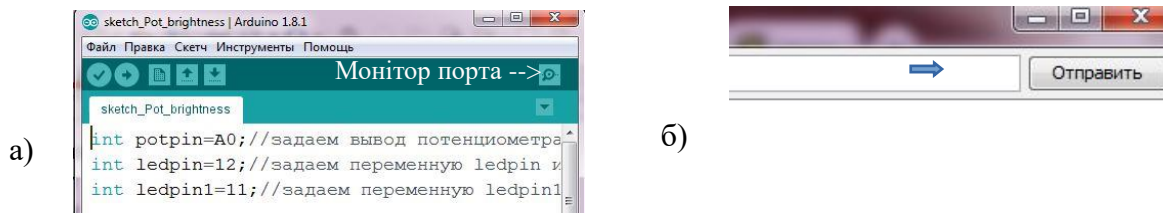


Рис. 3.7. Монітор порта (а) та вікно монітора (б)

Після цих пояснень зрозуміло, що функція Serial.print("Дані з потенціометру: ") просто друкує відповідний текст у вікні монітора. Функція Serial.println(PotValue) друкує значення, що прийшли з потенціометру через порт A0. Зауважимо, що тут використовується функція println(), яка друкує текст з перенесенням на новий рядок.

Тепер розберемо умовний оператор if(PotValue < 400). Його зміст полягає у наступному. Якщо значення PotValue, що прийшло з потенціометру через порт A0 менше 400, тоді виконуються наступні оператори і на всі світлодіоди подається напруга 0 В. В іншому випадку, тобто, якщо PotValue >= 400 (більше або дорівнює 400), на всі світлодіоди подається 5 В, тобто вони загораються.

Тепер, щодо значення 400. Нагадаємо, що значення, що зчитуються з аналогових портів перетворюються аналогоцифровим перетворювачем в значення від 0 до 1023. Тобто змінюючи положення потенціометру, напруга на ньому змінюється від 0 до 1023. Якщо крок зсуву складає $5\text{ В}/1023 \approx 0,0049\text{ В}$, тоді, після множення $0,0049 \cdot 400$ отримаємо значення 1,95 В, тобто 400 одиниць дорівнює десь 1,95 В.

До речі, монітор послідовного порту можна використовувати для вводу даних з метою управління компонентами проєкта. Зокрема, повернемося до проєкту із зумером, де він включався після натискання на кнопку. Розглянемо такий скетч.

```
int incomingByte;
int buzer=3;
void setup(){
  /* конфігурируємо виводи*/
  Serial.begin(9600);
  pinMode(buzer,OUTPUT); //зумер 3 як вихід
  digitalWrite(buzer, LOW); //зумер включений
}

void loop() {
  if (Serial.available() > 0) { //перевірка чи є дані на послідовному порту
    incomingByte = Serial.read();
    if(incomingByte == '1') //якщо до монітору відправили "1"
    {
      digitalWrite(buzer, HIGH); //зумер виключений
      delay(5000);
      Serial.println("Zummer OFF");
    }
  }
}
```

```

}
else if (incomingByte == '0'){
digitalWrite(buzzer, LOW); //зумер включений
Serial.println("Zummer ON");
delay(5000);
}
}
}
}

```

У цьому скетчі бачимо оператор `Serial.available() > 0`, який перевіряє чи є доступні дані у проміжному місці зберігання даних (буфері) послідовного інтерфейсу зв'язку. Це ті байти які вже надійшли і записані в буфер послідовного порту. У буфері може зберігатися до 64 байт. Якщо є, тоді виконується функція `Serial.read()`, яка зчитує дані (символ "1" або "0" з монітору порту. Якщо у вікні монітора було набрано та відправлено символ "1" (на зумер подається 5 В), тоді зумер виключається, якщо "0" - включається (на зумер подається 0 В).

Для управління яркістю світлодіодів можна використовувати потенціометр. Запишемо в скетч тільки два світлодіода, підключеного до пінам D12 і D11. Контакт D12 не має ШІМ (PWM), тому яскравість цього світлодіода буде змінюватися стрибком від 0 до 1 і навпаки. Контакт D11 може працювати у режимі широтно-імпульсної модуляції (ШІМ), що дозволяє плавно регулювати яскравість цього світлодіоду від 0 до максимального значення. Ївгвдуємо (п.1.2.3), що платформа Arduino встановлює на всіх виводах ШІМ частоту зміни сигналів 488,28 Гц та розділення 8 розрядів ($2^8 = 255$). Це означає, що напругу на цифрових ШІМ-виходах можна виставляти з кроком 0,196 вольт (5 вольт/255). А контакт D12 не має ШІМ (PWM).

```

int potpin = A0; // задаємо вивід потенціометра A0
int ledpin = 12; // задаємо змінну ledpin и приєднуємо до виводу 12 (без ШІМ)
int ledpin1 = 11; // задаємо змінну ledpin1 и приєднуємо до виводу 11 (з ШІМ)

int val = 0; // задаємо змінну val

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

void loop () {

val = analogRead (potpin); // зчитуємо дані порту в змінну val
Serial.println (val); // відправляємо значення змінної val в порт. (від 0 до 1023)
analogWrite (ledpin, val / 4); /* при обертанні потенціометра на ledpin значення буде змінюватись від 0 до 255.
Яскравість світлодіода змінюється стрибком 0 або 1. (немає ШІМ (PWM)) */
analogWrite (ledpin1, val / 4); /* при обертанні потенціометра на ledpin1 значення буде змінюватись від 0 до
255. Яскравість світлодіода змінюється плавно. (є ШІМ (PWM)) */

delay (100); // пауза 100мс
}

```


е) Управління 7-сегментним індикатором через зсувний регістр

Дані, вимірювані за допомогою входів плати Arduino, треба відображати будь-яким способом. Зрозуміло, що найпростішим шляхом для цього є використання функції `Serial.print()`, яка друкує дані у вікні монітора послідовного порту (Рис. 3.7.б). Однак набагато краще було б бачити результати вимірювання безпосередньо на платі. З цією метою розробники БПФ вмонтували на платі рідкокристалічний індикатор LCD, управління яким відбувається за допомогою двох зсувних регістрів (Рис. 3.8.).

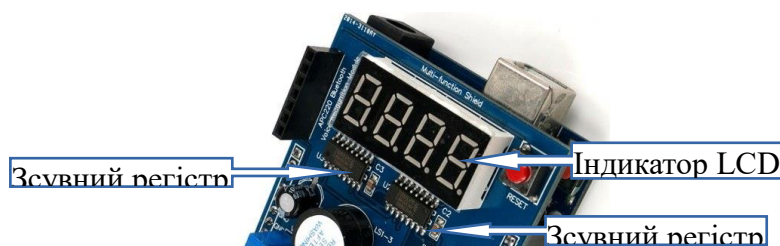


Рис.3.8. Рідкокристалічний індикатор LCD

Зсувний регістр 74HC595 з SPI інтерфейсом є одним з найпростіших регістрів, який перетворює послідовні дані у паралельні. Такі регістри використовуються в тих випадках, коли треба контролювати входи деяких елементів схемотехніки, чисельність яких перевищує можливості Arduino. Зокрема у рідкокристалічному індикаторі LCD дані представляються як сукупність чотирьох розрядів, кожен з яких має сім сегментів. А на платі БПФ (та відповідно на платі Arduino) виходів набагато менше. Крім того, деякі виходи вже схемотехнічно зайняті, зокрема, під світлодіоди. Саме тоді використовуються зсувні регістри, на вхід яких подається послідовність бітів (0 або 1), а на виході — 8 паралельних бітів. Тобто, 8 бітів (один байт) поступають на вхід зсувного регістру один за одним, відразу ж вони розподіляються на паралельні, тобто кожен з них потрапляє на свій вихід. Коли проходить останній біт спрацьовує засувка, і передача бітів припиняється. А сконфігуровані паралельні біти передаються на відповідні сегменти індикатора. Таким чином послідовність бітів на вході перетворюється на сукупність паралельних бітів на виході. У нашому випадку використовуються два зсувних регістри: один відповідає за дані розрядів, друга - за дані чисел (Рис. 3.9.):

- ніжка 14 мікросхеми підключена до контакту БПФ D8. Це лінія передачі даних (SDI). У скетчі позначено `DATA_DIO`. Саме на цей вхід подається послідовність бітів даних.
- ніжка 11 мікросхеми підключена до контакту D7. Це лінія тактирування (SFTCLK). У скетчі позначено `CLK_DIO`. Тут відбувається управління порядком надходження бітів.
- ніжка 12 мікросхеми підключена до контакту D4. Це лінія синхронізації даних, засувка (LCHCLK). У скетчі позначено `LATCH_DIO`.

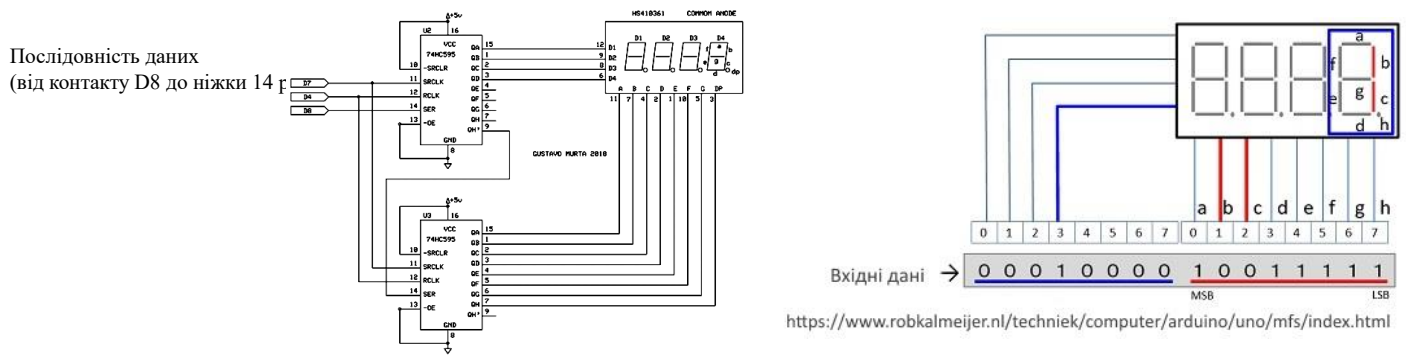


Рис. 3.9. Схема управління 4-розрядним 7-сегментним індикатором LCD

Ось так виглядає відповідний скетч:

```
#define LATCH_DIO 4 // визначаємо контакт D4 як LATCH_DIO (синхронізація даних)
#define CLK_DIO 7 // визначаємо контакт D7 як CLK_DIO (управління бітами)
#define DATA_DIO 8 // визначаємо контакт D8 як DATA_DIO (послідовність даних)

#define Pot1 0 // визначаємо аналоговий вхід як Pot1

/* Сегментні байтні карти для чисел від 0 до 9 */
const byte SEGMENT_MAP[] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0X80,0X90};
/* Байтні карти для чисел від 1 to 4 */
const byte SEGMENT_SELECT[] = {0xF1,0xF2,0xF4,0xF8};

void setup ()
{
  Serial.begin(9600);
  /* Встановлюємо статуси роботи вихідних контактів */
  pinMode(LATCH_DIO,OUTPUT);
  pinMode(CLK_DIO,OUTPUT);
  pinMode(DATA_DIO,OUTPUT);
}

/* Головна програма*/
void loop()
{
  int PotValue;
  PotValue = analogRead(Pot1);

  Serial.println(PotValue);
  /* Оновляємо дисплей поточним значенням лічильника */
  WriteNumberToSegment(0 , PotValue / 1000);
  WriteNumberToSegment(1 , (PotValue / 100) % 10);
}
```

```

WriteNumberToSegment(2 , (PotValue / 10) % 10);
WriteNumberToSegment(3 , PotValue % 10);
}

/* Введіть десяткове число від 0 до 9 на одну з 4 цифр дисплея */
void WriteNumberToSegment(byte Segment, byte Value)
{
    digitalWrite(LATCH_DIO,LOW);
    shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_MAP[Value]);
    shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_SELECT[Segment] );
    digitalWrite(LATCH_DIO,HIGH);
}

```

У цьому скетчі є дві новації. По-перше, це використання масивів, зокрема SEGMENT_MAP[] та SEGMENT_SELECT[], по-друге — використання функції WriteNumberToSegment(), яка не є стандартною функцією мови Arduino.

Теорія.

Масив - іменований набір однотипних змінних із доступом до окремих елементів по їх індексу.

Функція. Взагалі функції у програмуванні є дуже ефективним засобом організації скетчів, вони кодують одну дію тільки в одному місці програми, а потім використовуються у багатьох місцях. Тому функції скорочують текст скетчу і роблять його компактним. Крім того, функції скорочують шанси на появу помилки при необхідності зміни коду.

У цьому скетчі використовуються функції, зокрема WriteNumberToSegment() та shiftOut() з відповідними аргументами. Раніше використовувалися стандартні функції Arduino, такі як pinMode(), digitalWrite(), delay() та інші. У даному випадку функція shiftOut() є також стандартною функцією мови Arduino, вона виводить байт інформації на порт вхід/виходу послідовно (побітно). Кожен біт послідовно подається на заданий порт, після чого подається сигнал на синхронізуючий порт вхід/вихід, інформуючи про доступність до зчитування наступного біта. Біти можуть зчитуватися як зліва так і справа, тому треба вказати напрям за допомогою константи MSBFIRST(Most Significant Bit First).

Функція WriteNumberToSegment() не належить до вбудованих (стандартних) функцій Arduino, вона є створеною. Її аргументами є змінні Segment та Value, які мають тип byte, перший з яких вказує на розряд індикатора, другий - формує відповідний сегмент цифри. Тобто після вводу будь-якого десяткового числа від 0 до 9 воно перетворюється на двійкове число. У регістрі зсуву кожний біт цього числа надходить послідовно на ніжку 14 лівого регістру, а потім спрямовується паралельно окремо на кожну з чотирьох шин входу, визначаючи відповідний розряд індикатора. Після цього формуються таким же чином (на правому регістрі) відповідні сегменти кожної цифри. В якості самостійної роботи можна замінити ввод числа з клавіатури потенціометром з якого зчитується значення напруги.

ж) Вимір температури (цифровий датчик DS18B20)

Температурні датчики одні з найважливіших складових вимірювальних систем управління. Датчики температури необхідні для контролю безлічі життєво важливих і критичних процесів. Вони застосовуються практично в будь-яких сферах або виробництвах, де температура об'єкта впливає на якість роботи і підсумкової продукції, вимагає пильного температурного контролю. Особливої уваги потребує використання температурних датчиків у навчальній діяльності на уроках природничих дисциплін та на інтегрованих STEM-уроках.

Датчики температури засновані на різних фізичних ефектах. Зокрема, цифровий датчик DS18B20 відноситься до напівпровідникових датчиків, фізичний принцип роботи яких заснований на залежності від температури електричного струму на р-п переході діода або транзистора. Дана залежність близька до лінійної, що дозволяє створювати датчики, які не потребують складних схем корекції. Вихідним сигналом є падіння напруги на датчику, саме це падіння перетворюється на цифровий вигляд за допомогою аналого-цифрового перетворювача (АЦП) (Рис.3.11).

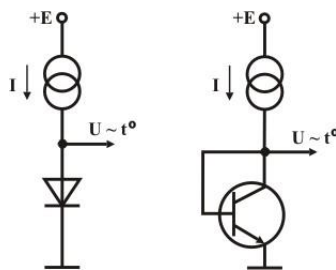


Рис. 3.11. Фізичний принцип роботи напівпровідникового датчика

Цифровий датчик DS18B20 представлений у різних корпусах (Рис. 3.12):

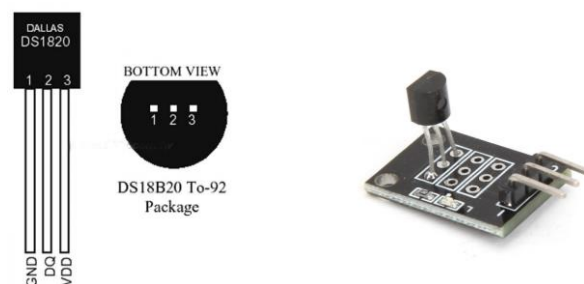


Рис. 3.12. Корпуси датчика DS18B20

Особливості цифрового датчика DS18B20:

Температурний діапазон вимірювань лежить в межах від -55 С до +125 С.

Похибка вимірювання не більше 0,5 С (для температур від -10С до + 85С).

Датчик харчується напругою від 3,3 до 5В.

Можна програмно задати максимальну роздільну здатність до 0,0625C, найбільше дозвіл 12 біт.

Інформація передається по протоколу 1-Wire.

Для приєднання до мікроконтролеру потрібні тільки 3 дроти.

Цифрові датчики передають значення вимірюваної температури в вигляді певного двійкового коду, який надходить на цифровий або аналоговий вхід Ардуіно і потім декодується по протоколу даних 1-Wire. Протокол 1-Wire вимагає окремого розгляду, вкажемо лише необхідний мінімум для розуміння принципів взаємодії. Обмін інформацією в 1-Wire відбувається завдяки таким операціям:

- ініціалізація - визначення послідовності сигналів, з яких починається вимір і інші операції. Провідний пристрій подає імпульс скидання, після цього датчик повинен подати імпульс присутності, повідомляє про готовність до виконання операції.
- запис даних - відбувається передача байта даних в датчик.
- читання даних - відбувається прийом байта з датчика.

Схема підключення цифрового датчика DS18B20 надається на рис. 3.13.

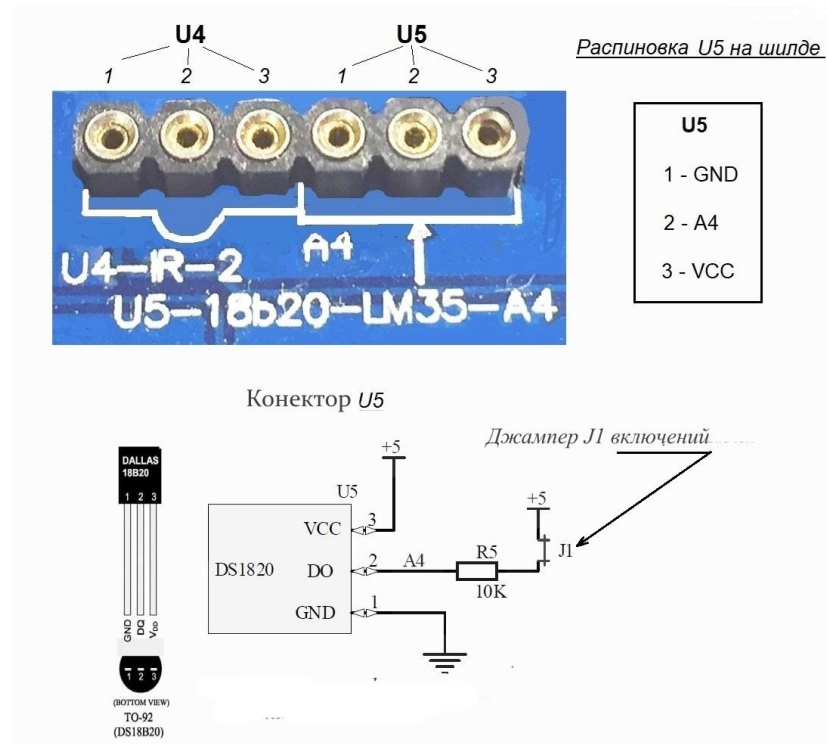


Рис.3.13. Схема підключення цифрового датчику DS18b20

Як зазначено у схемі маркована сторона датчика температури повинна бути спрямована на нижню сторону плати, що стоїть перед трьома кнопками. Також у назві "U5-18b20-LM35-A4" є стрілка, спрямована до середнього штифта (A4) заголовка датчика температури (GND зліва, DQ в середині та 5 V праворуч). Перемичка (джампер) J1 подає 5 В на середній штифт DQ через 10K резистор.

Послідовність дій (алгоритм) у процесі вимірювання температури за допомогою датчика DS18B20 відбувається таким чином.

- визначення адреси датчика, перевірка його підключення;
- на датчик подається команда з вимогою прочитати температуру і викласти виміряне значення в регістр.
- подається команда на зчитування даних з регістра і відправка отриманого значення в «монітор послідовного порту»,
- якщо потрібно, то проводиться конвертація в градуси Цельсія / Фаренгейт.

Нижче представлений відповідний скетч виміру температури:

```
//підключаємо необхідні бібліотеки
#include <TimerOne.h>
#include <MultiFuncShield.h>
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire oneWire(18); // Створюємо об'єкт OneWire ждя датчика DS18b20,
```

```

/* Створюємо об'єкт DallasTemperature для роботи з сенсорами, передаючи йому //
посилання на об'єкт для роботи з 1-Wire*/
DallasTemperature ds(&oneWire);
/* Створюємо об'єкт DallasTemperature для роботи з сенсорами, передаючи йому //
посилання на об'єкт для роботи з 1-Wire*/
// NOTE: Для коректної роботи датчика треба включити джампер J1
void setup() {
  Timer1.initialize(); //ініціалізація таймеру
  MFS.initialize(&Timer1); // ініціалізація бібліотеки БФП
  Serial.begin(9600);
  ds.begin(); // ініціалізація датчика
}

void loop() {
  ds.requestTemperatures(); // зчитуємо температуру з датчика
  MFS.write(ds.getTempCByIndex(0) , 1); // відправляємо значення температури на дисплей з
//1-м знаком після коми.
  Serial.println(ds.getTempCByIndex(0));
  delay(100);
}

```

е) Вимір температури (аналоговий датчик LM35)

Аналоговий датчик LM35 (Рис. 3.14) також відноситься до напівпровідникових датчиків, тобто вимірювання температури відбувається за допомогою визначення падіння напруги на р-п переході. Але для усунення всіх негативних явищ, пов'язаних з роботою такого переходу, використовується спеціальна схема, яка містить в своєму складі два чутливих елемента (транзистора) з різними характеристиками. Вихідний сигнал формується як різниця падінь напруги на кожному чутливому елементі. При відніманні значно скорочуються негативні моменти. Подальше підвищення точності вимірювання здійснюється калібруванням датчика за допомогою зовнішніх ланцюгів.

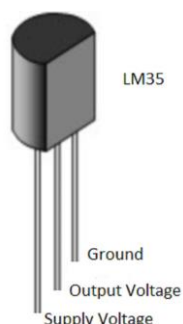


Рис.3.14. Аналоговий датчик LM35

Цей датчик підключається в той же роз'єм, що і попередний цифровий датчик DS18B20, але він має зворотну цоколювку. Але для коректної роботи треба прибрати джампер J1 (Рис. 3.15).

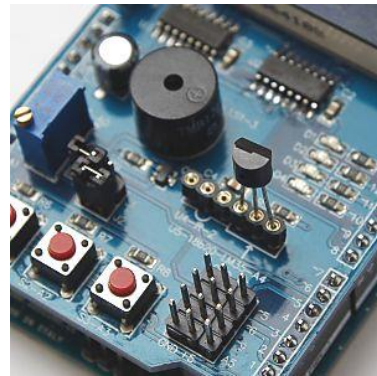
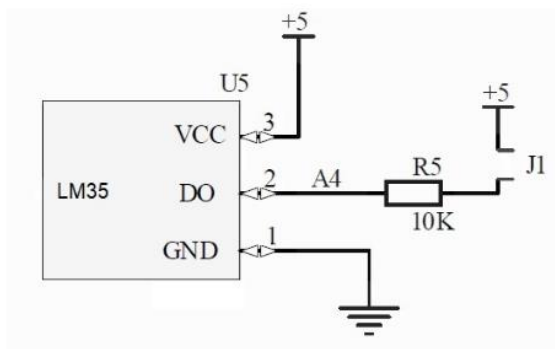


Рис. 3.15. Схема підключення аналогового датчика LM35

Нижче представлений відповідний скетч виміру температури:

```
#include <TimerOne.h>
#include <Wire.h>
#include <MultiFuncShield.h>
// NOTE: make sure jumper J1 is removed from shield, and that LM35 is inserted
correctly.
void setup() {
  // put your setup code here, to run once:
  Timer1.initialize();
  MFS.initialize(&Timer1); // ініціалізуємо бібліотеку БПФ

  // Ініціалізуємо за допомогою фільтра низьких частот.
  // Оберіть: SMOOTHING_NONE, SMOOTHING_MODERATE or SMOOTHING_STRONG
  MFS.initLM35(SMOOTHING_MODERATE);
}
void loop() {
  // put your main code here, to run repeatedly:
  int tempCentigrade = MFS.getLM35Data(); // отримати градус в 1/10 ступені.

  MFS.write((float)tempCentigrade / 10, 1); // відобразити температуру з одним знаком після
коми.

  delay(100);
}
```

ж) Вимір відстані за допомогою ультразвукового далекоміру HC SR04

Найбільш поширеним датчиком виміру відстані вважається ультразвуковий сонар. Ультразвукові датчики відстані дуже затребувані в робототехнічних проектах через свою відносну простоту, достатню точність та доступність. Вони можуть бути використані як прилади, які допомагають об'їжджати перешкоди, отримувати розміри предметів, моделювати карту приміщення і сигналізувати про наближення або видалення об'єктів. Одним з поширених

варіантів таких пристроїв є датчик відстані, в конструкцію якого входить ультразвуковий далекомір HC SR04.

Ультразвукові коливання - це механічні коливання, які відбуваються з частотою вище 20000 герц, а значить, більше верхньої межі коливань звуку, яка сприймається людиною. Основним призначенням ультразвукового датчика є вимір відстані до контрольованого об'єкта або реєстрація появи об'єкта в зоні "поля зору" датчика. В основу принципу дії будь-якого ультразвукового датчика закладено явище відображення акустичних хвиль, що поширюються в повітрі. Ультразвукові датчики використовують ультразвукові хвилі як інформаційні носії. Трансмітер (випромінювач) та ресивер (приймач) імпульсів звуку знаходяться в одному корпусі. Передатчик посилає імпульси звуку, вони відбиваються від об'єкта, до якого вимірюються відстань, сприймаються приймачем і перетворюються в напругу. Відстань до об'єкту визначається за формулою, в який головним фактором є вимірний час до приходу відбитого сигналу.

Ультразвукові способи вимірювання є електричними, тому що збудження коливань і їх прийом здійснюється за допомогою електрики. Найчастіше в датчиках застосовують п'єзоелементи, перетворювачі магнітострикційного виду. Для збудження коливань ультразвукової частоти застосовується ефект розтягування і стиснення п'єзокристалу, відомий як зворотний п'єзоефект. Тому п'єзоелемент застосовується як в якості приймача коливань, так і в якості випромінювача.

Ультразвуковий датчик відстані HC SR04 є приладом безконтактного типу, і забезпечує високоточне вимірювання і стабільність (Рис.3.16.). Діапазон дальності його вимірювання складає від 2 до 400 см. Випромінювач модулю HC SR04 посилає 40 імпульсів частотою 40 КГц. З метою підвищення точності виміру відстані треба правильно направити датчик: зробити так, щоб предмет був в рамках конуса діаграми спрямованості.



Рис. 3.16. Ультразвуковий датчик HC SR0-4

Основні характеристики ультразвукового датчика є такі:

Робоча напруга: 3.8 - 5.5 В;

Струм: 8 мА;

Частота: 40 КГц;

Максимальна дистанція вимірювання: 4 м;

Кут: 15°;

Розміри: 37x20x15 мм;

При використанні ультразвукового модуля HC SR04 треба використовувати бібліотеку БПФ, в основному таких, що використовувалися у попередніх проектах. У даній бібліотеці використовуються три рівня фільтрації для згладжування показань від модуля модуля HC SR04.

Відповідний скетч визначення відстані за допомогою HC SR04 представлений нижче.

```
//підключаємо необхідні бібліотеки
#include <TimerOne.h>
#include <Wire.h>
#include <MultiFuncShield.h>

const int TrigPin = 5; // Трансмітер підключений до D5
const int EchoPin = 6; // Ресивер підключений к D6

void setup() {
  pinMode(TrigPin, OUTPUT); // Трансмітер - вихідний режим
  pinMode(EchoPin, INPUT); // Ресивер — вхідний режим

  Timer1.initialize(); //ініціалізація таймеру
  MFS.initialize(&Timer1); // ініціалізація бібліотек БПФ

  // Ініціалізація із використанням фільтра нижніх частот: SMOOTHING_NONE,
  SMOOTHING_MODERATE or SMOOTHING_STRONG
  MFS.initSonar(SMOOTHING_MODERATE);
}

void loop() {
  MFS.write((int)MFS.getSonarDataCm(TrigPin, EchoPin)); /*виводимо дані безпосередньо на
  дисплей */
  delay(100); //пауза 100 мс
}
```

з) Керування кутом повороту сервомотора за допомогою потенціометра.

Сервопривод - це мотор, положенням вала якого ми можемо керувати. Від звичайного мотора він відрізняється тим, що йому можна точно в градусах задати положення, в яке постане вал. Сервоприводи використовуються для моделювання різних механічних рухів роботів (Рис. 3.17).



Рис. 3.17 Сервомотор MG995

Технічні характеристики MG995:

Маса: 55 грам;

Розміри: приблизно 40.7 x 19.7 x 42.9;

Крутний момент: 8.5 кг x см (при 4.8 В живлення), 10 кг x см (при 6 В);

Швидкість: 0.2 с / 60° (при 4.8 В), 0.16 с / 60° (при 6 В);

Робоче живлення: 4.8 - 7.2 В;

Ширина мертвої зони: 5 мкс;

Діапазон робочих температур: 0 °C - 55 °C.

Увага: Вихідний вал сервоприводу MG995 повертається приблизно на 120 градусів (60 градусів в кожному напрямку).

Підключення сервомотору відбувається за схемою, де використовуються два зсувних регістра, які сприймають дані з потенціометру (Рис. 3.18).

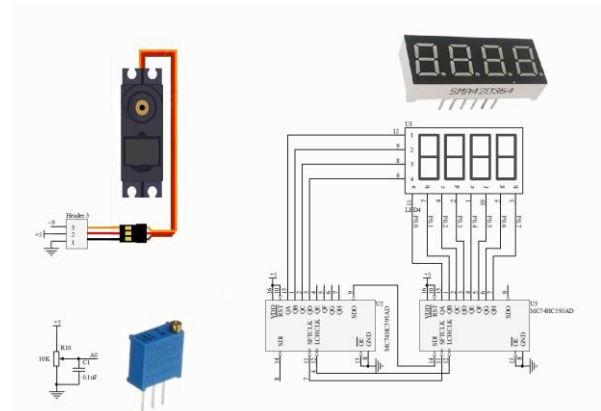
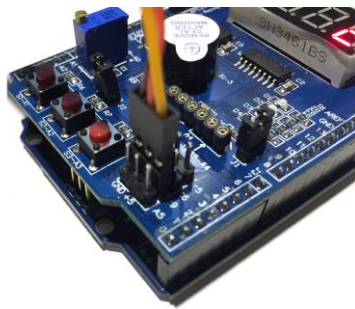


Рис.3.18. Підключення сервомотору та потенціометру

У даному проєкті для керування валом сервомотору використовується потенціометр, який підключається до аналогового контакту A0. Дані з потенціометру зчитуються та перетворюються та відображаються на індикаторі. У скетчі використовуються бібліотеки, зокроєма, бібліотека, SoftwareServo.h, яка спрощує написання програми.

Нжче представлений скетч керування кутом поворомут валу сервомотору.

```
// підключаємо необхідні бібліотеки для роботи БПФ
#include <TimerOne.h>
#include <Wire.h>
#include <MultiFuncShield.h>
#include <SoftwareServo.h> // підключаємо бібліотеку сервомотора
```

```
SoftwareServo myservo; // створюємо серво-об'єкт myservo для керування сервомотором
int potpin = 0; // Надаємо ім'я potpin аналоговому виводу A0, до якого підключена середня
ніжка потенціометра
```

```

int val; // змінна зчитування даних з аналогового входу
int ugot; // змінна для зберігання кута повороту сервомотора

void setup()
{
  Timer1.initialize();
  MFS.initialize(&Timer1); // инициализация multi-function shield library
  myservo.attach(9); // Подключаем цифровой вывод (pin 9) к сервомотору
  Serial.begin(9600);
}

void loop()
{
  val = analogRead (potpin); // зчитує значення потенціометра (значення від 0 до 1023)
  ugot = map (val, 0, 1023, 0, 180); /* перетворимо показання потенціометру в значення від 0
до 180 */
  myservo.write (ugot); /* встановлюємо положення сервоприводу відповідно до перетвореним
значенням в кут повороту */
  Serial.println (myservo.read ()); /* відправляємо в порт монітора значення кута повороту з
сервомотора */
  MFS.write (myservo.read ()); /* відправляємо значення кута повороту сервомотору на
дисплей */
  delay (15); // чекаємо, коли сервомотор відпрацює команду

  SoftwareServo :: refresh (); // команда поновлення сервоприводу, кожні 50 мс.
}

```

