І.Ш. Невлюдов, В.А. Андрусевич,
С.П. Новоселов, О.Г. Резніченко

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ В УПРАВЛІННІ ПРИСТРОЯМИ НА МІКРОКОНТРОЛЕРАХ

Харків - 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

І.Ш. Невлюдов, В.А. Андрусевич,
С.П. Новоселов, О.Г. Резніченко

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ В УПРАВЛІННІ ПРИСТРОЯМИ НА МІКРОКОНТРОЛЕРАХ

Навчальний посібник

Електронне видання

Харків 2023

УДК 621.311
Н40

Рекомендовано Вченою радою
Харківського національного університету радіоелектроніки
(Протокол №3/1 від 29.07.2022 р.)

Невлюдов І.Ш.
Технології Інтернету речей в управлінні пристроями на мікроконтролерах: Навчальний посібник [Електронний ресурс] / І.Ш. Невлюдов, В.А. Андрусевич, С.П. Новоселов, О.Г. Резніченко. – Електронне видання. – Харків: ХНУРЕ, 2023. – 214 с. – pdf 2,88 Mb.

ISBN 978-966-659-364-4

У навчальному посібнику подані відомості про сучасну концепцію Інтернету речей, що дозволяє здійснювати передачу та обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку. На реальних прикладах розглянуто процес підключення, моніторингу датчиків, віддаленого управління інтелектуальними виконавчими пристроями через Інтернет. Надано практичні рекомендації використання мікроконтролерів для організації взаємодії між периферійним пристроями та мережею Internet. Розглянуті задачі, що потребують застосування хмарних обчислень для віддаленого збору даних та моніторингу стану різноманітних сенсорів, їх оброблення та відображення кінцевому користувачеві у вигляді таблиць і трендів.

Розглянуті популярні апаратні платформи Інтернету речей та дана їх стисла характеристика. Наведені особливості розробки програм для Arduino на C++, надані загальні відомості про основні оператори, структуру програм, принципи застосування бібліотек і функцій. Розглянуті приклади створення вбудованої WEB-сторінки, застосування WiFi-модулів для віддаленого доступу до об’єкта керування за допомогою мережі Internet. Надані відомості про технологію взаємодії клієнт-сервер, розглянуті приклади застосування поширених соціальних мереж, зокрема Twitter, для управління розумним будинком.

Навчальний посібник призначено для здобувачів вищої освіти денної та заочної форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології». Може бути корисний аспірантам і фахівцям у промисловості, робота яких пов’язана з розробкою та організацією виробництв галузі автоматизації та приладобудування.

УДК 621.311

ISBN 978-966-659-364-4
DOI: 10.30837/978-966-659-364-4

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API	–	Application Programming Interface;
DFM	–	Design for Manufacturing;
DMP	–	Digital Motion Processor;
EEPROM	–	Electrically Erasable Programmable Read-Only Memory;
IDE	–	Integrated Development Environment;
I2C	–	Inter-Integrated Circuit;
IMU	–	Inertial Measurement Unit;
IoT	–	Internet of Things;
FCC	–	Federal Communications Commission;
HTTP	–	Hypertext Transfer Protocol;
LTE	–	Long Term Evolution;
MQTT	–	Message Queuing Telemetry Transport;
RFID	–	Radio-Frequency IDentification;
PRNG	–	Pseudorandom number generator;
SSID	–	Service Set Identifier;
SoC	–	System-on-a-Chip;
АРП	–	автоматичне регулювання підсилення;
АЦП	–	аналого-цифровий перетворювач;
ГПВЧ	–	генератор псевдовипадкових чисел;
ІЧ	–	інфрачервоний;
ОЗП	–	оперативний запам'ятовувальний пристрій;
ООП	–	об'єктно-орієнтоване програмування;
ПЗ	–	програмне забезпечення;
ПЗП	–	постійний запам'ятовувальний пристрій;
ПК	–	персональний комп’ютер;
ЦАП	–	цифро-аналоговий перетворювач;
ЦП	–	центральний процесор;
ЧПУ	–	числове програмне управління;
ШІМ	–	широтно-імпульсна модуляція.

ВСТУП

Галузь Інтернету речей – одна з головних світових тенденцій. Сучасна концепція Інтернету речей передбачає спілкування об’єктів, які використовують технології для комунікації між собою та з навколишнім середовищем. Така концепція дозволяє пристроям виконувати певні завдання, виключаючи людське втручання. Тому всі пристрої в різних інфраструктурних системах мають обробляти, аналізувати та обмінюватися інформацією, приймати рішення та вживати певні заходи залежно від результатів.

Інтернет речей – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу та обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку. Окрім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через проводові чи безпроводові мережі. Ці взаємопов'язані пристрої мають можливість зчитування і приведення в дію функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини за рахунок використання інтелектуальних інтерфейсів.

У даному навчальному посібнику покроково розглянуто процес підключення, моніторингу датчиків, віддаленого управління інтелектуальними виконавчими пристроями через Інтернет. Надано практичні приклади використання мікроконтролерів для організації взаємодії між периферійним пристроями та мережею Internet. Розглянуті задачі, що потребують застосування хмарних обчислень для віддаленого збору даних і моніторингу стану різноманітних сенсорів, їх оброблення та відображення кінцевому користувачеві у вигляді таблиць і трендів.

Як платформи для дослідження принципів застосування Інтернету речей у посібнику використовується популярна платформа Arduino та набори розповсюджених модулів.

Навчальний посібник складається з трьох розділів. Матеріал подано в зручній та доступній для вивчення студентами формі з великою кількістю ілюстрацій і практичними прикладами.

Перший розділ навчального посібника присвячено висвітленню основ концепції Інтернету речей, розглянуті популярні апаратні платформи Інтернету речей та надана їх стисла характеристика. Розглянуті особливості розробки

програм для Arduino на C++, надані загальні відомості про основні оператори, структуру програм, принципи застосування бібліотек і функцій.

У другому розділі розглядаються принципи керування пристроями IoE за допомогою Arduino. Наведено багато прикладів керування периферійними модулями, починаючи з двигунів постійного струму та закінчуючи акселерометрами та Bluetooth-модулями.

У третьому розділі наводяться відомості щодо елементів керування інтелектуальним будинком через мережу Ethernet. Розглянуті приклади створення вбудованої WEB-сторінки, застосування WiFi-модулів для віддаленого доступу до об’єкта керування за допомогою мережі Internet. Надані відомості про технологію взаємодії клієнт-сервер, розглянуті приклади застосування поширених соціальних мереж, зокрема Twitter, для управління розумним будинком.

Зміст навчального посібника «Технології Інтернету речей в управлінні пристроями на мікроконтролерах» відповідає програмі підготовки бакалаврів, що навчаються за спеціальністю 151 «Автоматизація та комп’ютерно-інтегровані технології».

1 ОСНОВИ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ

1.1 Концепція Інтернету речей

Інтернет Речей або Internet of Things (IoT) – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку [3, 4]. Окрім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через проводові чи безпроводові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини за рахунок використання інтелектуальних інтерфейсів. Цей термін включає в себе широкий спектр застосування, від споживчих пристроїв і до індустріальних активів, таких як машини, роботи, нафтогазові комплекси або навіть працівники.

Основною концепцією IoT є можливість підключення всіляких об'єктів (речей), які людина може використовувати в повсякденному житті, наприклад, холодильник, кондиціонер, автомобіль, велосипед і навіть кросівки. Всі ці об'єкти (речі) мають бути оснащені вбудованими давачами або сенсорами, які мають можливість обробляти інформацію, що надходить з навколишнього середовища, обмінюватися нею і виконувати різні дії залежно від отриманої інформації. Прикладом впровадження такої концепції є система «розумний будинок». Ця система аналізує дані навколишнього середовища і залежно від показників регулює температуру в приміщенні. У зимовий період регулюються інтенсивність опалення, а в разі спекотної погоди будинок має механізми відкривання й закривання вікон, завдяки чому провітрюється будинок, і все це відбувається без втручання людини [4].

Для застосування розумної системи потрібне обладнання, яке спочатку розпізнає одне одного, потім зможе отримувати необхідну інформацію та буде на зв'язку з комп'ютером, який опрацює усі дані. Інтернет речей може по-різному підключатися та обмінюватися даними (Wi-Fi, Bluetooth або навіть LTE). Систему об'єднує програмне забезпечення, яке створено спеціально для збору, обробки та управління даними.

Інтернет речей найчастіше працює на основі LoRaWan (далекі відстані) протоколу [5]. За допомогою цього протоколу IoT можна впроваджувати як у приміщенні, так і на відкритому повітрі. Базові станції завжди підключені до потрібного діапазону частоти, де приймають сигнали і запити від інших пристроїв. Потім сервер управляє усіма базовими станціями (шлюзами). На останньому етапі підключається сервер програм, який займається обробкою та передачею інформації абоненту.

Не існує точних рамок або списку приладів, де можна застосувати систему Інтернету. На практиці IoT можна впровадити навіть у приватному будинку, оскільки практично будь-який фізичний об'єкт можна перетворити на «розумний». Але якщо розібратися докладніше, то Інтернет речей можна запровадити в:

– розумний будинок – з розумною системою кондиціонування або обігріву тощо;

– промисловість – програмні системи, сенсори, аналіз даних, розумні машини та обладнання;

– охорона здоров'я – медичні дрони, індивідуальний підхід до пацієнтів, аналіз роботи лікаря;

– агросектор – прогноз кліматичних змін, обладнання для перевірки складу ґрунту, відстеження стану здоров'я тварин і навіть де знаходяться хворі тварини;

– рітейл – найпопулярніше це безконтактна оплата і спеціальні додатки для покупок і доставки онлайн.

На даний момент йде своєрідне змагання між апаратними платформами для інтернету речей, розроблених різними виробниками. Кожен з цих виробників розуміє, що чим успішніша й ефективніша буде його платформа, тим більший прибуток він зможе отримати (рис. 1.1).

При цьому слід зазначити, що частина з цих платформ орієнтована тільки на один аспект тематики інтернету речей (наприклад, SigFox фокусується на підключенні пристроїв), а інші є своєрідними платформами «все в одному» (наприклад, Particle.io), надаючи комплексне рішення для розробки інтернет-речей. Звичайно, наш список подібних платформ далеко не повний – їх зараз дуже багато.

Сучасна концепція Інтернету речей передбачає спілкування об’єктів, які використовують технології для комунікації між собою та з навколишнім середовищем. Така концепція дозволяє пристроям виконувати певні завдання, виключаючи людське втручання. Тому всі пристрої в різних інфраструктурних

системах мають обробляти, аналізувати та обмінюватися інформацією, приймати рішення та вживати певні заходи залежно від результатів.

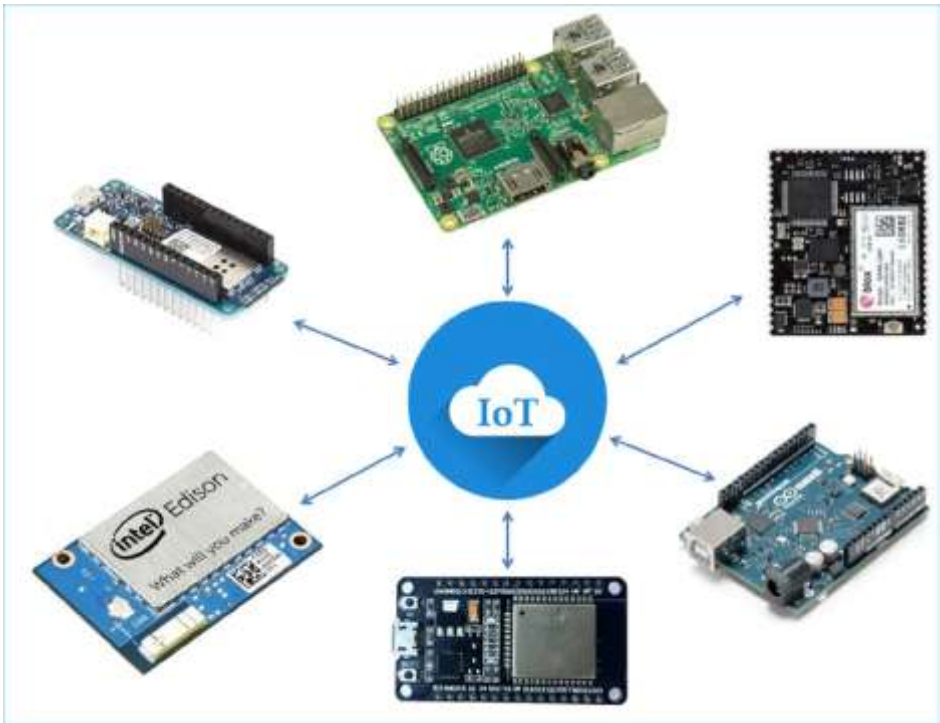


Рисунок 1.1 – Апаратні платформи Інтернету речей

Сьогоднішня система «розумного дому», мабуть, найбільш тісно пов’язана з Інтернетом речей. Концепція включає використання побутових пристроїв, які вже перейняли цю технологію, таких як термостати, системи відеоспостереження, холодильники, телевізори та багато іншого. Цей сегмент технології базується на використанні ситуаційних, децентралізованих радіомереж. Багато з цих систем вже можна побачити у приватних домогосподарствах та офісах, доступні нові й нові послуги – дистанційний моніторинг через смартфон у власних приміщеннях або автоматичні кліматичні системи управління в будинках.

Основними функціями таких систем є безпека будинку та успішне використання енергоресурсів. Полегшення повсякденного життя – важлива функція цієї концепції.

IoT являє великий інтерес для обробки інформації про об'єкти, що рухаються, особливо для автотранспорту. Ці технології дозволяють діагностувати експлуатацію транспортних засобів, уникати аварій, замовити необхідні запчастини і дати рекомендації щодо пошуку необхідної станції та встановлення терміну обслуговування транспортного засобу [4].

Для об'єднання повсякденних речей у мережу потрібні декілька технологій.

Для ідентифікації кожного об'єкта потрібна проста, компактна технологія. Тільки за наявності системи унікальної ідентифікації можна збирати та накопичувати інформацію про певний предмет. Такий функціонал можна забезпечити за допомогою мікросхем RFID (Radio-Frequency IDentification). Вони здатні без власного джерела струму передавати інформацію приладам зчитування. Кожна мікросхема має індивідуальний номер. Як альтернатива до даної технології для ідентифікації об'єктів можуть використовуватися QR-коди. Для визначення точного місця знаходження речі підійде технологія GPS, яка ефективно використовується вже сьогодні у смартфонах та навігаторах.

Для відслідковування змін у стані елементу чи оточуючого середовища об'єкти мають оснащуватися сенсорами. Для обробки та накопичення даних з сенсорів має використовуватися вбудований комп'ютер (наприклад Raspberry Pi, Intel Edison).

Обмін інформацією між пристроями може бути організовано з використанням технології безпроводових мереж (Wi-Fi, Bluetooth, ZigBee, 6LoWPAN).

Для передачі даних використовуються оптимізовані та легковагі протоколи типу MQTT. Вони грунтуються на принципах публікації і підписок, де кожен пристрій (давач або сенсор) взаємодіє з програмою на сервері (брокером).

Інтернет речей тісно перетинається з концепцією Індустрії 4.0 [2]. Комунікація на виробництві в рамках Індустрії 4.0 займає передове місце. В інтелектуальному виробництві обладнання, контролери і датчики взаємодіють як один з одним, так і безпосередньо з Ethernet або у хмарі. Закрита система стає відкритою. Але змінюється не тільки кількість інформації, що обробляється безпосередньо на місці. Підвищується і якість до абсолютно нового рівня.

Інформація про стан виробничого обладнання та пов'язане з ним прогнозування про можливі простої виробництва за допомогою інноваційних систем зворотного зв'язку подає тільки один із прикладів. Це стало можливим завдяки швидкому збільшенню обчислювальної потужності, яку також можна вже використовувати децентралізовано в периферії, на краю мережі або в основах виробництва [2].

Результатом цього стає більш гнучке й динамічне виробництво, яке в будь-який час може індивідуально і швидко реагувати на вимоги клієнтів. Децентралізована обчислювальна потужність в ідеології Індустрії 4.0 переробляє дані в інформацію безпосередньо в датчику. Рішення приймаються

децентралізовано. Релевантна для технологічного процесу, виробництва і підприємства інформація направляється безпосередньо в Ethernet і хмарний сервіс.

1.2 Апаратні платформи для проєктів інтернету речей (IoT)

1.2.1 Платформа Particle.io

Particle.io – це одна з найповніших комплексних платформ інтернету речей [6]. Це платформа «все в одному», яка пропонує рішення для розробки обладнання Інтернету речей, можливості підключення, хмари пристроїв та застосунків. Particle.io виробляє широку лінійку продуктів для розробки обладнання IoT як для швидких прототипів, так і для виробництва на рівні DFM. Створення продукту інтернету речей у ній починається з підключення пристроїв до Інтернету, і всі плати мікроконтролерів Particle підтримують зв'язок Wi-Fi, стільниковий зв'язок (2G/3G/LTE) або mesh-мережі. Деякі з цих плат підтримують одразу кілька типів зв'язку (рис. 1.2).

Мікроконтролери платформи Particle.io управляються спеціальною операційною системою, яка дозволяє розробнику легко інтегрувати пристрої з хмарою пристроїв та програмами particle. Як правило, їх пристрої та комунікаційні модулі поставляються із сертифікатами CE та FCC, які знижують вартість сертифікації, коли продукт готовий до масового виробництва. Їхні плати з відкритим вихідним кодом забезпечуються добрим рівнем технічної підтримки, що, звичайно ж, дуже зручно для розробки продуктів на основі цієї платформи.

Рисунок 1.2 – Плата розробника SARA-U201

1.2.2 Плати Espressif ESP8266 и ESP32 [7]

Коли справа доходить до створення пристроїв інтернету речей, асортименти продуктів Espressif та AI thinker – це, мабуть, найкраща річ для платформи Particle.io. З моменту випуску чіпа Wi-Fi ESP8266-01 кілька років тому чіпи та плати на базі ESP8266 перетворилися на один з найбільш переважних чіпсетів для пристроїв інтернету речей на основі Wi-Fi. Ці модулі досить дешево коштують, споживають мало електроенергії та прості у використанні. Це, крім інших чинників, привертає до них особливу увагу розробників апаратного забезпечення. Слідом за модулем ESP8266 компанія Espressif кілька років тому представила модуль ESP32, який побудований вже на основі двоядерного мікроконтролера і має ширший набір функціональних можливостей порівняно з модулем ESP8266, а коштує не набагато дорожче.

Приклад плати розробника на основі чіпу ESP32 показано на рис. 1.3 [8].

Рисунок 1.3 – Приклад плати розробника на основі чіпу ESP32

Мікросхеми ESP мають велику гнучкість і можуть використовуватися як модулі Wi-Fi, підключатися до інших мікроконтролерів або використовуватися в автономних режимах без використання додаткових мікроконтролерів.

Модулі мають невеликі форм-фактори і спрощують реалізацію функцій, необхідних для сфери інтернету речей, таких, наприклад, як оновлення прошивки «повітрем» (технологія OTA – over-the-air). Наявність плат розробки, таких як NodeMCU та інших сторонніх плат на базі ESP дозволяє розробникам детально ознайомитися з можливостями даної платформи, перш ніж використовувати її в проєктах. Також, як й інші плати на основі платформи Particle.io, плати ESP8266 поставляються із сертифікацією FCC та CE для зниження загальних витрат на сертифікацію пристрою після виготовлення. Плати ESP надають один з найнадійніших, виділених інтерфейсів Wi-Fi, що включає кілька протоколів, які підтримують технологію інтернету речей,

таких як, наприклад, протокол ESP Touch, який дозволяє пристрою безпечно та безперешкодно виходити в Інтернет через мережі Wi-Fi.

1.2.3 Плати розробки Intel IoT

Компанія Intel, без сумніву, є одним із найбільших гравців на ринку напівпровідникових елементів, тому не дивно, що серед їхніх продуктів є плати з функціями, що дозволяють використовувати інтернет речей. Хоча вони припинили підтримку деяких старих плат, певні з цих плат все ще використовуються для швидкого прототипування розробниками проєктів Інтернету речей. Однією з головних особливостей подібних плат компанії Intel є їх величезні обчислювальні можливості. Однією з найпопулярніших плат Intel у тематиці інтернету речей є обчислювальний модуль Intel Edison (рис. 1.4).

Цей обчислювальний модуль був розроблений для експертів, виробників, підприємців та для використання у промислових додатках інтернету речей. Модуль забезпечує простоту у розробці прототипів та використання у ряді комерційних підприємств. Його використання виправдано у випадках, коли критично важлива продуктивність системи.

Рисунок 1.4 – Обчислювальний модуль Intel Edison

Модуль Intel Edison [9] використовує 22-нм модуль Intel SoC (system-on-a-chip – система на кристалі), який включає двоядерний двопотоковий процесор Intel Atom з частотою 500 МГц і 32-розрядний мікроконтролер Intel® Quark, що працює на частоті 100 МГц. Однак цей модуль та більшість інших плат, таких як Intel Curie та Intel Galileo, були зняті з виробництва. Зараз найбільш популярною платформою для розробки апаратного забезпечення інтернету речей від компанії Intel є комплект для розробки інтернету речей Up Squared groove, який є платформою, розробленою спеціально для задоволення жорстких вимог промислових застосунків інтернету речей (рис. 1.5).

Рисунок 1.5 – Рішення від Intel на основі процесора Celeron

1.2.4 Плати розробки від компанії Adafruit

Adafruit – це один із найбільших інтернет-магазинів електронних компонентів [10]. Ця компанія також займається розробкою бібліотек для роботи з компонентами, які вона продає (дисплеї, адресні світлодіодні стрічки, датчики та багато іншого). Ті, хто стикався з платформою Arduino, знають, яку важливу роль відіграє компанія Adafruit в еко-системі Arduino.

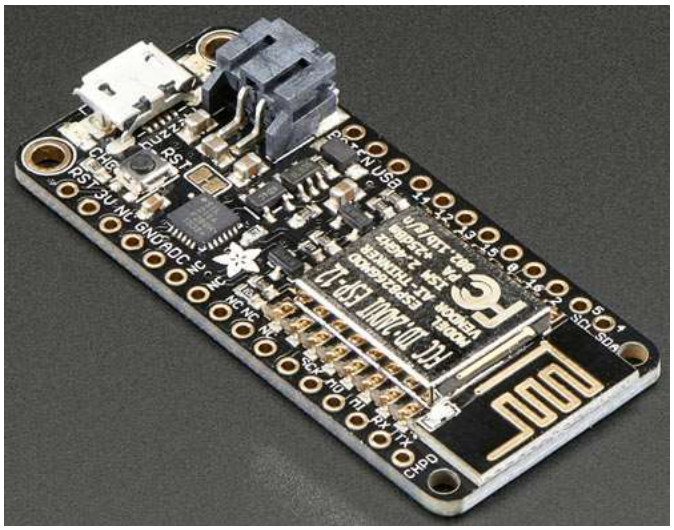


Рисунок 1.6 – Приклад плати ESP8266 від Adafruit

Кілька років тому компанія Adafruit почала зі спеціальних плат розробки, які дозволяють розробляти прототипи інтернету речей, що масштабуються. Крім плат розробки, як і платформа Particle.io, компанія Adafruit надає хмарні сервіси для пристроїв із клієнтськими бібліотеками для всіх основних платформ розробки обладнання інтернету речей, потужним API, красивими інформаційними панелями та універсальною безпечною платформою інтернету речей.

Основна відмінність між Adafruit та Particle полягає в тому, як розроблені їхні продукти. Adafruit.io сфокусована на спільноті розробників. Це рішення ідеально підходить для розробки проєктів прототипів. Платформа Particle.io, з іншого боку, має яскравіше виражений комерційний відтінок своїх продуктів.

1.2.5 Лінійка продуктів Arduino IoT

Плати Arduino [11] розпочали активно використовуватися задовго до того, як інтернет речей став мейнстрімом. Завдяки простоті програмування і легкості інтегрування в різні системи плати Arduino, що вбудовуються, відразу полюбилися всім, хто займається розробкою тих чи інших електронних проєктів. Ранні версії плат Arduino були переважно мікроконтролерами загального призначення, які підключалися до Інтернету за допомогою модулів GSM і Wi-Fi, але в міру того, як концепція інтернету речей почала отримувати все більший розвиток, стали розроблятися плати зі спеціальними функціями, що підтримують інтернет речей. (IoT). До них можна віднести такі плати, як Arduino 101 (розроблена спільно з Intel), MKR1000, Arduino WiFi Rev 2 та MKR Vidor 4000, яка є першою платою Arduino на основі чіпа FPGA (рис. 1.7).

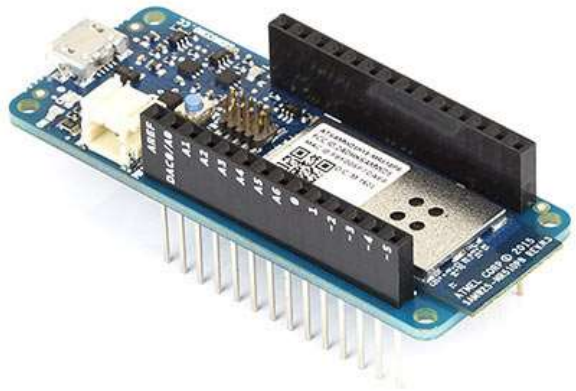


Рисунок 1.7 – Приклад плати Arduino MKR1000

Кожна з цих плат була створена з урахуванням технологій інтернету речей, і всі вони мають різні функції, які роблять їх більш підходящими для конкретних рішень, тобто кожна з цих плат ідеальна для використання у своїй конкретній ніші [12]. Наприклад, плата Arduino WiFi Rev 2 (рис. 1.8) поставляється з IMU (інерційний вимірювальний блок), що робить її зручною для створення програм на базі дронів.

Як і компанії Adafruit та Particle.io, Arduino також має хмарний сервіс, призначений для використання деякими платами Arduino, включаючи MKR1000, Arduino Yun/Yun Shield та Arduino 101/WiFi Shield 101. Хмара пристроїв Arduino (cloud.arduino.cc) пропонує виробникам простий

інструмент для підключення своїх пристроїв до Інтернету і вимагає мінімально короткого процесу налаштування для доступу до хмари Arduino.

Рисунок 1.8 – Плата Arduino WiFi Rev 2

1.2.6 Raspberry Pi [13]

Хоча плата Raspberry Pi більшою мірою є пристроєм загального призначення, було б несправедливо ігнорувати внесок співтовариства Raspberry у розробку низки продуктів та проєктів інтернету речей, які нині, як мовиться, у тренді. Як правило, плати Raspberry Pi досить дорогі і складні (виняток складає плата Raspberry Pi Pico, яка нещодавно з'явилася) для того, аби їх було доцільно використовувати для простого підключення будь-яких датчиків або виконавчих механізмів, проте вони знаходять застосування як агрегатори даних, концентратори і шлюзи пристроїв у проєктах інтернету речей. На рисунку 1.9 показано приклад плати Raspberry Pi 3 Model B+.

Рисунок 1.9 – Приклад плати Raspberry Pi 3 Model B+

Щоб привернути увагу промислової аудиторії Інтернету речей і загалом людей, які хотіли б використовувати Raspberry Pi у своїх продуктах, були запущені у виробництво обчислювальні модулі Raspberry Pi. Наприклад, обчислювальний модуль Raspberry Pi 3 (CM 3) (рис. 1.10) містить начинки плати Raspberry Pi 3 (процесор BCM2837 та 1 ГБ оперативної пам'яті), а також флеш-пристрій eMMC об'ємом 4 ГБ (що еквівалентно SD-карті в Pi), що працює на частоті процесора 1,2 ГГц, все це інтегровано на невеликій платі розміром 67,6x31 мм, яка вписується в стандартний роз'єм DDR2 SODIMM (той самий тип роз'єму, який використовується для пам'яті ноутбуків). Подібний функціонал робить продукти Raspberry придатним для використання як шлюзи та в проєктах, що вимагають високої швидкості обробки даних.

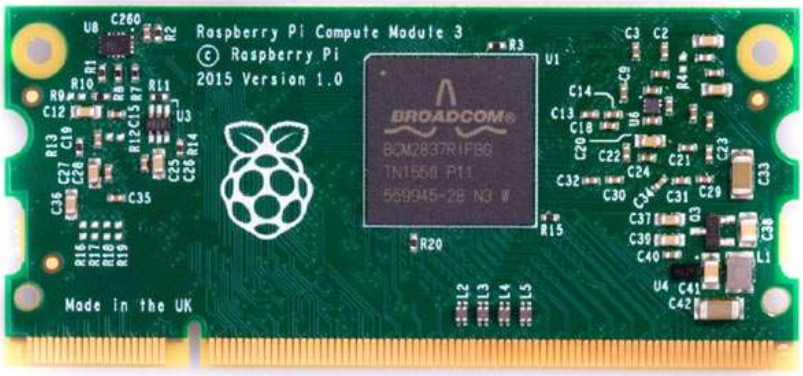


Рисунок 1.10 – Raspberry Pi 3 (CM 3)

1.3 Особливості розробки програм для Arduino на C++

1.3.1 Основні відомості про програмування Arduino

Arduino – це електронна платформа з відкритим вихідним кодом, заснована на простому у використанні апаратному та програмному забезпеченні [14]. Плати Arduino здатні зчитувати вхідні дані, оброблювати їх та перетворювати на вихід – активуючи, наприклад, двигун, вмикаючи світлодіод, або публікуючи щось в Інтернеті. Для цього використовується мова програмування Arduino (на основі Wiring) і програмне забезпечення Arduino (IDE) на основі обробки. Ця мова програмування заснована C++, з додаванням деяких функцій, які полегшують роботу з Arduino.

Програмування на C++ дозволяє використовувати більше можливостей Arduino. Наприклад, будь-який порт можна налаштувати як вхід і як вихід, а кількість використовуваних портів обмежується тільки їх наявністю на платі.

Для програмування плат Arduino існує спеціальне інтегроване середовище розробки Arduino IDE. Розглянемо зовнішній вигляд програми (рис. 1.11).

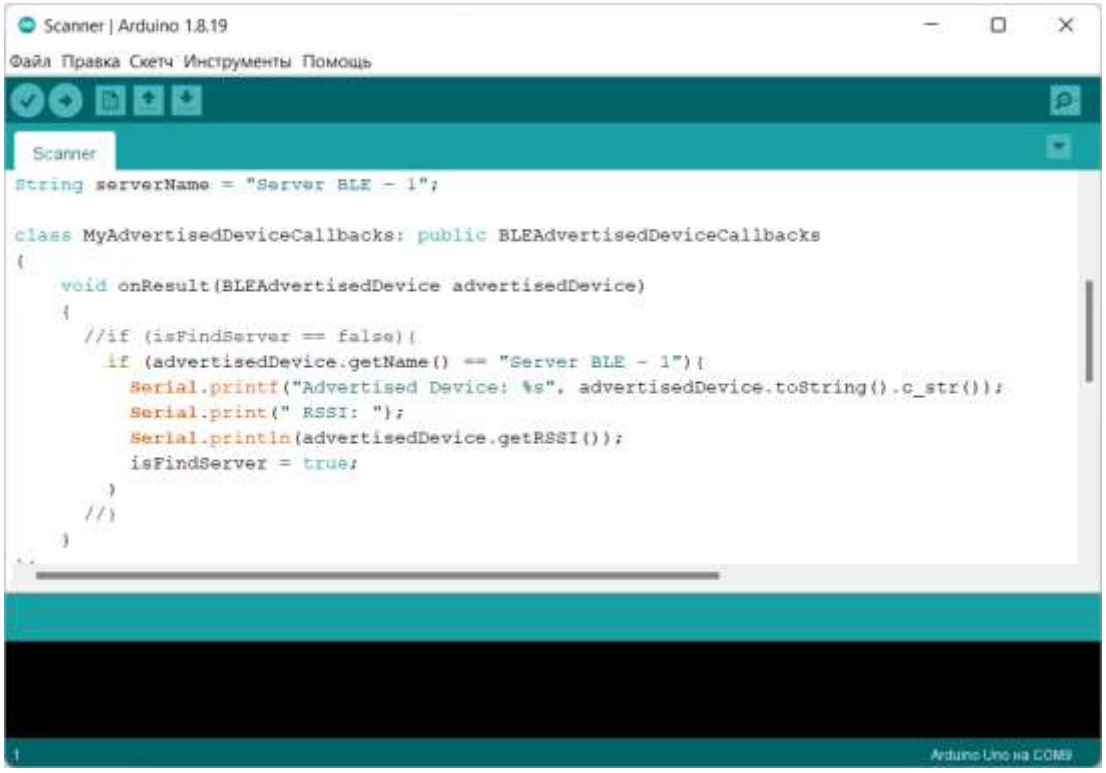


Рисунок 1.11 – Інтегроване середовище Arduino IDE

Вгорі, під рядком заголовка знаходиться рядок Меню з такими пунктами: Файл, Правка, Скетч, Налаштування, Сервіс.

Під рядком меню розташовується рядок інструментів, на який винесені часто використовувані команди:

– перевірити;

– завантажити;

– створити;

– зберегти;

– відкрити;

– монітор порту.

Нижче розташовується рядок з назвами відкритих вкладок. В кінці рядка знаходиться піктограма, з натисканням на яку відкривається меню, що належить до вкладок.

У центрі вікна знаходиться робоча область, в якій пишеться код програми. Під робочою областю знаходиться вікно повідомлень, куди виводяться повідомлення про результат компіляції, завантаження і т.д.

1.3.2 Структура програми для Arduino

Програма, написана в IDE Arduino, називається скетчем. Кожен скетч має складатися як мінімум з двох функцій.

Функція – структурна одиниця програми, яка має ім'я і містить деяку послідовність дій (рис. 1.12).

```
void setup() {
// Помістіть ваш код установок тут, він запуститься один раз
}
void loop() {
// Розмістить основний код тут, щоб запустити його на циклічне повторення
}
```

Рисунок 1.12 – Приклад основних функцій

Розглянемо структуру простої програми, що наведена на рис. 1.12. На початку програми, перед функцією setup, зазвичай оголошуються змінні. Після вмикання живлення плати першою виконується функція setup. Вона виконується тільки один раз. Зазвичай у ній ініціалізуються режими роботи портів: порти, до яких підключені різні датчики, встановлюються як входи (INPUT), а порти з виконавчими пристроями як виходи (OUTPUT).

Код, написаний у функції loop(), починає виконуватися після функції setup() і виконується в нескінченному циклі знов і знов. У цій функції відбувається основна робота: різні обчислення, отримання значень датчиків, вивід значень на порти. У таблиці нижче наведені у відповідності блоки з ArduBlock та їх аналоги – функції з C++.

1.3.3 Коментарі та команди

Коментар – це текст, який призначений для програмістів і не обробляється комп'ютером. Зазвичай коментарі використовуються для створення приміток до коду для подальшого використання. Коментарі можна використовувати під час тестування, щоб зробити неактивними певні рядки коду.

Коментарі в C ++ записуються одним з таких способів:

– символи «/*» (коса риска і зірочка), за якими йде будь-яка послідовність символів, включаючи переклади рядка, після чого ставляться символи «*/». Наприклад:

```
/*
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
```

```
analogWrite(ena,100);
analogWrite(enb,100);
/*
```

– символи «//» (дві косі риски), за якими йде будь-яка послідовність символів. Такі коментарі однорядкові, тому що коментують тільки код на одному рядку:

```
// analogWrite (ena, 100);
```

Мова С++ являє собою набір команд, які повідомляють комп'ютеру, що необхідно зробити. Цей набір команд зазвичай називається вихідний код або просто код. Командами є або «функції», або «ключові слова». Ключові слова (зарезервовані слова) є основними блоками побудови мови. В кінці кожної команди ставиться крапка з комою:

```
analogWrite (ena, 100);
```

1.3.4 Зміні. Основні типи даних у С++

Змінна – це виділена область («комірка») в пам'яті комп'ютера, в якій зберігатиметься інформація. У кожної змінної має бути своє унікальне ім'я на латиниці з урахуванням великих і маленьких літер. Наприклад, змінні «a» і «A» комп'ютер сприйматиме по-різному, навіть якщо вони відповідатимуть одним і тим самим числам.

Основна мета будь-якої програми полягає в обробці даних. Дані різного типу зберігаються й обробляються по-різному. У будь-якій алгоритмічній мові кожна константа, змінна, результат обчислення виразу або функції повинні мати певний тип.

Тип даних визначає:

– внутрішнє подання даних у пам'яті комп'ютера;

– безліч значень, які можуть приймати величини цього типу;

– операції та функції, які можна застосовувати до величин цього типу.

Виходячи з цих характеристик, програміст обирає тип кожної величини, яка використовується в програмі для подання реальних об'єктів.

Для опису основних типів даних визначені такі ключові слова:

– int (цілий): всі числа від -2147483648 до 2147483647;

– float (число з плаваючою точкою): всі числа від 3,4·10-38 до 3,4·1038. Використовувати тільки значення з точкою, навіть якщо воно ціле (10.0). Під час ділення цілочислового типу з метою отримати число з плаваючою точкою писати (float) перед обчисленням;

– double (суттєвий з подвійною точністю) всі числа від 3,4·10-308 до 3,4·10308;

– char (символьний): наприклад ‘a’, ‘1’ (числа теж можуть бути символами);

– bool (логічний): true або false. Внутрішня форма подана значенням false - 0. Будь-яке інше число інтерпретується як true.

Під час оголошення змінних можна повідомити їм значення шляхом розміщення знаку рівності та константи після імені змінної. Цей процес називається ініціалізацією і в загальному випадку має вигляд:

тип ім’я_змінної = константа;

Нижче наведено декілька прикладів:

```
char ch = 'a';
int first = 0;
float balance = 126.23;
```

1.3.5 Приклад простої програми для Arduino

Для прикладу розглянемо просту програму управління підключенням світлодіоду, що вбудовано в плату.

Для миготіння світлодіодом слід реалізувати алгоритм: підключити світлодіод; почекати (delay); вимкнути світлодіод (LOW); почекати. Всі дії виконуються постійно, тому ці команди будуть описані в void loop (). На рисунку 1.13 наведено приклад коду.

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


void loop() {
  digitalWrite(13, HIGH);   // ввімкнути світлодіод (HIGH - високий рівень напруги)
  delay(1000);              // чекати секунду
  digitalWrite(13, LOW);    // вимкнути світлодіод (LOW - низький рівень напруги)
  delay(1000);              // чекати секунду
}
```

Рисунок 1.13 – Приклад коду для миготіння світлодіодом

Стрілками на рисунку показані відкриваючі та закриваючі дужки, що відокремлюють код головної функції loop().

Виконаємо модифікацію коду, додавши змінні для зберігання номера порту контролеру, який відповідає за підключений світлодіод.

На рисунку 1.14 показано приклад ініціалізації 13 порту та призначення його змінній led.

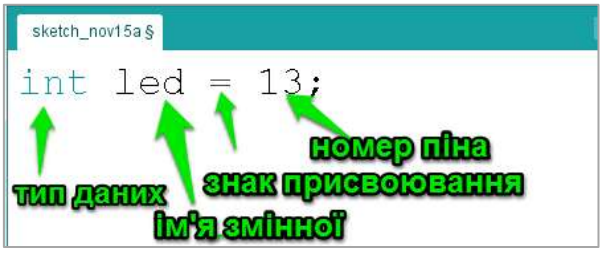


Рисунок 1.14 –Приклад ініціалізації 13 порту контролеру

Далі треба у всіх командах змінити число «13» на назву змінної «led» (рис. 1.15).

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


void loop() {
  digitalWrite(led, HIGH);   // ввімкнути світлодіод (HIGH - високий рівень напруги)
  delay(1000);               // чекати секунду
  digitalWrite(led, LOW);    // вимкнути світлодіод (LOW - низький рівень напруги)
  delay(1000);               // чекати секунду
}
```

Рисунок 1.15 – Використання імені змінної замість значення

Додаємо змінну для визначення часу перемикання світлодіоду (рис. 1.16).

```
int led = 13;
int t = 1000;
void setup() {
  pinMode(led, OUTPUT);
}


void loop() {
  digitalWrite(led, HIGH);   // ввімкнути світлодіод (HIGH - високий рівень напруги)
  delay(t);                  // чекати секунду
  digitalWrite(led, LOW);    // вимкнути світлодіод (LOW - низький рівень напруги)
  delay(t);                  // чекати секунду
}
```

Рисунок 1.16 – Використання змінної «t» для визначення часу

Після модифікації програми необхідно заново виконати компіляцію коду та завантаження його в контролер Arduino.

1.3.6 Виведення даних на монітор послідовного порту

Виведення повідомлення на екран комп'ютера здійснює плагін програми Arduino IDE, який називається монітор порту (рис. 1.17).

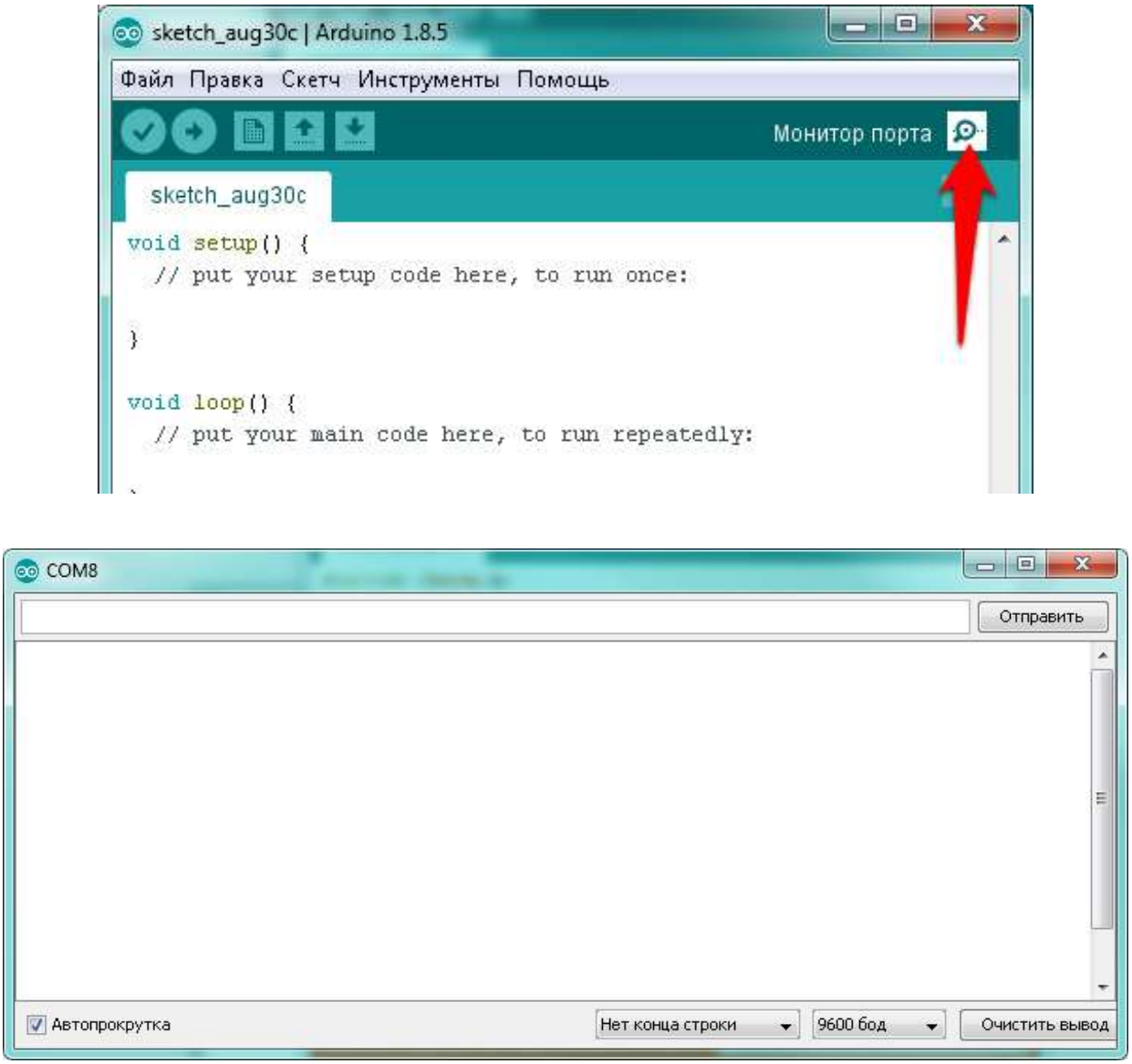


Рисунок 1.17 – Підключення і зовнішній вигляд монітора порту

Розглянемо приклад організації послідовної передачі даних між ПК і Arduino зі швидкістю 9600 біт за секунду. Для цього ініціюємо використання послідовного інтерфейсу передачі даних командою Serial.begin(9600).

Далі виведемо на екран текст «Hello World!» за допомогою команд Serial.println(), або Serial.print(). Перша виводить текст, вказаний у дужках на екран з нового рядка (рис. 1.18). Друга виводить текст на екран поспіль (рис. 1.19). Відмінність написання функцій полягає в закінченні «ln».

Рисунок 1.18 – Результат виконання функції Serial.println()

Рисунок 1.19 – Результат виконання функції Serial.print()

Для виведення на монітор:

– тексту – використовуються подвійні лапки. Зверніть увагу, що на екран виведеться весь текст, включаючи пробіли;

– символу – використовуються одинарні лапки;

– значення змінної – використовується ім'я змінної.

Варіанти перегляду тексту на екран зображено на рисунку 1. 20.

1.3.7 Арифметичні операції С++

У сучасному житті дуже складно обійтися без арифметичних операцій. Нам постійно доводиться щось рахувати: складати, множити, віднімати, ділити тощо. Програмування – не виняток. Нам у 99.9% випадків доведеться ними користуватися в ході написання своїх програм. Більшість арифметичних операторів – збігаються з математичними і в написанні, і відповідно до дії (табл. 1.1).

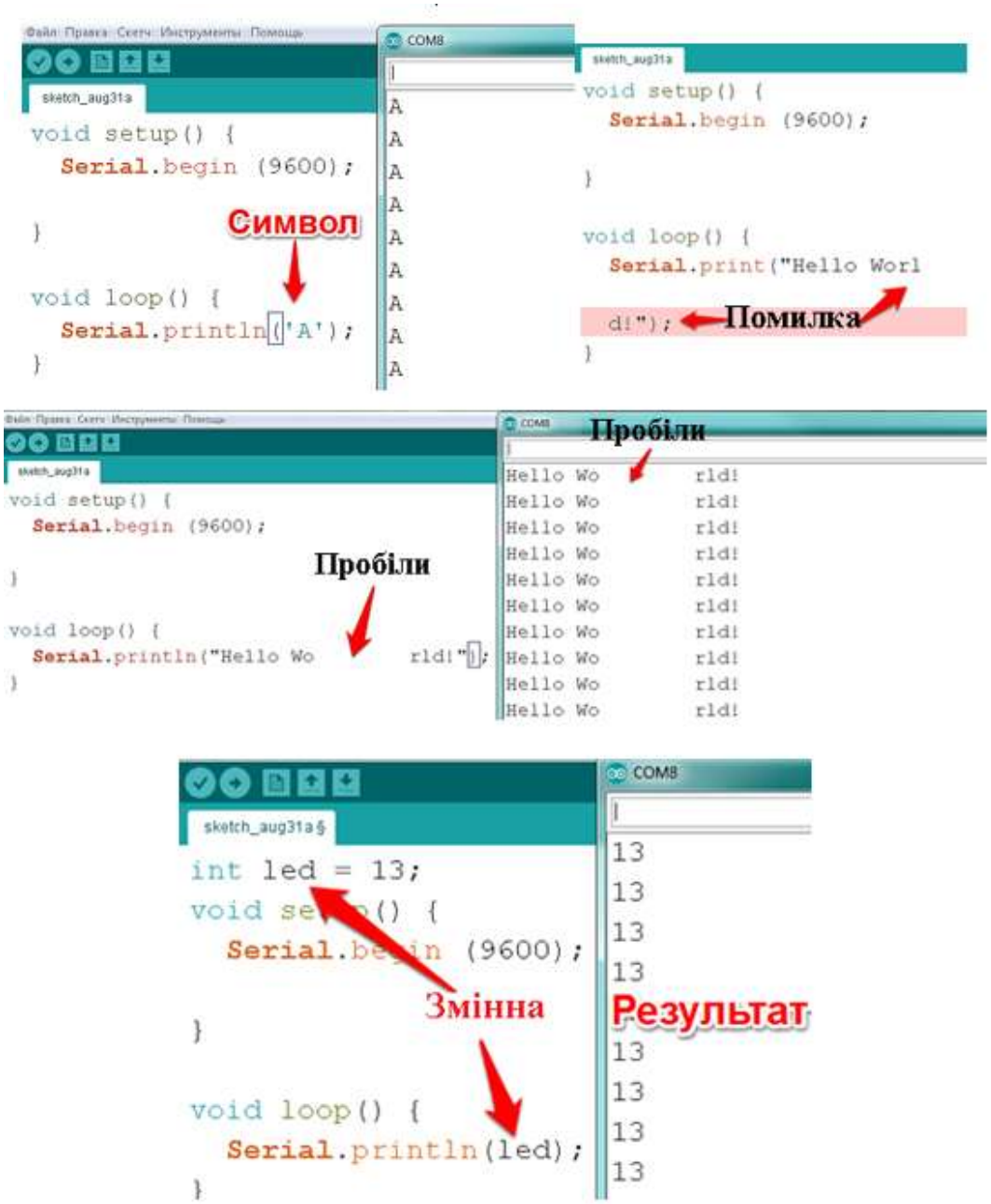


Рисунок 1.20 – Варіанти перегляду тексту на екрані

Таблиця 1.1 – Арифметичні операції в С++

Оператор	Операція
+	Додавання даних
-	Віднімання даних
*	Множення даних
/	Розподіл даних

Приклад коду з виконанням арифметичної операції і результат зображені на рисунку 1.21. Зверніть увагу на коментарі.

Рисунок 1.21 – Виконання арифметичної операції

1.3.8 Читання цифрового і аналогового входу

Arduino IDE має набір стандартних функцій для керування входом/виходом. Розробник може легко обробляти як цифрові сигнали, так і аналогові.

Цифрові входи і виходи (цифрові I/O) на платах Arduino дозволяють підключати до Arduino датчики, приводи та інші мікросхеми. Вивчення того, як використовувати їх, дозволить вам використовувати Arduino для виконання реально корисних речей, таких як читання стану вхідних перемикачів, підсвічування індикаторів та керування релейними виходами.

На відміну від аналогових сигналів, які можуть приймати будь-яке значення в межах діапазону, цифрові сигнали мають тільки два окремих значення: високий (HIGH, 1) і низький (LOW, 0) рівні. Цифрові сигнали можна використовувати в ситуаціях, де вхід або вихід прийматиме одне з цих двох значень. Наприклад, один з випадків, коли можна використовувати цифровий сигнал – це увімкнення і вимкнення світлодіоду.

Функція digitalRead(pin_number) зчитує цифрове значення з цифрового виводу: pin_number – це номер цифрового I/O виведення, який ви збираєтеся прочитати. Ця функція повертає одне з двох значень: HIGH або LOW. Якщо вивід ні до чого не приєднаний, функція digitalRead() може повернути випадкову величину, як HIGH, так і LOW. Приклад використання функції зображений на рисунку 1.22.

Для того щоб прочитати показання на аналоговому вході як аналогові, слід викликати функцію analogRead (pin_number). Приклад використання функції зображений на рисунку 1.23.

Рисунок 1.22 – Використання функції digitalRead()

Рисунок 1.23 – Використання функції analogRead()

Порти контролера, які є аналоговими входами, можуть також використовуватися як цифрові входи під ім'ям A0, A1 і т.д. (рис. 1.24).

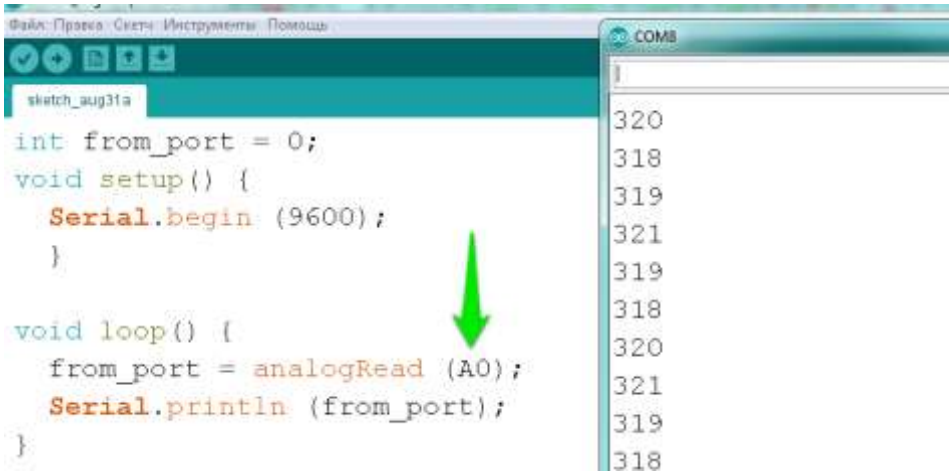


Рисунок 1.24 – Використання порту без ініціалізації змінної

1.3.9 Умовний оператор

Іноді, залежно від умови, потрібно виконати різні дії. Для цього використовується оператор if (рис. 1.25).

Рисунок 1.25 – Приклад умови вибору

Якщо умова вірна, тобто істинна, то виконується спеціально вказаний для цього випадку фрагмент коду. Якщо ж умова не вірна, тобто хибна, то виконується або інша спеціально зазначена частина коду, або не виконується нічого і робота мікроконтролера триває далі за кодом.

Конструкція оператора вибору – це безпосередня вказівка оператора, потім необхідна умова в круглих дужках і певний код, що виконується в разі істинності цієї умови. Але у випадках, коли важливо враховувати не тільки істинність, а й хибність, тобто не виконання поставленої вами умови, після фігурних дужок пишеться слово else, що в перекладі означає «інакше», і ставляться такі самі фігурні дужки, тільки вже код у них виконуватиметься в разі хибності заданої умови (рис.1.26).

Оскільки мікроконтролер працює з числами і різними значеннями, то в умові ми можемо перевіряти не тільки абсолютну рівність двох значень, але також робити різні висновки про те, чи більше це значення ніж задане, більше або дорівнює, менше, менше або дорівнює або просто не дорівнює (табл. 1.2).

```
void loop() {
  if (УМОВА) {
    код, який виконується. якщо умова правильна
  }
  else {
    код, який виконується. якщо умова НЕ правильна
  }
}
```

Рисунок 1.26 – Приклад запису умовного оператора

Таблиця 1.2 – Оператори порівняння

Операція	Опис	Приклад
A > C	A більше C	5 > 1
N >= K	N більше або дорівнює K	7 >= 7
Y < E	Y менше E	6 < 10
T<= N	T менше або дорівнює N	1 <= 20
A != B	A не дорівнює B	8 != 10
A == C	A дорівнює C	20 == 20

Розглянемо приклад використання умовного оператора. Згенеруємо число від 0 до 3. Якщо число дорівнює 3, то вивести повідомлення. Якщо число не дорівнює 3, вивести повідомлення про невиконання умови (рис. 1.27).

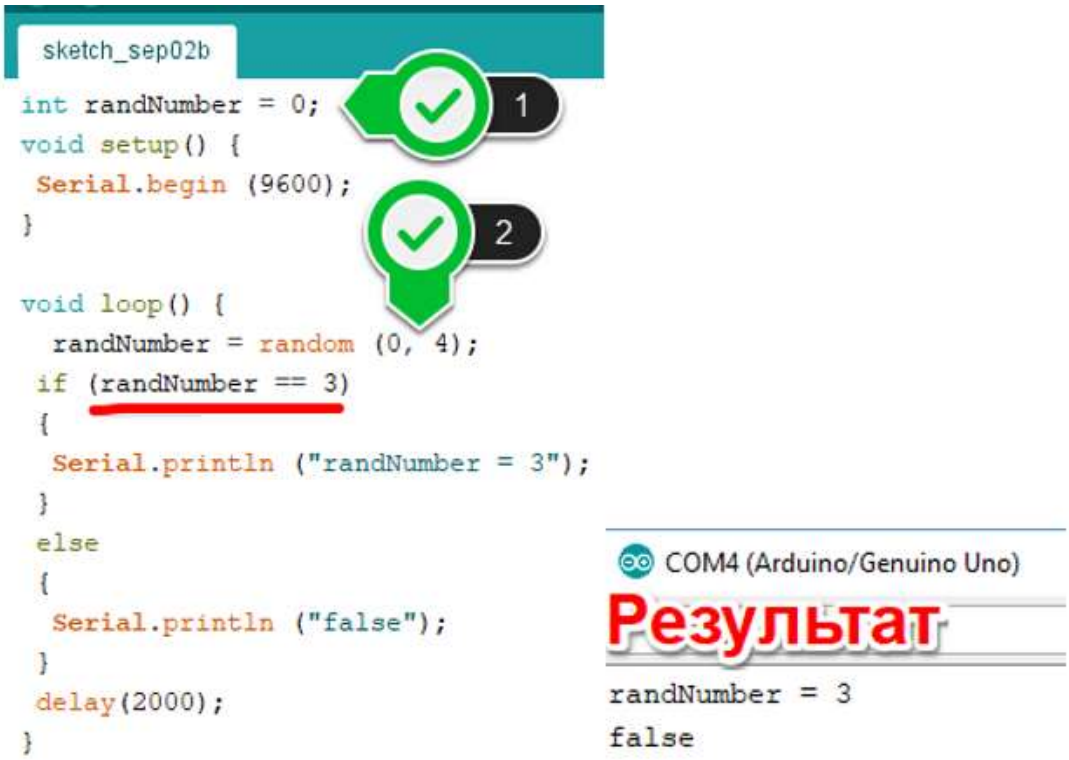


Рисунок 1.27 – Приклад програми з використанням умовного оператора

1.3.10 Оператор вибору

Оператор if дуже простий і зручний у використанні, але тут варто зазначити, що з ним зручно працювати, тільки якщо нам необхідно перевірити якусь одну умову. А якщо їх буде декілька? Уявіть, що вам дали завдання обробити чотири сигнали потенціометра, вивести їх на екран. Частина коду зображена на рисунку 1.28.

```
P_resistor = analogRead (A0);
if (P_resistor == 100)
{
  Serial.println ("P_resistor = 100");
}
if (P_resistor == 200)
{
  Serial.println ("P_resistor = 200");
}
if (P_resistor == 300)
{
  Serial.println ("P_resistor = 300");
}
if (P_resistor == 400)
{
  Serial.println ("P_resistor = 400");
}
```

Рисунок 1.28 – Неправильне використання умовного оператора

Такий каскад із умов з боку мікроконтролера не створить для нього жодних проблем, але з точки зору сприйняття людиною такої великої кількості однотипних умовних операторів – це вкрай незручно.

У таких випадках використовують ще один оператор вибору, який має назву switch, що в перекладі означає комутатор або перемикач. Його зручність полягає в тому, що нам не потрібно тепер створювати щоразу нову умову, аби перевірити одне й те саме значення, а лише необхідно вказати в круглих дужках оператора змінну, яка прийматиме це значення, а сам оператор здійснить пошук з можливих варіантів потрібного нам значення і виконає відповідний йому код (рис. 1.29).

Використовуючи switch, попереднє завдання реалізовується, як вказано на рисунку 1.30.

```
switch (ЗМІННА){
  case значення 1:
  /*   якщо ЗМІННА дорівнює ЗНАЧЕННЯ 1, то виконується код,
   що розміщується до оператора break  */
  break;
  case 2:
   /*   якщо ЗМІННА дорівнює ЗНАЧЕННЯ 2, то виконується код,
   що розміщується до оператора break  */
  break;
  case 3:
   /*   якщо ЗМІННА дорівнює ЗНАЧЕННЯ 3, то виконується код,
   що розміщується до оператора break  */
  break;
  default:
  /*
   якщо ЗМІННА НЕ дорівнює жодному з вказаних значень,
   то виконується код, вказаний в цьому блоці та відбувається вихід
  */
}
```

Рисунок 1.29 – Використання оператора вибору switch

```
 P_resistor = analogRead (A0);
switch (P_resistor)
 {
   case 100:
   Serial.println ("P_resistor = 100");
   break;

   case 200:
   Serial.println ("P_resistor = 200");
   break;

   case 300:
   Serial.println ("P_resistor = 300");
   break;

   case 400:
   Serial.println ("P_resistor = 400");
   break;

   default:
   Serial.println ("P_resistor = else");
 }
```

Рисунок 1.30 – Приклад використання оператора вибору switch

1.3.11 Генератор випадкового числа

Досить часто в робототехніці потрібно отримувати випадкові числа для виконання будь-яких дій. Для того щоб отримати випадкове число, можна, звичайно, написати свою функцію, а можна використовувати і стандартні. Для генерації випадкових чисел у C++ використовують генератор псевдовипадкових чисел.

Генератор псевдовипадкових чисел (ГПВЧ, англ. Pseudorandom number generator, PRNG) – алгоритм, який породжує послідовність чисел, елементи якої майже незалежні один від одного та підкоряються заданому розподілу (зазвичай рівномірному). Іноді генератор псевдовипадкових чисел називають генератором випадкових чисел.

В Arduino IDE реалізовано дві функції для генерації псевдовипадкових чисел: random() і randomSeed().

Функція random() дозволяє генерувати псевдовипадкові числа із вказаного діапазону. Синтаксис команди:

random(x); // діапазон [0; x-1] або від 0 до (x - 1)
random(a,b); // діапазон [a; b -1] від a до (b -1)

Вказавши один параметр, ми генеруємо число від нуля до числа, на 1 менше від параметра. Для двох параметрів діапазон починається від першого параметра, а закінчується значенням, на 1 менше від другого параметра. Приклад отримання випадкових чисел наведено на рисунку 1.31.

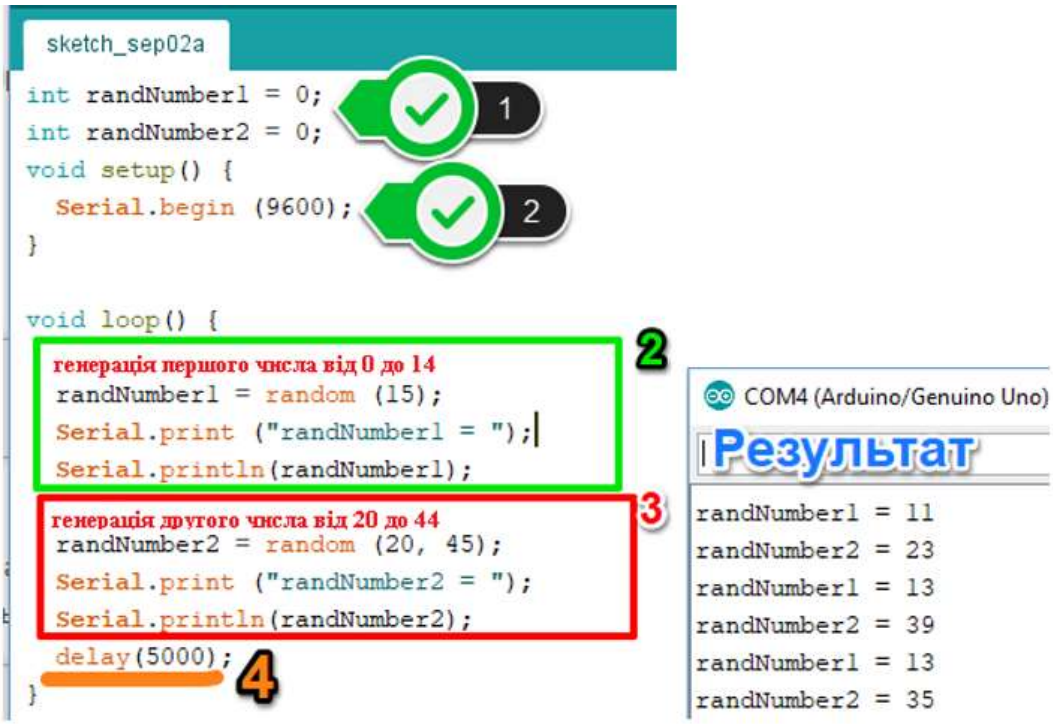


Рисунок 1.31 – Використання функції random()

Якщо з кожним запуском програми необхідно отримувати різні послідовності значень, що генеруються функцією random(), то слід ініціалізувати генератор псевдовипадкових чисел з випадковим параметром. Наприклад, можна використовувати значення, що віддається функцією analogRead() з непідключеного порту вхід/виходу (рис. 1.32).

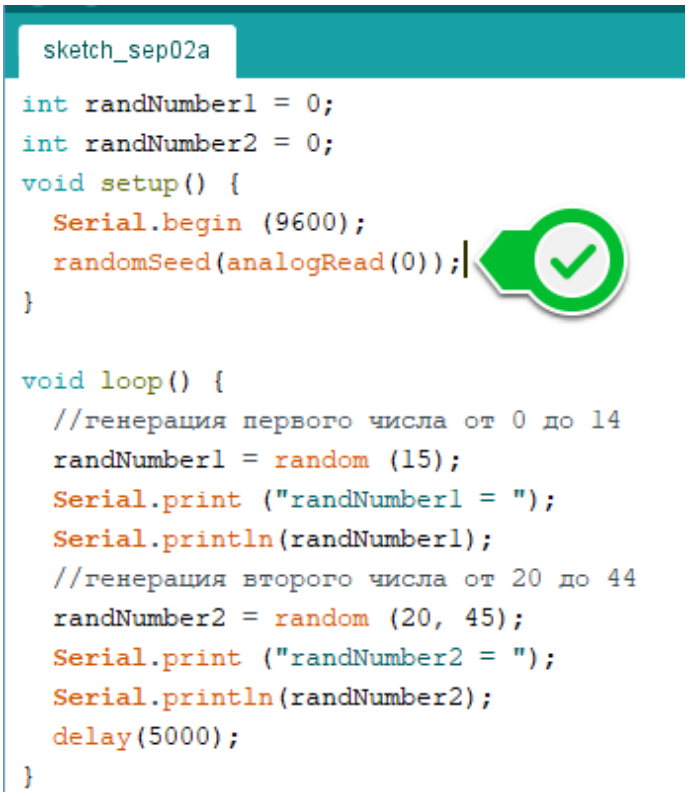


Рисунок 1.32 – Використання функцій analogRead() та randomSeed ()

У деяких випадках необхідно отримувати однакову послідовність з кожним запуском програми на Arduino. У цьому випадку форматувати генератор псевдовипадкових чисел слід викликом функції randomSeed() з фіксованим параметром.

1.3.12 Оператор циклу while

Цикл – багаторазове проходження одним і тим самим кодом програми. Цикли необхідні програмісту для багаторазового виконання одного й того самого коду, поки істинна певна умова. Якщо умова завжди істинна, то такий цикл називається нескінченним, у такого циклу немає точки виходу.

Цикл while перекладається як «поки» і до тих пір, поки його умова виконується, він нескінченно довго виконує код, вказаний у його фігурних дужках. Наприклад, на будь-якому етапі роботи мікроконтролера нам важливо отримати сигнал з якого-небудь піна Arduino, і, поки сигналу немає, програма не повинна нічого виконувати. Саме в цьому випадку використовується цикл

while, якщо в умові ми вкажемо зчитування піна і його перевірку на рівність нулю (рис. 1.33). Код, який знаходитися між фігурними дужками {}, називається тілом циклу.

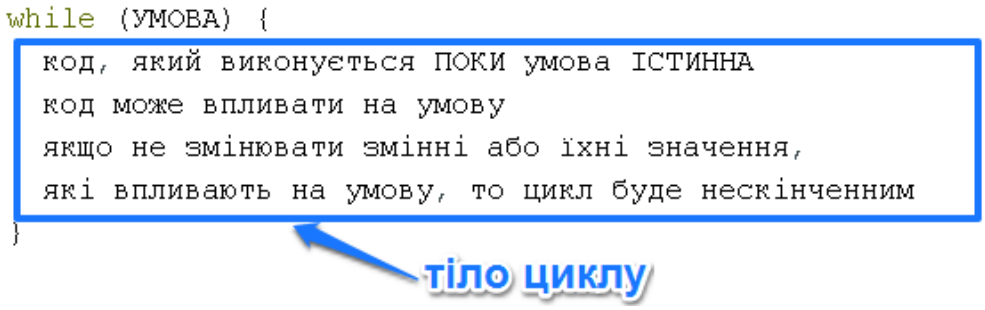


Рисунок 1.33 – Конструкція циклу while

Приклади використання циклу while зображені на рисунку 1.34. Зверніть увагу на оголошення змінної у функції loop(). У C++ змінні можна оголошувати не тільки перед setup(). У цьому випадку змінні, описані в loop(), можна використовувати тільки в межах її фігурних дужок {}.

```
void loop() {
  приклад 1
  bool signal_PIN3;оголошення змінної, яка приймає значення або 0 або 1
  while (signal_PIN3 == 0)цикл виконується поки на вході 3 низький рівень сигналу (0)
  {
    signal_PIN3 = digitalRead (3); зчитує значення з 3-го піна
  }
  приклад 2
  int n = 0;
  while (n <= 5) цикл виконується поки змінна менше 5
  {
    n = n +2; до кожного "старого" значення n додати 5
  } після виконання циклу n = 6
}
```

Рисунок 1.34 – Приклади використання циклу while

Важливо ще раз наголосити, що while виконуватиметься до тих пір, поки умова буде істинна, і як тільки вона стане хибна, під час чергової перевірки умови станеться вихід з циклу.

1.3.13 Оператор циклу do while

Оператор циклу do while називається оператором циклу з післяумовою (рис. 1.35) і працює практично за тим самим принципом, що й while, але з однією лише відмінністю: спершу відбувається виконання тіла циклу, а вже потім перевірка умови. Цей цикл використовується, якщо необхідно

хоча б один раз виконати код, вказаний у фігурних дужках незалежно від істинності чи хибності умови, зазначеної нижче. У випадку зі звичайним циклом while з попередньо заданою помилковою умовою мікроконтролер переходить відразу ж до виконання коду, написаного після фігурних дужок циклу і, таким чином, пропустить код, вказаний у тілі циклу.

```
do
{
  код, який виконується ПОКИ умова істинна.
  цикл буде нескінченним, якщо код не впливає
  на змінні або значення в умові
} while (УМОВА)
```

Рисунок 1.35 – Конструкція циклу do while

Зверніть увагу на синтаксис оператора do while: після умови в круглих дужках ставиться крапка з комою, оскільки тіло циклу вже було описано в фігурних дужках вище.

1.3.14 Оператор циклу for

Цикл, який діє за принципом виконання заданої кількості ітерацій, називається циклом for: в умові ви вказуєте дію, яка виконується на початку циклу, потім саму умову роботи та дію, яка виконується в кінці кожного проходження циклу (рис. 1.36).

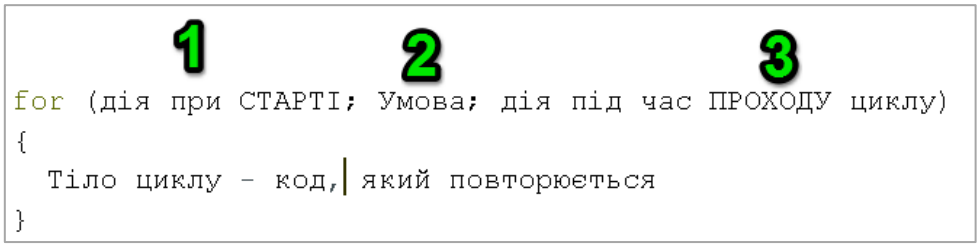


Рисунок 1.36 – Конструкція циклу for

На рисунку 1.37 показано приклад застосування декількох циклів For для почергового підключення світлодіодів задану кількість разів.

Спершу вказується ім'я оператора – for. Потім у дужці, яка відкривається, вказується дія на початку циклу – в нашому випадку це ініціалізація змінної pin і присвоюється їй значення 2, оскільки до другого піну підключений перший світлодіод. До речі, якщо не вказати, чому спочатку дорівнює змінна pin, то компілятор за замовчуванням присвоїть їй нульове значення. Потім ставимо

крапку з комою і вказуємо умову, за якої цикл має виконуватися – в нашому випадку це робота циклу, поки змінна pin менша або дорівнює 11. Після чого знову ставимо крапку з комою і вказуємо крок збільшення або зменшення значення pin. Запис pin++ означає збільшення змінної pin на одиницю, така операція називається інкремент і рівносильна запису pin = pin+1. А запис pin-- навпаки, зменшить значення pin на один, ця операція називається декрементом.

```
void loop() {
  for (int pin = 2; pin <=11; pin++)

    {
      digitalWrite(pin, HIGH);
      delay(100);
    }
  for (int pin = 2; pin <=11; pin++)   цикл для вимкнення світлодіодів
    {
      digitalWrite(pin, Low);
      delay(100);
    }
}
```

Рисунок 1.37 – Використання декількох циклів for

1.3.15 Функції та їх застосування

Розбиття коду на функції дозволяє створювати частини коду, які виконують певні завдання. Після виконання функції відбувається повернення в місце, звідки вона була викликана. Причиною створення функції є необхідність виконувати однакову дію кілька разів.

Існують дві обов'язкові функції в скетчах Arduino setup() і loop(). Інші функції мають створюватися за дужками цих функцій. У наступному прикладі буде створена проста функція множення двох чисел (рис. 1.38).

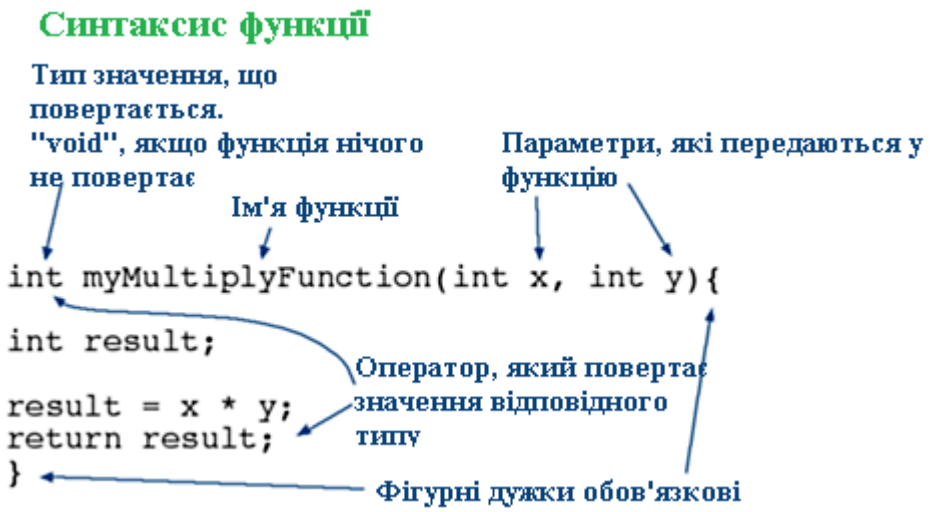


Рисунок 1.38 – Приклад опису і виклику функції в Arduino IDE

Ім'я функції і тип значення, яке повертається, називається прототипом функції. Для виклику функції множення їй передаються параметри даних:

```
void loop(){
  int i = 2;
  int j = 3;
  int k;
  k = myMultiplyFunction(i, j); // k містить 6
}
```

Створену функцію необхідно задекларувати поза дужками будь-якою іншою функцією, таким чином «myMultiplyFunction()» може стояти вище або нижче функції «loop()».

Весь скетч матиме такий вигляд:

```
void setup(){
 Serial.begin(9600);
}
void loop(){
 int i = 2;
 int j = 3;
 int k;
 k = myMultiplyFunction(i, j); // k містить 6
 Serial.println(k);
 delay(500);
}
int myMultiplyFunction(int x, int y){
 int result;
 result = x * y;
 return result;
}
```

Наступна функція зчитуватиме дані з датчика функцією analogRead() і розраховуватиме середнє арифметичне, потім створена функція буде масштабувати дані за 8 бітами (0-255) та інвертувати їх:

```
// датчик підключений до виводу 0
int ReadSens_and_Condition(){
 int i;
 int sval;
 for (i = 0; i < 5; i++){
  sval = sval + analogRead(0);  // сенсор на аналоговому вході 0
```

```
 }
 sval = sval / 5;  // середнє
 sval = sval / 4;  // масштабування по 8 бітам (0 - 255)
 sval = 255 - sval; // інвертування вихідного значення
 return sval;
}
```

Виклик функції здійснюється присвоєнням її змінної:

```
int sens;
sens = ReadSens_and_Condition();
```

2 КЕРУВАННЯ ПРИСТРОЯМИ IOT ЗА ДОПОМОГОЮ ARDUINO

2.1 Управління двигунами постійного струму

Мотор-редуктор – це мікроелектродвигун постійного струму (рис. 2.1), що встановлено в пластмасовий корпус, в якому знаходиться понижуючий швидкість обертання редуктор з пластмасових шестерень і який збільшує зусилля на валу механізму [17]. На вал мотор-редуктора насаджується колесо з гумовою покришкою. Вал виходить з двох сторін корпусу редуктора. Колесо може встановлюватися з будь-якого боку корпусу. На другу частину валу мотор-редуктора встановлюють диск з отворами, що дозволяє працювати оптичному датчику контролю параметрів обертання вала.

Рисунок 2.1 – Мотор-редуктор

Характеристики мотор-редуктора наведені в таблиці 2.1.

Таблиця 2.1 – Характеристики мотор-редуктора

Напруга живлення, В: – номінальна – гранична – критична	 6 8 < 3, та > 12
Струм, мА: – холостого ходу – під час роботи	 70…120 670
Передавальне число редуктора	48:1
Швидкість обертання без навантаження з живленням 6 В	90 обертів за хвилину
Крутний момент з живленням 6 В	2 кілограми на сантиметр
Шум	65 dB

Напрямок обертання залежить від полярності прикладеного живлення. Із підключенням мотор-редуктор короткий час споживає більше струму, ніж під час обертання ротора двигуна після закінчення розгону. Якщо мотор-редуктор зупинений надмірним навантаженням на валу (загальмований), то споживання струму значно зростає. Нагрівання корпусу свідчить про перевантаження мотора. Температура корпусу мотора говорить про те, в якому режимі роботи він знаходиться. Чим сильніше нагрівається мотор, тим менше він прослугує. Сильне нагрівання призводить до псування мотора і редуктора.

Для управління двигуном постійного струму можна використовувати модуль L298N H-bridge. Він використовується для пристроїв, напруга живлення яких знаходиться в діапазоні від 5 до 35 В. Крім того, на багатьох подібних платах є вбудований 5 В регулятор, який дає можливість живити інші пристрої.

Перш ніж перейти до керування двигуном постійного струму і кроковим двигуном, розберемося з підключенням модуля L298N (рис. 2.2).

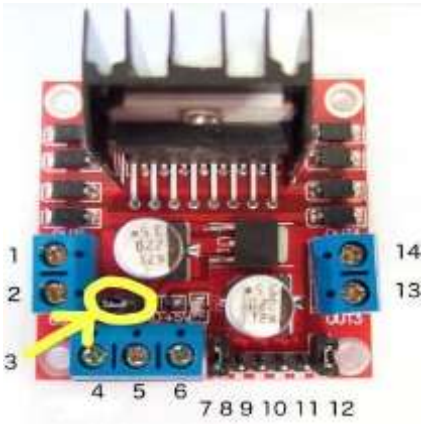


Рисунок 2.2 – Драйвер двигуна постійного струму L298N

Виходячи з наведеного рисунка, драйвер підключається так:

– контакт 1 – «+» для двигуна постійного струму №1, або для крокового двигуна A+;

– контакт 2 – «-» для двигуна постійного струму №1, або для крокового двигуна A–;

– контакт 3 – перемикач на 12 В, потрібно його зняти, якщо використовується напруга живлення більше 12 В;

– контакт 4 – живлення двигуна забезпечується за цього виходу. Максимальна напруга живлення постійним струмом 35 В.

– контакт 5 – GND (земля);

– контакт 6 – живлення 5 В, якщо перемикач на 12 В замкнутий. У даному випадку використовується для живлення Arduino і т.п.;

– контакт 7 – перемикач для двигуна постійного струму №1. Можна підключити до ШІМ-виходу для керування швидкістю двигуна постійного струму;

– контакт 8 – вхід від контролера (IN1);

– контакт 9 – вхід від контролера (IN2);

– контакт 10 – вхід від контролера (IN3);

– контакт 11 – вхід від контролера (IN4);

– контакт 12 – перемикач для двигуна постійного струму №2. У разі використання крокового двигуна підключати сюди нічого не треба. Можна підключити до ШІМ-виходу для керування швидкістю двигуна постійного струму;

– контакт 13 – «+» двигуна постійного струму №2, або кроковий двигун B+;

– контакт 14 – «–» двигуна постійного струму №2, або кроковий двигун B–.

Даний модуль дає можливість керувати одним або двома двигунами постійного струму.

Якщо у проєкті використовується кілька двигунів, необхідно переконатися, що у них витримана однакова полярність під час підключення. Інакше, в процесі задання руху, наприклад, за годинниковою стрілкою, один з них обертатиметься у протилежному напрямку. З точки зору програмування Arduino це незручно.

Плюс джерела живлення підключають до четвертого піна L298N, мінус (GND) – до 5 піна. Якщо напруга джерела живлення менше 12 В, перемикач, позначений 3 на рисунку вище, можна залишити. При цьому буде можливість використовувати 5-вольтовий пін 6 з модуля для живлення Arduino [15].

Для прикладу виконаємо підключення двох двигунів до контролера, використовуючи драйвер, та створимо керуючу програму для управління рухом. Схема підключення показана на рисунку 2.3.

У нашому прикладі два двигуни постійного струму, так що цифрові піни D4, D5, D6 і D7 будуть підключені до пінів IN1, IN2, IN3 і IN4 відповідно. Після цього підключимо пін D19 до піну 7 на L298N (попередньо прибравши перемикач) і D3 до піну 12 (прибрати перемикач).

Напрямок обертання ротора двигуна керується сигналами HIGH або LOW на кожен привід (або канал). Наприклад, для першого мотора, HIGH на IN1 і LOW на IN2 забезпечить обертання в одному напрямку, а LOW і HIGH змусить обертатися в протилежний бік.

При цьому двигуни не обертатимуться, поки не буде сигналу HIGH на піні 7 для першого двигуна або на 12 піні для другого. Зупинити їх

обертання можна подачею сигналу LOW на ті самі, зазначені вище піни. Для керування швидкістю обертання використовується ШІМ-сигнал.

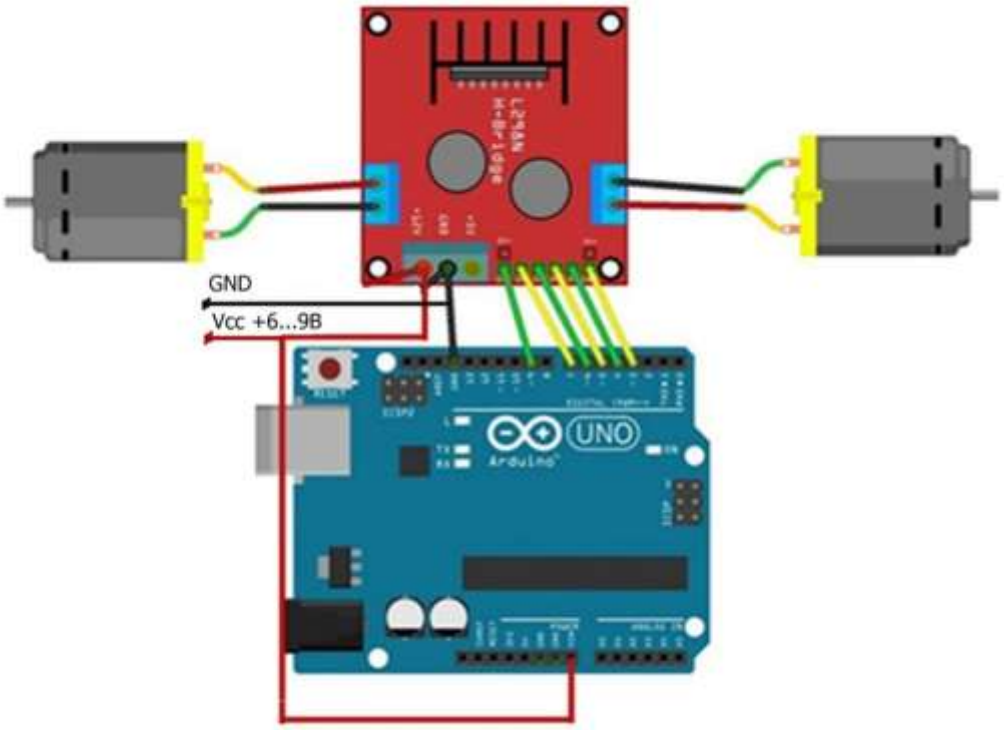


Рисунок 2.3 – Схема підключення двигунів постійного струму до Arduino

Для керування двигунами напишемо таку програму:

```
// перший двигун
int ena = 9;
int in1 = 7;
int in2 = 6;
// другий двигун
int enb = 3;
int in3 = 5;
int in4 = 4;
void setup()
{
// ініціалізуємо всі піни для керування двигунами як outputs
pinMode(ena, OUTPUT);
pinMode(enb, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
}
void One()
```

```
{
// ця функція забезпечить обертання двигунів в двох напрямках на
//встановленої швидкості
// запуск двигуна A
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
// встановлюємо швидкість 200 з доступного діапазону 0~255
analogWrite(ena, 200);
// запуск двигуна B
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
// встановлюємо швидкість 200 з доступного діапазону 0~255
analogWrite(enb, 200);
delay(1000);
// міняємо напрям обертання двигунів
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
delay(1000);
// вимикаємо двигуни
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}
void Two()
{
// ця функція забезпечує роботу двигунів у всьому діапазоні
можливих швидкостей
// зверніть увагу, що максимальна швидкість визначається самим
// двигуном і напругою живлення
// ШІМ-значення генеруються функцією analogWrite()
// і залежать від вашої плати керування
// запускають двигуни
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
// прискорення від нуля до максимального значення
for (int i = 0; i <256; i++)
{
analogWrite(ena, i);
```

```
analogWrite(enb, i);
delay(50);
}
// гальмування від максимального значення до мінімального
for (int i = 255; i>= 0; --i)
{
analogWrite(ena, i);
analogWrite(enb, i);
delay(50);
}
// тепер відключаємо мотори
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}
void loop()
{
One();
delay(1000);
Two();
delay(1000);
}
```

Скетч, наведений вище, спрацьовує у відповідності зі схемою підключення, що наведена на рисунку 1.31. Двигуни постійного струму і Arduino живляться від зовнішнього джерела живлення.

2.2 Управління кроковим двигуном

Крокові двигуни – це особливий вид двигунів, який дозволяє керувати обертанням ротора покроково. Це означає, що ротор може мати кілька стійких положень на один оберт. Коли положення ротора було задано, двигун прагне його зберегти незмінним до тих пір, поки не буде задано нове положення.

Даний тип двигунів застосовується для контрольованого й точного переміщення різних механізмів. Зокрема, застосовується в різних роботах, ЧПУ фрезерних, токарних верстатах тощо. Ще кроковий двигун володіє великою надійністю, тому що у нього крім підшипників немає елементів, які труться. Тому ресурс двигуна визначається тільки якістю підшипників. Завдяки своїй надійності даний тип двигунів широко застосовується в космонавтиці. На рисунку 2.4 показаний зовнішній вигляд крокового двигуна.

Рисунок 2.4 – Тестова плата з кроковим двигуном 28BYJ-48

Розглянемо двигун з маркуванням 28BYJ-48 [18]. Для здійснення повного оберту він робить 32 кроки. Двигун має 5 виводів: один загальний вивід, який підключається до плюса живлення і 4 керуючих виводів, які підключаються до мінуса живлення за допомогою керуючого пристрою. Керуючі виводи споживають значний струм, тому для них потрібна спеціальна мікросхема (ULN2003AN). Ця мікросхема називається драйвер крокового двигуна. Щоб змусити ротор двигуна обертатися, необхідно подавати сигнали на драйвер у певній послідовності. Драйвер у свою чергу їх посилить і передасть на двигун.

Керуючі послідовності сигналів формуються залежно від режиму роботи двигуна – крок або напівкрок. Як правило, використовуються тільки режими з дробленням кроку. У таблиці 2.2 подана послідовність для напівкрокового режиму роботи.

Таблиця 2.2 – Комбінації керуючих сигналів

Колір і номер дроту	Послідовність керуючих сигналів (напівкроки)							
	1	2	3	4	5	6	7	8
4 Помаранчевий	1	1	0	0	0	0	0	1
3 Жовтий	0	1	1	1	0	0	0	0
2 Рожевий	0	0	0	1	1	1	0	0
1 Синій	0	0	0	0	0	1	1	1

Для полегшення роботи з кроковим двигуном існує бібліотека Strepper.h. Вона містить такі функції:

– *Stepper (intsteps, int pin1, int pin2, int pin3, int pin4).* За допомогою цієї функції створюється змінна (об'єкт) типу Stepper, яка відповідатиме підключеному до Arduino крововому двигуну. У функції задаються номери портів, до яких підключений крововий двигун, і кількість кроків, необхідних

для повного оберту вала навколо своєї осі. Функцію необхідно ставити на початку програми перед функціями loop() і setup() у вигляді:

```
Stepper motorName = Stepper (steps, pin1, pin2, pin3, pin4),
```

або

```
Stepper motorName (steps, pin1, pin2, pin3, pin4);
```

Якщо значення steps невідомо, необхідно подивитися в документації двигуна, скільки градусів становить один крок. Потім необхідно розділити 360 градусів на цю величину. Зверніть увагу, що значення має бути ціле число;

– setSpeed (long rpms) – функція призначена для задання швидкості обертання двигуна в кількості обертів на хвилину. Аргумент – ціле позитивне число. Функція не приводить вал у рух, але задає його швидкість після виклику функції step();

– step (steps) – повертає вал двигуна на вказане число кроків. Швидкість визначається останнім викликом функції setSpeed(). Зверніть увагу, що ваша програма продовжить виконання, поки не закінчиться виконання функції step(). Тому потрібно намагатися задавати високу швидкість обертання і мінімальну кількість кроків. Позитивне число обертає двигун в одну сторону. Негативне – в іншу.

Нижче наведено приклад коду для керування кроковим двигуном із використанням бібліотеки Stepper.h:

```
#include <Stepper.h> //підключення бібліотеки для КД
Stepper myStepper (32, 8, 9, 10, 11); // оголошення КД
void setup()
{
/* задання швидкості обертання 32 оберта в хвилину. За хвилину зробить повний оберт*/
myStepper.setSpeed(32);
}
void loop()
{
/* зробити 32 кроки проти годинникової стрілки */
myStepper.step(-32);
}
```

2.3 Керування пристроями IoT за допомогою інфрачервоного зв'язку

Сьогодні дуже широко використовується дистанційне керування пристроями за допомогою інфрачервоного випромінювання. Таким чином

керуються кондиціонери, телевізори, музичні центри та інша техніка. Людське око не може бачити ІЧ-випромінювання, оскільки воно знаходиться вище видимого спектра, тобто має дуже велику довжину хвилі понад 700 нм (людина бачить кольори в діапазоні від 400 нм до 700 нм). Але ІЧ-випромінювання можна зробити видимим, наприклад, за допомогою звичайної фотокамери. Якщо навести камеру на ІЧ-пульт і натиснути на ньому кнопку, то на камері можна побачити світлодіодні мерехтіння.

Для організації найпростішого ІЧ-зв'язку необхідні ІЧ-приймач і ІЧ-випромінювач [19]. Приймач має три виводи, зліва направо: цифровий вихід, земля, живлення +5 В (рис. 2.5).

Рисунок 2.5 – Інфрачервоний датчик і пульт

На відміну від звичайного інфрачервоного фотодіода, ІЧ-приймач може приймати й обробляти інфрачервоний сигнал, що являє собою ІЧ-імпульси фіксованої частоти і певної тривалості – пачки імпульсів. Це технологічне рішення позбавляє від випадкових спрацьовувань, які можуть бути викликані фоновим випромінюванням і перешкодами з боку інших приладів, які випромінюють в інфрачервоному діапазоні.

Наприклад, значні перешкоди для приймача ІЧ-сигналів можуть створювати люмінесцентні освітлювальні лампи з електронним баластом. Зрозуміло, що використовувати ІЧ-приймач замість звичайного ІЧ-фотодіода не зручно, адже ІЧ-модуль є спеціалізованою мікросхемою, заточеною під певні потреби.

В основу кожного пульта покладено генератор імпульсів, що працює в частотному діапазоні між 30 і 40 кГц, сигнал якого промодульовано кодом тієї чи іншої команди. Для наочності розглянемо графік, зображений на рис. 2.6.

Як видно з рисунка 2.6, сигнал від пульта дистанційного керування має досить складну форму. На початку проходить пусковий імпульс, який запускає систему дешифрування у приймальному пристрої, далі відправляються коди

самого сигналу, які відповідають певній команді в цифровому вигляді, потім закривається стоп-імпульс, який зупиняє роботу дешифратора у приймачі.

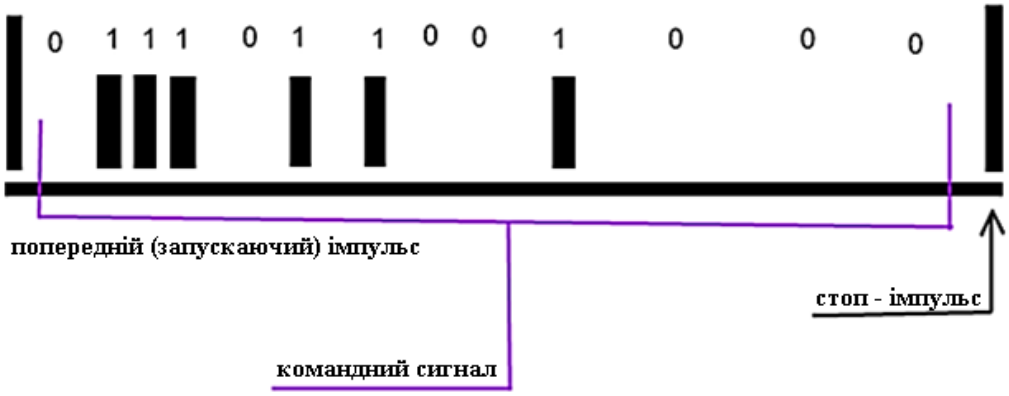


Рисунок 2.6 – Система кодувань пульта дистанційного керування

Дана система кодувань була вперше впровадження фірмою PHILIPS і отримала назву RC-5. Причому генератор у самому пульті вмикається лише після надходження команди (натискання будь-якої кнопки), без команди генератор не працює, перебуваючи в режимі очікування.

З метою захисту від помилкових спрацьовувань практично в усіх пультах передбачене також блокування генератора з одночасним натисканням більш ніж однієї кнопки [19].

Мікросхема приймача ІЧ-випромінювання включає (рис. 2.7):

– PIN-фотодіод;
– регульований підсилювач;
– смуговий фільтр;
– амплітудний детектор;
– інтегруючий фільтр;
– граничний пристрій;
– вихідний транзистор.

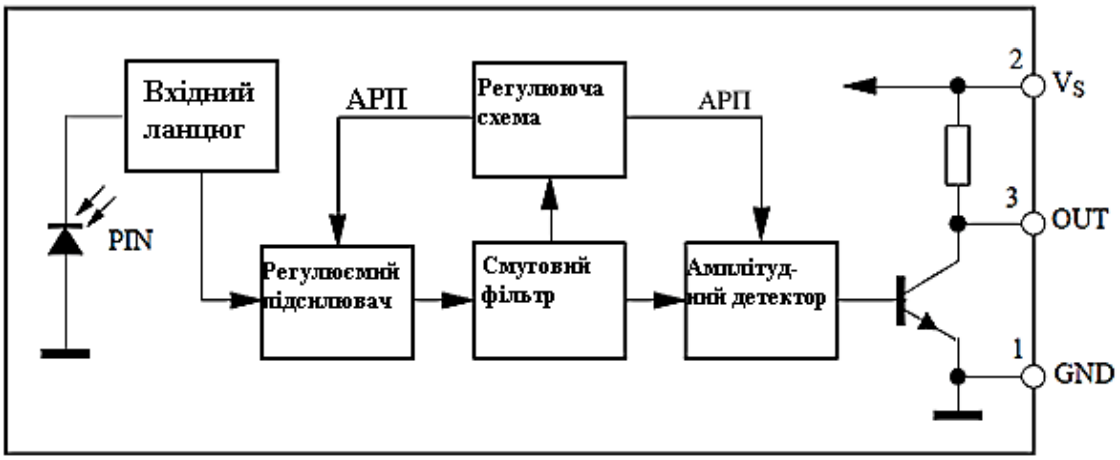


Рисунок 2.7 – Структурна схема ІЧ-модуля

PIN-фотодіод – це різновид фотодіода, у якого між областями *n* і *p* розташована область з власного напівпровідника (*i*-область). Область власного напівпровідника – це по суті прошарок з чистого напівпровідника без внесених до нього домішок. Саме цей шар і надає PIN-діоду його особливі властивості. До речі, PIN-діоди (не фотодіоди) активно застосовуються в СВЧ-електроніці. Погляньте на свій мобільний телефон, у ньому також використовується PIN-діод.

У звичайному стані струм через PIN-фотодіод не протікає, оскільки до схеми він підключений у зворотному напрямку (у зворотному зміщенні). Оскільки під дією зовнішнього інфрачервоного випромінювання в *i*-області виникають електронно-діркові пари, то в результаті через діод починає протікати струм. Цей струм потім перетвориться в напругу і надходитиме на регульований підсилювач.

Далі сигнал з регульованого підсилювача надходить на смуговий фільтр. Він слугує захистом від перешкод. Смуговий фільтр налаштований на певну частоту. Так в ІЧ-приймачах переважно використовуються смугові фільтри, налаштовані на частоту 30; 33; 36; 36,7; 38; 40; 56 і 455 кілогерц. Щоб сигнал, який випромінюється пультом ДК (дистанційного керування) міг бути прийнятий ІЧ-приймачем, він має бути модульованим такою самою частотою, на яку налаштований смуговий фільтр ІЧ-приймача. Ось так, наприклад, виглядає модульований сигнал від інфрачервоного діода, який випромінюється (рис. 2.8).

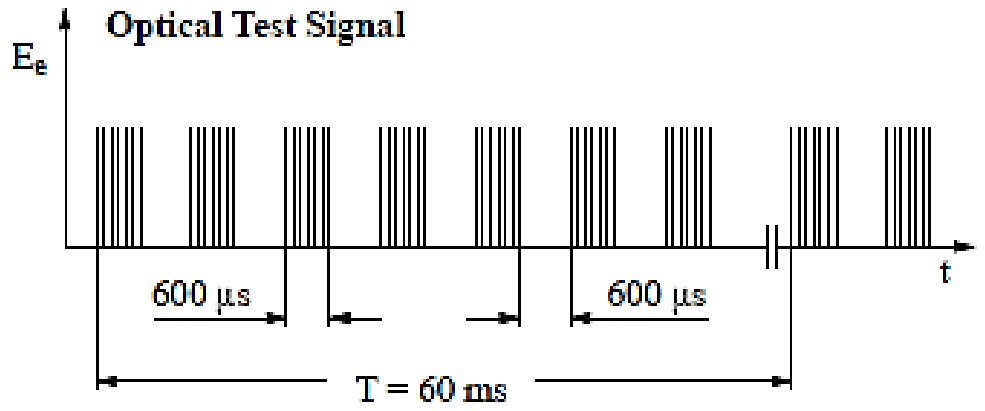


Рисунок 2.8 – Схема модульованого сигналу
від інфрачервоного діода, який випромінює

Сигнал на виході ІЧ-приймача зображений на рисунку 2.9.

Варто зазначити, що частота смугового фільтра невелика. Тому ІЧ-модуль з фільтром на 30 кілогерц цілком може приймати сигнал частотою 36,7 кілогерц і більше. Правда, при цьому відстань прийому помітно знижується.

Після того, як сигнал пройшов через смуговий фільтр, він надходить на амплітудний детектор та інтегруючий фільтр. Інтегруючий фільтр

необхідний для заглушення коротких одиночних сплесків сигналу, які можуть бути викликані перешкодами. Далі сигнал надходить на пороговий пристрій, а потім на вихідний транзистор.

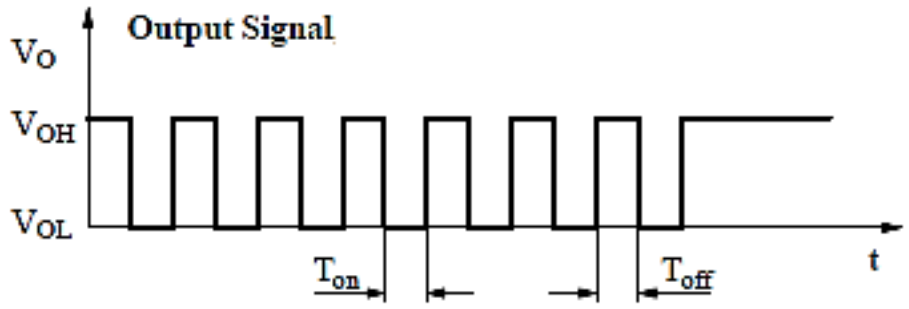


Рисунок 2.9 – Сигнал на виході ІЧ-приймача

Для стійкої роботи приймача коефіцієнт підсилення регульованого підсилювача контролюється системою автоматичного регулювання підсилення (АРП). Оскільки корисний сигнал являє собою пачку імпульсів певної тривалості, то через інерційності АРП сигнал встигає пройти через тракт підсилення та інші вузли схеми.

У випадку, якщо тривалість системи імпульсів велика, система АРП спрацьовує і приймач перестає приймати сигнал. Така ситуація може виникнути, коли ІЧ-приймач засвічений люмінесцентною лампою з електронним баластом, який працює на частотах 30–50 кілогерц. У такому випадку промодульоване інфрачервоне випромінювання парів ртуті лампи може пройти захисний смуговий фільтр фотоприймача і викликати спрацьовування АРП. Природно, при цьому чутливість ІЧ-приймача спадає.

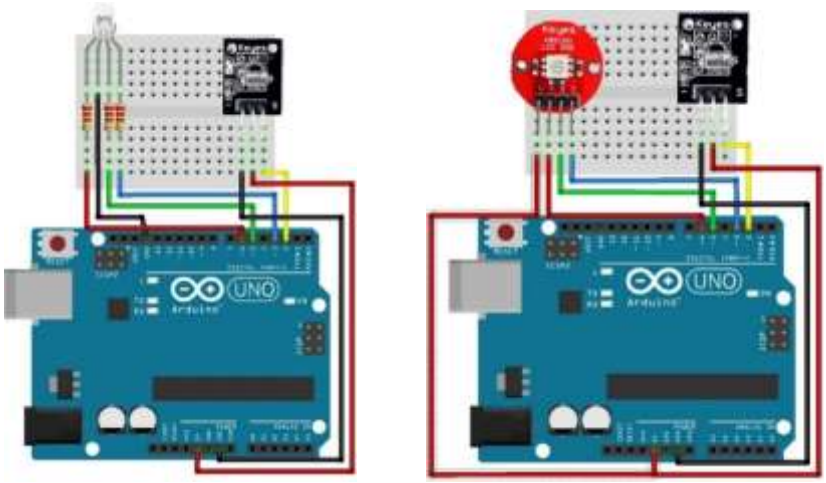
Автоматичне регулювання порога виконує аналогічну функцію, що і АРП, керуючи граничними значеннями реакції порогового пристрою. Регулювання виставляє рівень порога спрацьовування таким чином, щоб зменшити число помилкових імпульсів на виході модуля. За відсутності корисного сигналу число помилкових імпульсів може досягати 15-ти за хвилину.

Форма корпусу ІЧ-модуля сприяє фокусуванню випромінювання, що приймається на чутливу поверхню фотодіода. Матеріал же корпусу пропускає випромінювання з довжиною хвилі від 830 до 1100 нм. Таким чином, у пристрої реалізований оптичний фільтр. Для захисту елементів приймача від впливу зовнішніх електричних полів у модулі встановлено електростатичний екран. На фотографії показані ІЧ-модулі марки HS0038A2 і TSOP2236. Для порівняння поруч показані звичайні ІЧ-фотодіоди КДФ-111В і ФД-265 (рис. 2.10 і 2.11) [20].

Рисунок 2.10 – ІЧ-приймачі HS0038A2 і TSOP2236

Рисунок 2.11 – ІЧ-фотодіоди КДФ-111В і ФД-265

Розглянемо приклад використання ІЧ-модуля керування для обміну даних між двома платами Arduino. Як випромінювач використаємо ІЧ-світлодіод [20]. З натисканням на кнопку формуватимемо повідомлення, яке надійде на ІЧ-світлодіод і буде передано як ІЧ-сигнал у вигляді 1 байту даних. Для цього необхідно зібрати дві схеми (рис. 2.12).



а б

Рисунок 2.12 – Схеми ІЧ-системи обміну повідомленнями:
а) модуль передавання; б) модуль приймання даних

Розпіновка модуля випромінювання показана на рисунку 2.13.

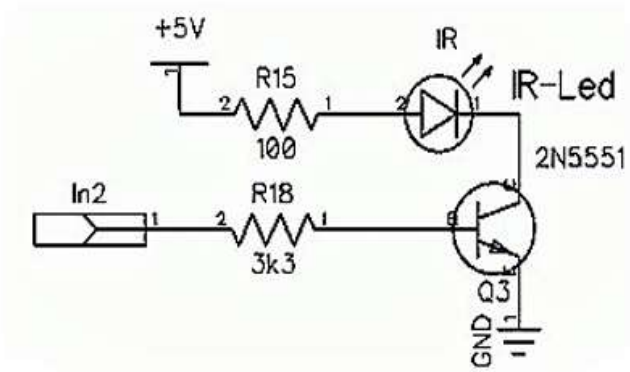


Рисунок 2.13 – Модуль ІЧ-випромінювання

Виходячи з наведеної схеми, в модулі приймача ІЧ-приймач під'єднаємо до 2-го цифрового порту, а RGB - світлодіод до 3, 5 і 6 аналогових виходів (рис. 1.40, б).

Далі напишемо програму, яка з натисканням кнопки передаватиме її номер на приймач. З отриманням номера вмикатимемо діод залежно від натиснутої кнопки. Програма для приймача матиме такий код:

```
#include <IRremote.h>  // бібліотека для роботи з ІЧ сенсорами

// розкоментуйте наступний рядок для загального анода
//#define COMMON_ANODE

int redPin = 9;
int greenPin = 6;
int bluePin = 5;
IRrecv irrecv(11);   // вказуємо пін (вихід), до якого підключений прийомник decode_results results;

void setup() {
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);
 Serial.begin(9600);  // виставляємо швидкість COM-порту
```

```
 irrecv.enableIRIn(); // запускаємо прийом
}
void loop() {
 if (irrecv.decode(&results)) {        // якщо дані прийшли то виконуємо умову
  Serial.println(results.value, DEC); // відправляємо отримані дані в консоль
  switch (results.value) {       // залежно від отриманого значення будемо виконувати ту, чи іншу умову
   case (1): {
     setColor(255, 0, 0);
     break;
    }
   case (2): {
     setColor(0, 255, 0);
     break;
    }
   case (3): {
     setColor(0, 0, 255);
     break;
    }
   case (4): {
     setColor(0, 0, 0);
     break;
    }
  } // приймаємо наступну команду
  irrecv.resume();
 }
}
// функція яка задає значення діода.
void setColor(int red, int green, int blue) {
#ifdef COMMON_ANODE
 red = 255 - red;
 green = 255 - green;
 blue = 255 - blue;
#endif
 analogWrite(redPin, red);
 analogWrite(greenPin, green);
 analogWrite(bluePin, blue);
}
```

Програма для передавача матиме такий код:

```
#include <IRremote.h>
IRsend irsend;
const int digital = 2;
int i = 0;
void setup()
{ // вказуємо що на 2 цифровий вхід буде підключена кнопка
 // задаємо вхід як вхідний.
 // Діод підключається за замовчуванням до 3-ої ніжки.
pinMode(digital, INPUT);
 }
void loop() {
  if( digitalRead(digital) == HIGH)
  {
   irsend.sendRC5(i, 8); // Передаємо 8 біт інформації
   i++;
   delay(2000);
  }
  else
  {
  irsend.sendRC5(i, 8);
delay(10);
  if(i == 4)
  i=0;  }   }
```

Розглянемо наступний приклад, передачу текстового повідомлення через ІЧ-канал і виведення повідомлення на монітор.

Код ІЧ-приймача реалізовано так:

```
#include «IRremote.h»
// Вказуємо пін (вихід), до якого підключений приймач
IRrecv irrecv(11);
decode_results results;
int redPin = 9;
int greenPin = 6;
int bluePin = 5;
void setup()
{pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);
Serial.begin(9600);
// виставляємо швидкість COM – порту
```

```
 irrecv.enableIRIn();
// Запускаємо прийом
}
void loop() {
  // Якщо дані прийшли то виконуємо умову
 if (irrecv.decode(&results))
 {
// Відправляємо отримані дані в консоль
  Serial.write(results.value);
// В залежності від отриманого значення будемо виконувати ту, чи іншу умову.
switch(results.value)
  {
   case('r'):
   {
    setColor(255,0,0);
    break;
   }
   case('g'):
   {
    setColor(0,255,0);
    break;
  }
   case('b'):
   {
    setColor(0,0,255);
    break;
   }
   case('c'):
   {
    setColor(0,0,0);
    break;
   }
   default:
   {
    Serial.write(results.value);
   }
}
// Приймаємо наступну команду
irrecv.resume();
}}
// Функція яка задає значення діода.
void setColor(int red, int green, int blue)
```

```
{#ifdef COMMON_ANODE
red = 255 - red;
green = 255 - green;
blue = 255 - blue;
#endif
analogWrite(redPin, red);
analogWrite(greenPin, green);
analogWrite(bluePin, blue);
}
```

Код ІЧ-передавача реалізовано так:

```
#include <IRremote.h>
IRsend irsend;
int i = 0;
void setup()
{
Serial.begin(9600);
}
void loop() {
  if (Serial.available()> 0) {
  i = Serial.read();
   irsend.sendRC5(i, 8);
delay(1000);}}
```

Для того, щоб перевірити наведений код, можна скористатися схемою на рисунку 1.40.

2.4 Керування мобільним роботом за допомогою ІЧ-пульта

Розглянемо наступний приклад – керування мобільним роботом з а допомогою ІЧ-пульта [20].

Інфрачервоний пульт (рис. 2.14) дистанційного керування – один з найпростіших способів взаємодії з електронними приладами. Так, практично в кожному будинку є кілька даних пристроїв: телевізор, музичний центр, відеоплеєр, кондиціонер. Але найцікавіше місце, де застосовується інфрачервоний пульт – дистанційне керування роботом. На нашому прикладі ми спробуємо реалізувати такий спосіб керування за допомогою Arduino UNO.

Для прийому сигналу з пульта потрібен ІЧ-датчик.

Ми можемо детектувати інфрачервоне випромінювання звичайним фотодіодом/фототранзистором, але на відміну від нього, ІЧ-датчик сприймає

інфрачервоний сигнал тільки на частоті 38 кГц (іноді 40кГц). Ця властивість дозволяє датчику ігнорувати багато інших сторонніх світлових шумів від ламп освітлення і сонця.

Рисунок 2.14 – ІЧ-пульт

Для даного прикладу скористаємося ІЧ-датчиком VS1838B, який має такі характеристики:

– частота: 38 кГц;

– напруга живлення: 2,7 - 5,5 В;

– споживаний струм: 50 мкА.

Можна також використовувати інші датчики, наприклад: TSOP4838 [21], TSOP1736, SFH506. Датчик має три виводи. Якщо подивитися на нього з боку приймача ІЧ-сигналу, як показано на рисунку 2.15:

– зліва буде – вихід на контролер;

– по центру – від’ємний контакт живлення (земля);

– справа – додатній контакт живлення (2.7–5.5 В).

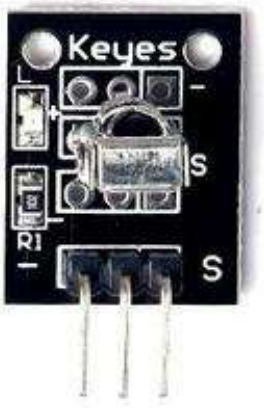


Рисунок 2.15 – ІЧ-датчик TSOP4838

Схема підключення двигунів і драйвера залишається така сама, як у підрозділі 2.2, додається тільки ІЧ-приймач, який підключається до 11 входу на Arduino (рис. 2.16).

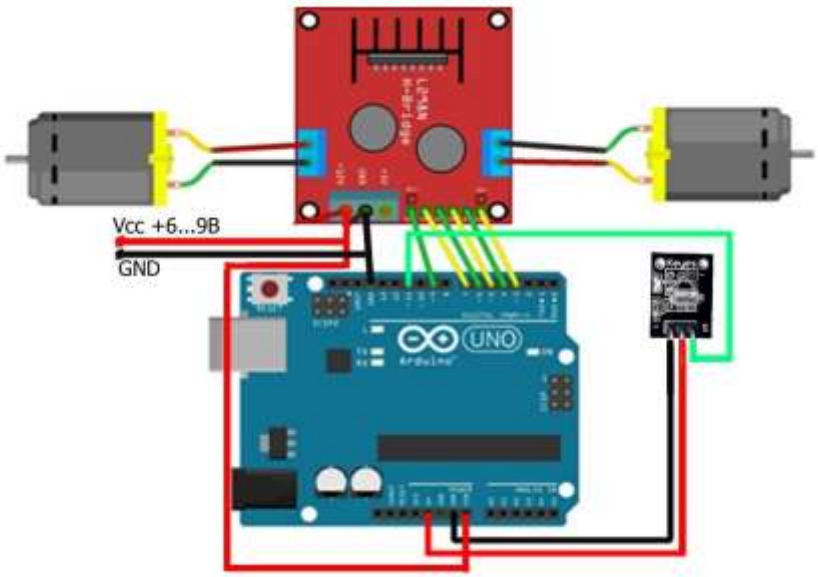


Рисунок 2.16 – Схема керування двигунами постійного струму
за допомогою ІЧ-пульта керування

Двигуни постійного струму і Arduino живляться від зовнішнього джерела живлення. Код управління двигунами має такий вигляд:

```
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
// перший двигун
int ena = 9;
int in1 = 6;
int in2 = 7;
// другий двигун
int enb = 3;
int in3 = 5;
int in4 = 4;
void runForward() // їхати вперед
{
  digitalWrite(in1, LOW);
  digitalWrite(in4, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
}
void steerRight() // поворот на право
```

```
{
  digitalWrite(in1, HIGH);
  digitalWrite(in4, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in2, LOW);
}
void steerLeft() // поворот наліво
{
  digitalWrite(in2, HIGH);
  digitalWrite(in1, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void stepBack() // їхати назад
{
  digitalWrite(in1, HIGH);
  digitalWrite(in4, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
}
void STOP() //стоп
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
  analogWrite(ena,255);
  analogWrite(enb,255);
     }
void loop()
{
  if (irrecv.decode(&results))
  {Serial.println(results.value);
  delay(16);
```

```
    switch (results.value)
    {
      case 16736925: // КОД_ВПЕРЕД
        runForward();
        break;
      case 16754775: // КОД_НАЗАД
        stepBack();
        break;
      case 16720605: // КОД_ВЛІВО
        steerLeft();
        break;
      case 16761405: // КОД_ВПРАВО
        steerRight();
        break;
      case 16712445: // КОД_СТОП
        STOP();
break;      }
    delay(16);
irrecv.resume();
  }}
```

У цьому прикладі використовується стандартна бібліотека Arduino-IRremote, яку можна знайти за адресою: https://github.com/z3t0/Arduino-IRremote

2.5 Реалізація руху мобільного робота за лінією

Завдання проходження маршруту є одним з найбільш поширених завдань у робототехніці. Його поширеність пов'язана з простотою і з великою кількістю способів реалізації [22].

Необхідними для робота елементами є датчики виявлення лінії. Вони призначені для того, щоб робот міг їздити заданою траєкторією. Як датчики лінії використовуватимуться відбиваючі оптопари (рис. 2.17).

Рисунок 2.17 – Відбиваюча оптопара

Кожна оптопара складається з випромінювача (лампочки або світлодіода) і приймача (фоторезистора, фототранзистора або фотодіода). В основі їх роботи лежить здатність поверхні відбивати частину світла, яке на неї падає.

Відбите світло потрапляє на приймач, і тоді з кількості відбитого світла можна судити про наявність або відсутність предмета перед оптопарою. Це проілюстровано на рисунку 2.18. Крім того, світло по-різному відбивається від поверхонь різного кольору. Тому за однакової відстані до поверхні оптопара може спрацювати, якщо поверхня білого кольору, і може не спрацювати, якщо поверхня чорного кольору.

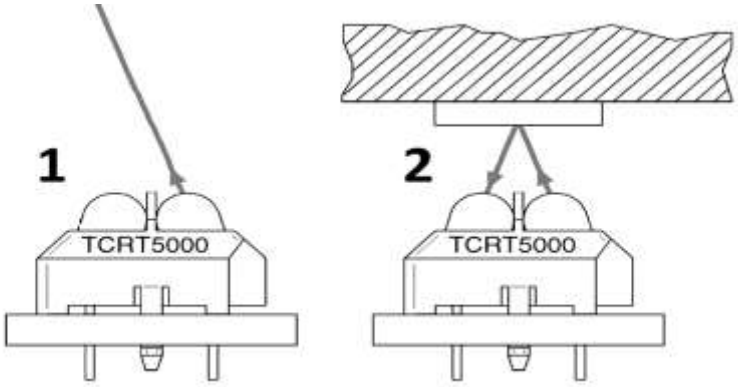


Рисунок 2.18 – Принцип дії оптопари

Принцип дії оптопари:

1) світло випромінюється світлодіодом, але не відбивається і не потрапляє на фотоприймач;

2) відбите світло надходить на фотоприймач.

Для реалізації задачі слідування за лінією зберемо схему пристрою. Схема підключення двигунів і драйвера залишається така сама, як на попередньому експерименті, додається тільки 2 датчики лінії, які ми підключаємо до D2 і D8 портів Arduino (рис. 2.19).

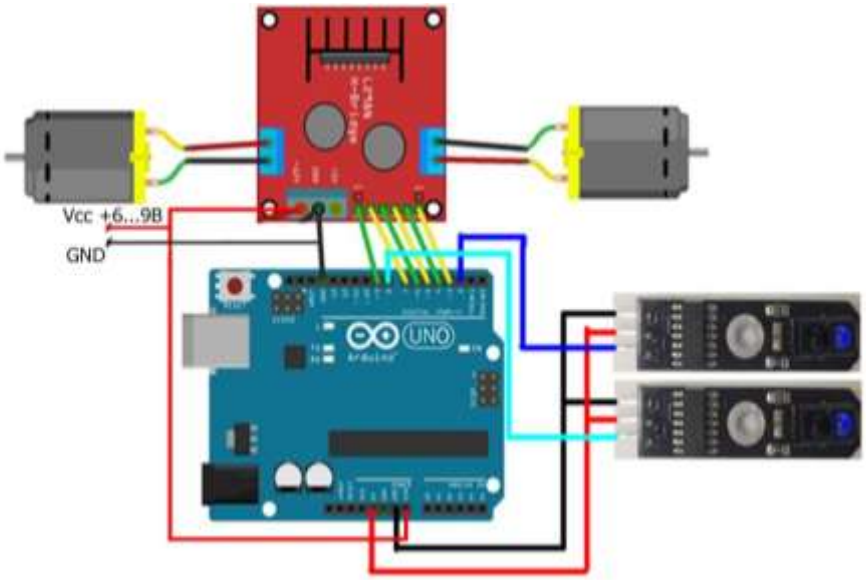


Рисунок 2.19 – Схема управління роботом для слідування за лінією

Двигуни постійного струму та Arduino живляться від зовнішнього джерела живлення.

Скетч, наведений нижче, спрацьовує відповідно до схеми підключення, яку ми розглядали вище:

```
// перший двигун
int ena = 9;
int in1 = 6;
int in2 = 7;
// другий двигун
int enb = 3;
int in3 = 5;
int in4 = 4;
// датчики лінії
int LS = 2;
int RS = 8;
void setup()
{
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
  pinMode(LS, INPUT);
  pinMode(RS, INPUT);
}
void runForward() // їхати вперед
{
  digitalWrite(in1, LOW);
  digitalWrite(in4, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
}
void steerRight() // поворот направо
{
  digitalWrite(in1, HIGH);
  digitalWrite(in4, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in2, LOW);
}
void steerLeft() // поворот наліво
{
```

```
  digitalWrite(in2, HIGH);
  digitalWrite(in1, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void stepBack() // їхати назад
{
  digitalWrite(in1, HIGH);
  digitalWrite(in4, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
}
void STOP() //стоп
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void loop()
{
if((digitalRead(LS)==LOW)  &&  (digitalRead(RS)==LOW))        //  рух
вперед
{
runForward();
  analogWrite(ena,150);
  analogWrite(enb,150);
}
if((digitalRead(LS)==LOW) && (digitalRead(RS)==HIGH))    // поворот
направо
{
steerRight();
  analogWrite(ena,100);
  analogWrite(enb,100);
}
if((digitalRead(LS)==HIGH)  &&  (digitalRead(RS)==LOW))          //
поворот вліво
{
steerLeft();
  analogWrite(ena,100);
  analogWrite(enb,100);
}
```

```
      if((digitalRead(LS)==HIGH) && (digitalRead(RS)==HIGH))      // пошук
лінії
      {
         runForward();
       analogWrite(ena,100);
      analogWrite(enb,100);
      }}
```

2.6 Вивчення основ роботи з датчиком відстані

Для визначення дистанції до об’єкта в робототехніці використовуються спеціальні пристрої – далекоміри [23]. Вони відрізняються за принципом вимірювання відстані, а також за способом подання інформації про відстані.

За способом вимірювання відстані є лазерні, світлові та звукові, а за способом подання інформації про відстані є далекоміри, за допомогою яких можна безпосередньо виміряти відстань або ж тільки визначити, чи є об'єкт у межах якоїсь відстані.

На рисунку 2.20 наведено приклад світлового далекоміра з дискретним виходом, тобто ми будемо знати, чи є перешкода в межах досяжності робота.

Рисунок 2.20 – Світловий далекомір з дискретним виходом

Світловий далекомір з дискретним виходом дозволяє виявити об'єкт, не використовуючи механічний контакт з ним, що дає можливість визначити присутність об'єкта на деякій відстані. За допомогою нього можна встановити відсутність або наявність перешкоди на заданій відстані або ближче. Визначити точну відстань до перешкоди не можна.

Інше застосування модуля – лічильник обертів або вимірювач швидкості обертання. У разі циклічного лінійного переміщення модуль далекоміра застосовується для визначення швидкості та положення рухомої деталі. Датчик реагує на відображення ІЧ-випромінювання в контрольованій зоні.

Для вимірювання швидкості обертання на рухливий елемент конструкції наносять широкі білі й чорні смуги. Випромінювання датчика направляють на смуги, що чергуються. Під час обертання з виходу модуля надходитимуть імпульси, частота яких говорить про швидкість обертання.

У модуль далекоміра входять два напівпровідникових фотоприлади: світлодіод інфрачервоного випромінювання і фотоприймач. Випромінювання світлодіода відбивається від перешкоди і надходить на фотоприймач. Випромінювання датчика модульовано частотою 38 кГц для роботи фотоприймача. Для цього до складу модуля входить генератор, виконаний на поширеній мікросхемі таймер 555.

Характеристики далекоміра:

– живлення: напруга 3–5,5 В;

– дистанція виявлення білої перепони 2–40 см;

– частота приймача 38 кГц;

– ефективний кут огляду 35°;

– робоча температура від -10 до +50 °С.

Інший тип далекоміра – ультразвуковий. Даний пристрій надає можливість вимірювати відстань до перешкоди у сантиметрах.

Згідно з документацією, відстань до перешкоди можна отримати за такою формулою:

Відстань до перешкоди (см) = довжина імпульсу (мкс)/58.

Ультразвуковий далекомір зображено на рисунку 2.21.

Рисунок 2.21 – Ультразвуковий далекомір

Принцип роботи далекоміра простий: з передавача надсилаються 8 імпульсів частотою 40 КГц, які відбиваються від перешкоди і приймаються ультразвуковим приймачем [24]. Дистанція виходить за такою формулою:

Ширина імпульсу (у мікросекундах) / 58 = Дистанція (у сантиметрах).

Схема підключення двигунів і драйвера показана на рисунку 2.22. До попередньої схеми додається тільки ультразвуковий датчик об'єкта, який підключений до D10 і D11 пінів Arduino.

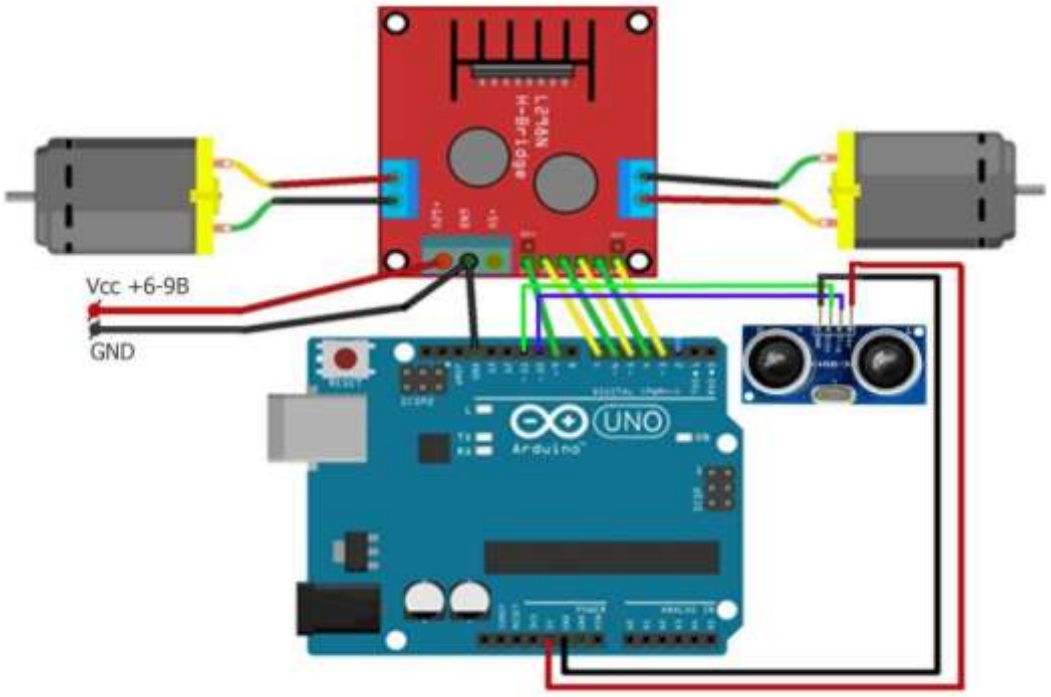


Рисунок 2.22 – Схема модуля управління роботом
на основі датчика відстані

Скетч, наведений нижче, керує роботом, який шукає об'єкти та їде до них:

```
// перший двигун
int ena = 9;
int in1 = 6;
int in2 = 7;
// другий двигун
int enb = 3;
int in3 = 5;
int in4 = 4;
// Піни, які використовуються ультразвуковим далекоміром
const int Trig = 10;
const int Echo = 11;
// Змінні, для зберігання даних з далекоміра
unsigned int time_us=0;
unsigned int distance_sm=0;
```

```
void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
 }
void runForward() // Їхати вперед
{
  digitalWrite(in1, LOW);
  digitalWrite(in4, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  analogWrite(ena,250);
 analogWrite(enb,250);
}
void steerLeft() // Поворот на ліво
{
  digitalWrite(in2, HIGH);
  digitalWrite(in1, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(ena,100);
 analogWrite(enb,100);
}
int HC_SR04() // Функція визначення відстані з далекоміра
{
  digitalWrite(Trig, HIGH); // Подаємо сигнал на вихід
мікроконтролера
  delayMicroseconds(10); // Утримуємо 10 мікросекунд
  digitalWrite(Trig, LOW); // Потім прибираємо
time_us=pulseIn(Echo, HIGH); // Заміряємо довжину імпульсу
distance_sm=time_us/58; // Перераховуємо в сантиметри
return distance_sm; // Повертаємо значення
}
void loop()
{
  Serial.println(HC_SR04());
  while (HC_SR04()<50)
```

```
{
      runForward();
   }
  steerLeft();
}
```

2.7 Застосування акселерометра у робототехніці

Гіроскоп являє собою пристрій, який реагує на зміну кутів орієнтації контрольованого тіла. Акселерометр – це пристрій, який вимірює проєкцію удаваного прискорення, тобто різниці між істинним прискоренням об'єкта та гравітаційним прискоренням. Таким чином, гіроскоп реагує на зміну в просторі незалежно від напрямку руху, а за допомогою акселерометра може вимірювати лінійні прискорення предмета та штучно розрахувати розташування предмета у просторі.

Триосьовий гіроскоп з триосьовим акселерометром MPU-6050 [25] (рис. 2.23) застосовується для визначення положення в просторі, у системах стабілізації положення, стабілізації прямолінійного руху і руху за заданою кривою. Наприклад, у балансирних роботах, в ігрових приставках, застосовується у робототехніці, для вимірювання кутів нахилу, швидкості обертання, в авіамоделюванні, його застосовують в автопілоті. Датчик може застосовуватися для вимірювання перевантажень тощо.

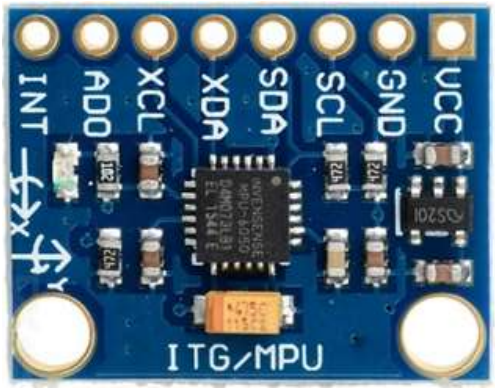


Рисунок 2.23 – Акселерометр MPU6050

Характеристики MPU6050:
– напруга живлення 2,375–3,46 В;
– споживаний струм до 4 мА;
– інтерфейс передачі даних – I2C;
– максимальна швидкість I2C–400 кГц;

– вхід для інших датчиків I2C;

– внутрішній генератор на 8 МГц (поза модулем можливість підключити зовнішній кварцовий резонатор на 33,768 кГц або 19,2 МГц).

Функції MPU6050:

– триосьовий MEMS гіроскоп з 16-бітовим АЦП;

– триосьовий MEMS акселерометр з 16-бітовим АЦП;

– Digital Motion Processor (DMP);

– slave I2C для підключення до мікроконтролера;

– master I2C для підключення до мікросхеми додаткового датчика;

– регістри даних датчиків;

– температурний сенсор;

– самоперевірка гіроскопа і акселерометра.

У цьому пристрої гіроскоп використовують у парі з акселерометром, тому що дані від обох датчиків доповнюють і коректують один одного.

Дані вимірювань датчиків можна зчитувати як з регістрів зберігання, так і користуватися функціями FIFO мікросхеми MPU-6050.

Мікросхема MPU-6050 містить Digital Motion Processor (DMP), він необхідний для того, щоб обробляти дані, які отримані з датчиків гіроскопа і акселерометра. Все це робиться для того, щоб підвищити точність одержуваних даних.

За допомогою MPU-6050 можна визначати положення об'єкта у просторі, а також рух об'єкта або його зіткнення. Наприклад, діагностувати падіння об'єкта або поштовх об перешкоду, щоб обходити її.

Область застосування досить широка. Даний модуль можна використовувати для координації різних пристроїв – від простого детектора руху до системи орієнтації різних роботів або керування рухами будь-яким пристроєм. Крім того, його часто застосовують для стабілізації польоту моделей літальних апаратів, що можливо через спільне використання гіроскопа і акселерометра.

Розглянемо приклад вихідного коду для отримання даних з акселерометра:

```
#include "Wire.h" // Ця бібліотека дозволяє спілкуватися з пристроями I2C
const int MPU_ADDR=0x68; // I2C адреса датчика
int16_t accelerometer_x, accelerometer_y, accelerometer_z; // змінні для даних акселерометра
int16_t gyro_x, gyro_y, gyro_z; // змінні для даних гіроскопа
int16_t temperature; // змінні для даних температури
```

```
void setup() {
  Serial.begin(9600);

  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Починає передачу в підлеглий пристрій I2C (плата GY-521)
  Wire.write(0x6B); // PWR_MGMT_1
  Wire.write(0); //
  Wire.endTransmission(true);
}
void loop() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // починаючи з регістра 0x3B
  Wire.endTransmission(false); // параметр вказує, що Arduino відправить перезавантаження. В результаті з'єднання залишається активним
  Wire.requestFrom(MPU_ADDR, 7*2, true); // запросить в цілому 7*2=14 регістрів

  // "Wire.read()<<8 | Wire.read();"означає, що два регістра зчитуються і зберігаються в одній змінній

  accelerometer_x = Wire.read()<<8 | Wire.read();
  accelerometer_y = Wire.read()<<8 | Wire.read();
  accelerometer_z = Wire.read()<<8 | Wire.read();
  temperature = Wire.read()<<8 | Wire.read();
  gyro_x = Wire.read()<<8 | Wire.read();
  gyro_y = Wire.read()<<8 | Wire.read();
  gyro_z = Wire.read()<<8 | Wire.read();

  // виведення даних
  Serial.print("aX = "); Serial.print(accelerometer_x);
  Serial.print(" | aY = "); Serial.print(accelerometer_y);
  Serial.print(" | aZ = "); Serial.print(accelerometer_z);
  // перетворення показників датчика для температури
  Serial.print(" | tmp = "); Serial.print(temperature/340.00+36.53);
  Serial.print(" | gX = "); Serial.print(gyro_x);
  Serial.print(" | gY = "); Serial.print(gyro_y);
  Serial.print(" | gZ = "); Serial.print(gyro_z);
  Serial.println();

  delay(500);
}
```

Глобальні змінні бібліотеки для роботи з акселерометром:

– int accel_x_OC – містить вимірювання положення акселерометра відносно осі x під час калібрування;

– int accel_y_OC – містить вимірювання положення акселерометра відносно осі y під час калібрування;

– int accel_z_OC – містить вимірювання положення акселерометра відносно осі z під час калібрування;

– int gyro_x_OC – містить вимірювання положення гіроскопа відносно осі x;

– int gyro_y_OC – містить вимірювання положення гіроскопа відносно осі y;

– int gyro_z_OC – містить вимірювання положення гіроскопа відносно осі z;

– float temp_scalled – містить абсолютне значення температури у градусах Цельсія;

– float accel_x_scalled – дані осі x акселерометра мінус дані калібрування;

– float accel_y_scalled – дані осі y акселерометра мінус дані калібрування;

– float accel_z_scalled – дані осі z акселерометра мінус дані калібрування;

– float gyro_x_scalled – дані гіроскопа відносно осі x мінус дані калібрування;

– float gyro_y_scalled – дані гіроскопа відносно осі y мінус дані калібрування;

– float gyro_z_scalled – дані гіроскопа відносно осі z мінус дані калібрування;

Функції в програмі Arduino для роботи з mpu6050:

– MPU6050_ReadData() – ця функція зчитує дані з акселерометра, гіроскопа і датчика температури. Після зчитування даних значення змінних (temp_scalled, accel_x_scalled, accel_y_scalled, accel_z_scalled, gyro_x_scalled, gyro_y_scalled and gyro_z_scalled) оновлюються;

– MPU6050_ResetWake() – ця функція скидає налаштування чіпа на значення за замовчуванням. Рекомендується використовувати скидання налаштувань перед налаштуванням чіпа на виконання певного завдання;

– MPU6050_SetDLPF(int BW) – ця функція налаштовує вбудований фільтр низьких частот. Змінна int BW має містити значення (0–6). Пропускна спроможність фільтра змінюватиметься відповідно до поданої нижче таблиці 2.3.

Таблиця 2.3 – Пропускна спроможність фільтра низьких частот

int BW	Пропускна спроможність фільтра
0 або Any	нескінченність
1	184
2	94
3	44
4	21
5	10
6	5

Якщо int BW не в діапазоні 0–6, фільтр низьких частот відключається, що відповідає установці – нескінченність;

– MPU6050_SetGains(int gyro, int accel) – ця функція використовується для установки максимального значення шкали вимірювань (табл. 2.4):

Таблиця 2.4 – Максимальне значення шкали вимірювань

int gyro	Макс. знач.[угол/с]	int accel	Макс. знач. [м/с^2]
0	250	0	2g
1	500	1	4g
2	1000	2	8g
3	2000	3	16g

– програмування акселерометра робота в середовищі програмування Arduino IDE;

– MPU6050_ReadData() – ця функція використовує масштабні коефіцієнти для розрахунку результату. Якщо не використовуються значення (0–3), MPU6050_ReadData() відобразити необроблені значення з датчика з похибкою калібрування. Для отримання оброблених значень встановіть змінні для калібрування (accel_x_OC, accel_y_OC, accel_z_OC, gyro_x_OC, gyro_y_OC and gyro_z_OC) в нуль;

– MPU6050_OffsetCal() – ця функція дозволяє відкалібрувати акселерометр і гіроскоп. Розраховані значення записуються у змінні accel_x_OC, accel_y_OC, accel_z_OC, gyro_x_OC, gyro_y_OC і gyro_z_OC для подальшої корекції. Для проведення калібрування необхідно розташувати осі x і y axes плати MPU6050 в горизонтальній площині, а вісь z – перпендикулярно до основи. Навіть незначні переміщення плати під час калібрування знижують точність розрахунку базової точки. Вісь z калібрується щодо сили земного тяжіння – 9.81 м / с2 (1g), що враховано в коді.

Результат виконання програми показано на рисунку 2.24.

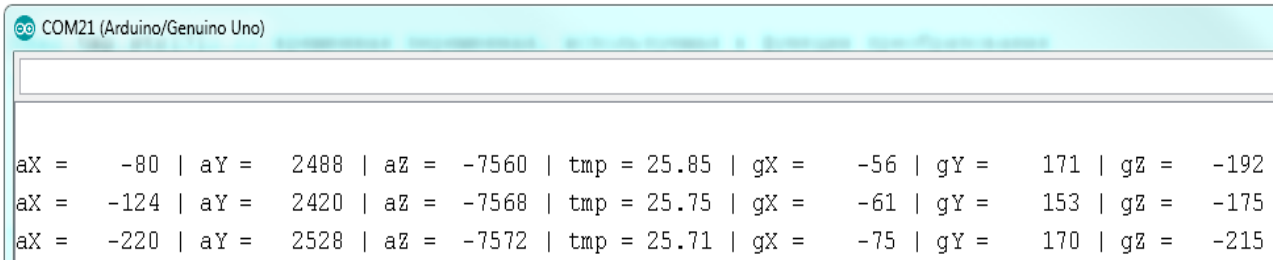


Рисунок 2.24 – Результат виконання програми тестування акселерометра

Наступним кроком розглянемо код на C++ для керування роботом балансиром:

```
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
 #include "Wire.h"
#endif

#define MIN_ABS_SPEED 30 //30

MPU6050 mpu;

// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

//PID
double originalSetpoint = 96.50; //186.50
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;

//adjust these values to fit your own design
double Kp = 70.45;   //10
double Kd = 4.4; //0.18
double Ki = 600; //25
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

double motorSpeedFactorLeft = 0.6;
double motorSpeedFactorRight = 0.5;
```

```
      //MOTOR CONTROLLER
      int ENA = 9;
      int IN1 = 7;
      int IN2 = 6;
      int IN3 = 5;
      int IN4 = 4;
      int ENB = 3;
      LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4,
motorSpeedFactorLeft, motorSpeedFactorRight);

      volatile bool mpuInterrupt = false; // indicates whether MPU
interrupt pin has gone high
      void dmpDataReady()
      {
       mpuInterrupt = true;
      }
      void setup()
      {
       // join I2C bus (I2Cdev library doesn't do this automatically)
       #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
       Wire.begin();
       TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
       #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
       Fastwire::setup(400, true);
       #endif
        Serial.begin(9600);
       mpu.initialize();
       devStatus = mpu.dmpInitialize();
       // supply your own gyro offsets here, scaled for min sensitivity
       mpu.setXGyroOffset(220); //
      mpu.setYGyroOffset(76);  // 76пе
       mpu.setZGyroOffset(-85 ) ; // -85
       mpu.setZAccelOffset(1688);  //1788  1688  factory  default  for  my
test chip
       // make sure it worked (returns 0 if so)
       if (devStatus == 0)
       {
       // turn on the DMP, now that it's ready
       mpu.setDMPEnabled(true);
       // enable Arduino interrupt detection
       attachInterrupt(0, dmpDataReady, RISING);
       mpuIntStatus = mpu.getIntStatus();
```

```
      // set our DMP Ready flag so the main loop() function knows it's
okay to use it
      dmpReady = true;
      // get expected DMP packet size for later comparison
      packetSize = mpu.dmpGetFIFOPacketSize();

      //setup PID
      pid.SetMode(AUTOMATIC);
      pid.SetSampleTime(10);
      pid.SetOutputLimits(-225, 225);
      }
      else
      {
      // ERROR!
      // 1 = initial memory load failed
      // 2 = DMP configuration updates failed
      // (if it's going to break, usually the code will be 1)
      Serial.print(F("DMP Initialization failed (code "));
      Serial.print(devStatus);
      Serial.println(F(")"));
      }
     }
     void loop()
     {
      // if programming failed, don't try to do anything
      if (!dmpReady) return;

      // wait for MPU interrupt or extra packet(s) available
      while (!mpuInterrupt && fifoCount < packetSize)
      {
      //no mpu data - performing PID calculations and output to motors
      pid.Compute();
      motorController.move(output, MIN_ABS_SPEED);

      }

      // reset interrupt flag and get INT_STATUS byte
      mpuInterrupt = false;
      mpuIntStatus = mpu.getIntStatus();
      // get current FIFO count
      fifoCount = mpu.getFIFOCount();

      // check for overflow (this should never happen unless our code is
too inefficient)
```

```
      if ((mpuIntStatus & 0x10) || fifoCount == 1024)
      {
      // reset so we can continue cleanly
      mpu.resetFIFO();
      Serial.println(F("FIFO overflow!"));

      // otherwise, check for DMP data ready interrupt (this should
happen frequently)
      }
      else if (mpuIntStatus & 0x02)
      {
      // wait for correct available data length, should be a VERY short wait
      while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
      // read a packet from FIFO
      mpu.getFIFOBytes(fifoBuffer, packetSize);

      // track FIFO count here in case there is > 1 packet available
      // (this lets us immediately read more without waiting for an
interrupt)
      fifoCount -= packetSize;

      mpu.dmpGetQuaternion(&q, fifoBuffer);
      mpu.dmpGetGravity(&gravity, &q);
      mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
      input = ypr[1] * 180/M_PI + 180;
      }
     }
```

У програмі використовуються бібліотеки для зв'язку акселерометра з Arduino за I2C інтерфейсом.

Для перевірки роботоспроможності наведеного коду необхідно зібрати схему, наведену на рисунку 2.25.

Виходячи із схеми, необхідно підключити цифрові піни Arduino до драйвера. У нашому прикладі два двигуни постійного струму, так що цифрові піни D4, D5, D6 і D7 будуть підключені до пінів IN1, IN2, IN3 і IN4 відповідно. Після цього підключіть пін D19 до піну 7 на L298N (попередньо забравши конектор) і D3 до піну 12 (знов-таки, прибравши конектор).

Напрямок обертання ротора двигуна керується сигналами HIGH або LOW на кожен привід (або канал). Наприклад, для першого мотора, HIGH на IN1 і LOW на IN2 забезпечить обертання в одному напрямку, а LOW і HIGH змусить обертатися у протилежний бік [25].

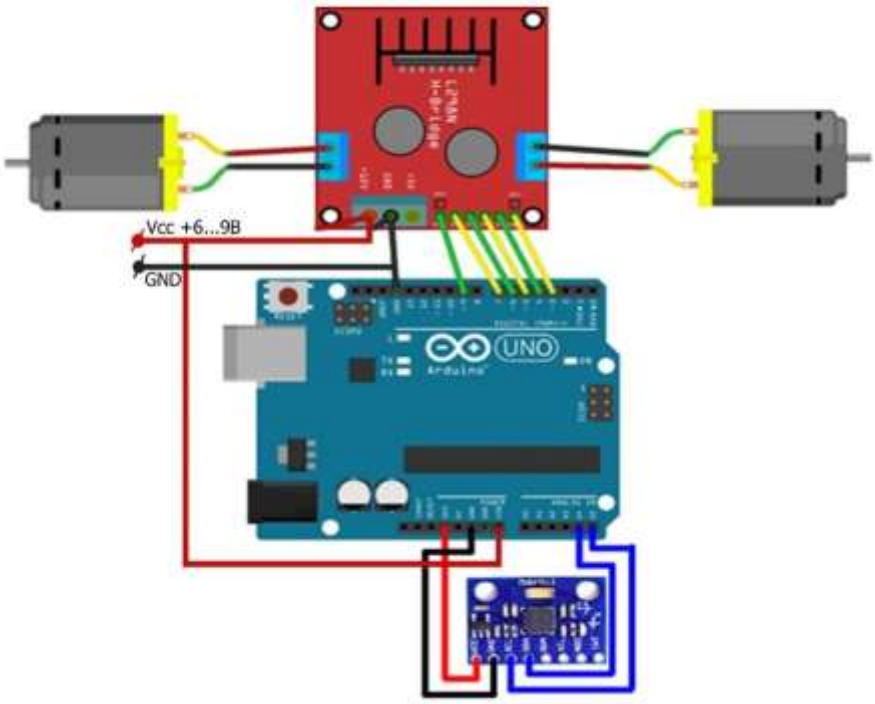


Рисунок 2.25 – Схема робота-балансира

При цьому двигуни не обертатимуться, поки не буде сигналу HIGH на піні 7 для першого двигуна або на 12 піні для другого. Зупинити їх обертання можна подачею сигналу LOW на ті самі, зазначені вище, піни. Для керування швидкістю обертання використовується ШІМ-сигнал.

Якщо після запуску робота він падає, потрібно підбирати кут нахилу корпуса:

```
//PID
double originalSetpoint = 96.50; //186.50
```

Зверніть увагу на наступні числові характеристики. У разі надмірного «підсмикування» їх потрібно міняти на закоментовані значення:

```
//adjust these values to fit your own design
double Kp = 70.45;   //10
double Kd = 4.4; //0.18
double Ki = 600; //25
```

2.8 Використання технології Bluetooth

У промисловому Інтернеті Речей пристроям часто потрібно надсилати невеликі пакети даних у дуже «зашумленому» іншими сигналами оточенні. За необхідності налаштування десятків або сотень пристроїв Wi-Fi стає занадто

складним і неоптимальним вибором. Недоліком Bluetooth є низька пропускна спроможність (порівняно з Wi-Fi), але її вистачає для передачі технічної і текстової інформації між пристроями.

Bluetooth також використовується для облаштування «розумного будинку», оскільки пристрої автоматично об'єднуються в осередкову мережу [26]. Це означає, що всі пристрої можуть продовжувати спілкуватися між собою, навіть якщо майстер відключається – його «повноваження» приймає інший пристрій і продовжує роботу.

Організація Bluetooth Special Interest Group офіційно прийняла стандарт Bluetooth 5 у грудні 2016 року. Нова версія розроблена спеціально з урахуванням потреб «розумного будинку» та Інтернету Речей: порівняно з попередньою версією, діапазон частот розширено в 4 рази, вдвічі збільшилася швидкість пропускання і на 800% поліпшена проникність сигналу крізь перешкоди. Bluetooth 5.0 підтримує зворотну сумісність, з ним можна використовувати пристрої попередніх версій, але в такому випадку загальні параметри мережі опустяться до характеристик найстаршого підключеного пристрою.

Arduino дозволяє завантажувати скетчі не тільки проводовим способом через USB, але і безпроводовим, наприклад, за допомогою Bluetooth. Завдяки такому способу прошивки програмувати Arduino можна на деякій відстані від комп'ютера, що, безсумнівно, зручно під час налагодження якогось стаціонарного об'єкта з Arduino, який проблематично піднести і підключити до комп'ютера через USB.

Для завантаження прошивки за Bluetooth можна використовувати класичний модуль HC-06. У даному уроці буде показаний приклад програмування Arduino по Bluetooth через HC-06 (рис. 2.26) [26].

Рисунок 2.26 – Bluetooth-модуль HC-06

Модуль працює в пасивному режимі, тобто Bluetooth-модуль може працювати в одному з двох режимів: Master або Slave.

Slave (ведений) – найбільш часто зустрічається режим роботи, в якому пристрій очікує підключення, сам при цьому не здатний з'єднатися з іншим пристроєм. Даний режим може використовуватися в роботі Bluetooth-розетки, метеостанції і будь-якому іншому пристрої, до якого планується підключення, наприклад, зі смартфона або комп'ютера.

У режимі Master (ведучий) пристрій, навпаки, є ініціатором з'єднання і може підключитися до Slave-модуля. Для створення зв'язку між двома Arduino за допомогою Bluetooth будуть потрібні два модуля, один з яких налаштований на роботу в режимі master, інший – slave, якщо необхідно підлаштувати параметри модуля за допомогою AT-команд.

Розпінування:

– STATE – сюди дублюється сигнал з вбудованого світлодіода, коли модуль активний, світлодіод блимає, коли зв'язок встановлено – горить;

– RXD – на цьому піні модуль приймає дані (тобто у вашому скетчі сюди слід відсилати дані);

– TXD – сюди модуль відправляє дані;

– GND – земля;

– VCC – живлення 5 В;

– EN – увімк./вимк., якщо подати сюди логічну одиницю (або просто логічну одиницю), то модуль вимкнеться, якщо логічний нуль (або просто не підключати цей пін) не працюватиме.

При бажанні можна налаштувати деякі параметри модуля, наприклад, його ім'я, швидкість передачі даних, pin-код.

Тепер розглянемо програму, яка виконуватиметься на Arduino. Її завданням буде побайтове (по одному символу) зчитування команди за програмно організованим послідовним портом, після виявлення символу кінця рядка, вона має проаналізувати прийняту команду і виконати її, змінюючи колір RGB-світлодіода або активуючи п’єзовипромінювач.

Текст програми такий:

```
#define RED_LED 9
#define GREEN_LED 10
#define BLUE_LED 11
#define BUZZER 6
String command = ««;
char symbol;
void setRed(int intensity)
{
  analogWrite(RED_LED, intensity);
```

```
}

void setGreen(int intensity)
{
  analogWrite(GREEN_LED, intensity);
}

void setBlue(int intensity)
{
  analogWrite(BLUE_LED, intensity);
}

void playSound()
{
  tone(BUZZER, 2000);
  delay(500);
  noTone(BUZZER);
}

void playMelody1()
{
  for (int i = 1000; i <4000; i++)
  {
    tone(BUZZER, i);
    delay(1);
  }
  noTone(BUZZER);
}

void playMelody2()
{
for (int i = 4000; i> 1000; i--)
  {
    tone(BUZZER, i);
    delay(1);
  }
  noTone(BUZZER);
}

void setup() {
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  Serial.begin(9600);
```

```
}

void loop()
{
  if ( Serial.available())
  {
    symbol =  Serial.read(); Serial.print(symbol);
    if (symbol == '\n')
    {
      switch (command.charAt(0))
{
        case 'R':
          setRed(command.substring(1).toInt());
        break;
        case 'G':
          setGreen(command.substring(1).toInt());
        break;
        case 'B':
          setBlue(command.substring(1).toInt());
        break;
        case 'S':
          playSound();
        break;
        case 'M':
          switch (command.charAt(1))
          {
            case '1':
         playMelody1();
            break;
            case '2':
              playMelody2();
            break;
          }
        break;
      }
      command = ««;
    }
    else
    {
      command = command + symbol;Serial.print(command);
}
  }
}
```

Тепер можна обмінюватися даними з Arduino через протокол Bluetooth, наприклад, використовуючи програму для мобільного телефону.

2.9 Питання для самоперевірки

2.9.1 Тест. Управління двигунами постійного струму

1. Для чого використовуються вбудовані функції?
   1) для збільшення швидкості роботи програми;
   2) для спрощення читання коду;
   3) щоб зменшити розмір програми;
   4) для видалення непотрібних функцій.
2. Що називають прототипом функції?
   1) опис функції, включаючи її ім'я, тип значення, імена і типи параметрів;
   2) опис функції, включаючи її ім'я, тип значення, типи параметрів;
   3) ім'я функції і тип значення, що повертається;
   4) опис функції, включаючи її ім'я, тип значення, імена і типи.
3. В яких випадках необхідно використовувати оператор return у тілі функції?
   1) завжди;
   2) якщо необхідно, щоб функція повернула значення;
   3) якщо необхідно забезпечити вихід з функції в довільному місці;
   4) якщо зазначений тип значення, яке повертається, в тому числі й void.
4. Напрямок обертання мотора-редуктор залежить від...
   1) способу підключення його до плати Arduino;
   2) полярності прикладеного живлення;
   3) типу функції програмного керування;
   4) кількості використовуваних моторів у роботі.
5. Драйвер двигуна постійного струму дає можливість...
   1) керувати трьома сервомоторами;
   2) змінювати полярність мотора-редуктора;
   3) керувати одним або двома двигунами постійного струму;
   4) змінювати полярність сервомотора.
6. Якщо в проєкті використовується кілька двигунів, то потрібно переконатися в тому, що...
   1) відбувається нагрів корпусу двигунів;

2) із підключенням мотор-редуктор короткий час споживає мало струму;
3) драйвер двигуна постійного струму можна використовувати для живлення Arduino;
4) що у них витримана однакова полярність з підключенням.

Відповіді на тест

1	2	3	4	5	6
2)	3)	2)	2)	3)	2)

2.9.2 Тест. Управління кроковим двигуном

1. Цілий тип даних в С++…
   1) Char;
   2) Double;
   3) Bool;
   4) Int.
2. Функція – це…
   1) послідовність операцій, яка знаходиться поза інших функцій і може бути використана в будь-якій іншій функції;
   2) різновид керуючої конструкції у високорівневих мовах програмування, призначена для організації багаторазового виконання набору інструкцій;
   3) абстрактний тип даних, що являє собою упорядкований набір значень, в якому деяке значення може зустрічатися більше одного разу;
   4) конструкція більшості мов програмування, що дозволяє утримувати в собі набір змінних різних типів.
3. Оберіть правильний приклад опису функції loop()

1) void loop()
{
myStepper.step(-32);
}
	2) int loop()
	{
	myStepper.step(-32);
	}

3) loop()
{
myStepper.step(-32);
}
	2) char loop()
	{
	myStepper.step(-32);
	}

4. Функції, які мають схоже призначення, об'єднуються в...
    1) скрипти;
    2) скетчі;
    3) інструменти;
    4) бібліотеки.
5. Бібліотека підключається на початку програми рядком виду...
    1) void setup()
    2) void loop()
    3) #include <назва>
    4) #define <назва>
6. Вид двигунів, який дозволяє керувати обертанням ротора покроково:
    1) сервомотор;
    2) драйвер двигуна;
    3) двигун постійного струму;
    4) кроковий двигун.

Відповіді на тест

1	2	3	4	5	6
4)	1)	1)	4)	3)	4)

2.9.3 Тест. Керування пристроями IoT за допомогою інфрачервоного зв'язку

1. COM-порт являє собою...
    1) послідовний односпрямований інтерфейс;
    2) послідовний двоспрямований інтерфейс;
    3) послідовний триспрямований інтерфейс;
    4) послідовний чотириспрямований інтерфейс.
2. Яке призначення COM-порту?
    1) обмін бітовою інформацією;
    2) обмін байтовою інформацією;
    3) відправка байтової інформації;
    4) відправка інформації.
3. Довжина хвилі ІЧ-випромінювання...
    1) більше 1000 нм;
    2) більше 780 нм;
    3) менше 400 нм;
    4) менше 780 нм.

4. Напівпровідниковий діод, в якому забезпечується можливість впливу оптичного випромінювання на *p*-*n*-перехід:
   1) тунельний діод;
   2) світлодіод;
   3) фотодіод;
   4) звернений діод.
5. Мікросхема приймача ІЧ-випромінювання включає...
   1) PIN-фотодіод;
   2) звернений діод;
   3) тунельний діод;
   4) діод Шотки.
6. Мікросхема приймача ІЧ-випромінювання включає...
   1) квадратичний діодний детектор;
   2) фазовий детектор;
   3) амплітудний детектор;
   4) демодулятор.

Відповіді на тест

1	2	3	4	5	6
2)	1)	2)	3)	1)	3)

2.9.4 Тест. Реалізація руху мобільного робота за лінією

1. Якими знаками закінчується більшість рядків коду в С++?
   1) : (двокрапка)
   2) ; (крапка з комою)
   3) , (кома)
   4) . (крапка)
2. Кожна оптопара складається з …
   1) піроелектричного кристала;
   2) випромінювача і приймача;
   3) світлочутливої напівпровідникової пластини;
   4) ротора і підшипника.
3. В основі роботи оптопари лежить...
   1) здатність поверхні поглинати частину світла, яке на неї падає;
   2) здатність поверхні поглинати звукові хвилі;
   3) здатність поверхні відбивати звукові хвилі;
   4) здатність поверхні відбивати частину світла, яке на неї падає.

4. За однакової відстані до поверхні оптопара може не спрацювати, якщо поверхня...
   1) білого кольору;
   2) чорного кольору;
   3) червоного кольору;
   4) синього кольору.
5. Бібліотека IRremote.h використовується для...
   1) крокового двигуна;
   2) сервомотора;
   3) двигуна постійного струму;
   4) мотора-редуктора.
6. Як в C++ записується оператор присвоювання?
   1) =
   2) ==
   3) !=
   4) <=

Відповіді на тест

1	2	3	4	5	6
2)	2)	4)	2)	3)	1)

2.9.5 Тест. Основи роботи з далекоміром

1. За допомогою далекоміра можна...
   1) виявити нахил під час повороту;
   2) виявити кольорову лінію;
   3) виявити рух об'єкта;
   4) встановити відсутність або наявність перешкоди на заданій відстані або ближче.
2. Який датчик використовується як лічильник обертів або вимірювач швидкості обертання?
   1) ультразвуковий далекомір;
   2) датчик дотику;
   3) далекомір;
   4) відбиваюча оптопара.
3. Дистанція якого датчика обчислюється за формулою: Ширина імпульсу (в мікросекундах) / 58 = Дистанція (в сантиметрах)...
   1) ультразвуковий далекомір;

2) датчик дотику;
3) далекомір;
4) відбиваюча оптопара.

4. У модуль далекоміра входить...
   1) піроелектричний кристал;
   2) світлодіод інфрачервоного випромінювання і фотоприймач;
   3) оптичний датчик;
   4) кольоровий світлодіод.
5. Кожна оптопара складається з...
   1) піроелектричного кристала;
   2) світлочутливої напівпровідникової пластини;
   3) випромінювача і приймача;
   4) ротора і підшипника.
6. Частота приймача далекоміра...
   1) 28 кГц;
   2) 38 кГц;
   3) 48 кГц;
   4) 58 кГц.

Відповіді на тест

1	2	3	4	5	6
4)	2)	1)	2)	3)	2)

2.9.6 Тест. Застосовування акселерометра в робототехніці

1. Гіроскоп являє собою пристрій...
   1) який вимірює проєкцію удаваного прискорення;
   2) виявлення в контрольованій зоні руху сторонніх об'єктів;
   3) визначення відстані до об'єкта;
   4) реагує на зміну кутів орієнтації контрольованого тіла.
2. Акселерометр – це пристрій …
   1) який вимірює проєкцію удаваного прискорення;
   2) виявлення в контрольованій зоні руху сторонніх об'єктів;
   3) визначення відстані до об'єкта;
   4) реагує на зміну кутів орієнтації контрольованого тіла.
3. Функція MPU6050_ReadData()...
   1) зчитує дані з акселерометра, гіроскопа і датчика температури;
   2) скидає налаштування чіпа на значення за замовчуванням;

3) налаштовує вбудований фільтр низьких частот;
4) використовує масштабні коефіцієнти для розрахунку результату.

4. Функція MPU6050_ReadData()...
   1) використовує масштабні коефіцієнти для розрахунку результату;
   2) зчитує дані з акселерометра, гіроскопа і датчика температури;
   3) налаштовує вбудований фільтр низьких частот;
   4) скидає налаштування чіпа на значення за замовчуванням.
5. Функція MPU6050_ResetWake()…
   1) використовує масштабні коефіцієнти для розрахунку результату;
   2) налаштовує вбудований фільтр низьких частот;
   3) скидає налаштування чіпа на значення за замовчуванням;
   4) зчитує дані з акселерометра, гіроскопа і датчика температури.
6. Функція MPU6050_SetDLPF(int BW)…
   1) використовує масштабні коефіцієнти для розрахунку результату;
   2) налаштовує вбудований фільтр низьких частот;
   3) скидає налаштування чіпа на значення за замовчуванням;
   4) зчитує дані з акселерометра, гіроскопа і датчика температури.

Відповіді на тест

1	2	3	4	5	6
4)	1)	4)	2)	3)	2)

2.9.7 Тест. Програмування Arduino із застосуванням технології Bluetooth

1. Символ кінця рядка позначається...
   1) \t
   2) \n
   3) \e
   4) \f
2. База даних – це…
   1) об'єкт, який можна використовувати для створення інтерфейсу застосунку, який призначений для користувача;
   2) засіб вибору необхідної інформації;
   3) обслуговуючий пристрій у системах автоматичної обробки інформації;
   4) спеціальне сховище інформації, яке обслуговує ОС і знаходиться на доступному в пристрої носії інформації.

3. Метод – це…
   1) засіб вибору необхідної інформації;
   2) деяка сутність у комп'ютерному просторі, яка володіє певним станом і поведінкою;
   3) це функція або процедура, яка належить певному класу чи об'єкту;
   4) система з компілятора, який переводить вихідний код програми в проміжне подання.
4. Для роботи з базою даних в AI можуть використовуватися функції...
   1) записи і читання;
   2) формування запиту;
   3) створення змінної;
   4) створення форм.
5. В режимі роботи Slave Bluetooth-модуль...
   1) очікує підключення;
   2) очікує підключення і може підключитися;
   3) відключений;
   4) є ініціатором з'єднання і може підключитися.
6. В режимі роботи Master Bluetooth-модуль...
   1) очікує підключення;
   2) очікує підключення і може підключитися;
   3) відключений;
   4) є ініціатором з'єднання і може підключитися.

Відповіді на тест

1	2	3	4	5	6
2)	4)	3)	1)	1)	4)

3 ЕЛЕМЕНТИ КЕРУВАННЯ ІНТЕЛЕКТУАЛЬНИМ БУДИНКОМ ЧЕРЕЗ МЕРЕЖУ ETHERNET

3.1 Відправка даних на віддалений сервер

3.1.1 Теоретичні відомості

Даний підрозділ навчального посібника присвячений проєкту створення метеостанції, яка відправлятиме дані про температуру та вологість навколишнього середовища на віддалений сервер, використовуючи Wi-Fi.

Зазвичай схема Wi-Fi-мережі містить не менше однієї точки доступу і не менше одного клієнта, який підключається до точки. Також можливе підключення двох клієнтів у режимі точка–точка (Ad-hoc), коли точка доступу не використовується, а клієнти з'єднуються безпосередньо. Точка доступу передає свій ідентифікатор мережі (SSID) за допомогою спеціальних сигнальних пакетів на швидкості 0,1 Мбіт/с кожні 100 мс. Тому 0,1 Мбіт/с – найменша швидкість передачі даних для Wi-Fi. Знаючи SSID мережі, клієнт може з'ясувати, чи можливе підключення до даної точки доступу. Під час потрапляння в зону дії двох точок доступу з ідентичними SSID приймач може обирати між ними на підставі даних про рівень сигналу.

Стандарт Wi-Fi не описує всіх аспектів побудови безпроводових локальних мереж, тому кожен виробник обладнання вирішує цю задачу по-своєму, застосовуючи ті підходи, які він вважає найкращими з тієї чи іншої точки зору.

Хоча цим використання Wi-Fi не обмежується, звичайно ж, найчастіше такий зв'язок використовується для отримання безпроводового доступу до мережі Інтернет. Саме так ми використовуватимемо його в цьому уроці.

Спосіб організації роботи між віддаленими комп'ютерами, який найчастіше використовується в мережі Інтернет, є система «клієнт–сервер» (рис. 3.1) [27].

Така система являє собою обчислювальну або мережну архітектуру, в якій мережні запити замовників послуг, так звані клієнти, виконуються постачальниками даних, які називаються серверами. Фізично і клієнт, і сервер є ні що інше, як програмне забезпечення. Зазвичай вони взаємодіють через комп'ютерну мережу за допомогою мережних протоколів і знаходяться на різних обчислювальних машинах, але можуть виконуватися також і на одній машині. Програми-сервери очікують від клієнтських програм запити і надають

їм свої ресурси у вигляді даних (наприклад, завантаження файлів, потокове відтворення мультимедіа чи робота з базами даних) або сервісних функцій (наприклад, робота з електронною поштою, спілкування за допомогою систем миттєвого обміну повідомленнями, перегляд web-сторінок тощо).

Рисунок 3.1 – Система «клієнт–сервер»

Таким чином, стає зрозуміло, що, як правило, комп'ютери і програми, які входять до складу інформаційної системи, не є рівноправними. Деякі з них володіють ресурсами (файлова система, процесор, принтер, база даних тощо), інші мають можливість звертатися до цих ресурсів.

Для спілкування між пристроями в розподілених системах необхідно мати можливість адресувати пристрої, від яких і до яких рухаються запити. Зазвичай для цього в мережах використовується IP-адреси, що складаються з чотирьох чисел від 0 до 255. Крім IP-адреси кожен з пристроїв має унікальну MAC-адресу, за якою його однозначно можна визначити всередині мережі. Якщо на комп'ютері водночас запущено декілька програм-серверів, то до адреси комп'ютера додається ще так званий «номер порту», який визначає, якому з виконуючих серверів надійшли дані. Так, наприклад, сервери, які відображатимуть сторінки сайтів, зазвичай займають порт 80, і адреса сайту в мережі Інтернет має виглядати так: www.google.com:80. Однак, за замовчуванням програма-браузер (клієнт) формує запити саме до сервера з портом 80, тому номер порту зазвичай не вказується.

Одним з прикладів, який добре ілюструє роботу систем «клієнт–сервер», є проєкт «Налаштування ThingSpeak» (рис. 3.2) [28]. Розташований він на сайті https://thingspeak.com/ (IP-адреса 184.106.153.149) і призначений для віддаленого моніторингу показань різних датчиків (не тільки температури і вологості, але й будь-яких інших), відображення показників датчиків на графіках, ведення архіву зібраних даних тощо.

Особливість проєкту полягає в тому, що абсолютно будь-яка людина може виготовити або придбати пристрій, здатний передавати деякі дані з датчиків в установленому форматі, і підключити його до проєкту.

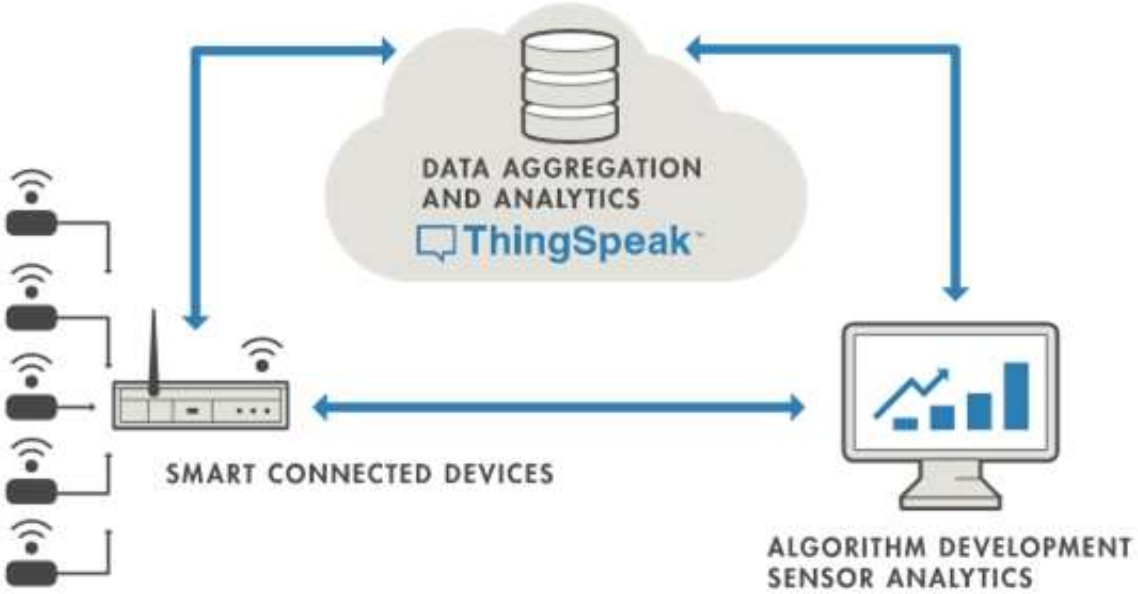


Рисунок 3.2 – Архітектура проєкту «ThingSpeak»

При цьому зовсім не обов'язково виставляти дані зі своїх датчиків на загальний огляд, оскільки датчики можуть бути як публічними (відкритими для всіх), так і приватними, щоправда, з обмеженням за кількістю.

Ще один великий плюс проєкту полягає в тому, що на даний момент створено програми для мобільних пристроїв (на операційних системах Android, iOS та ін.), які дозволяють отримувати дані з сервера проєкту і обробляти їх на мобільному пристрої.

3.1.2 Wi-Fi-модуль ESP8266

Модуль ESP8266 являє собою самостійний мікроконтролер із вбудованим Wi-Fi-інтерфейсом [29]. За необхідності цей мікроконтролер може бути запрограмований в Arduino IDE і використовуватися взагалі без додаткового обладнання.

На базі чіпа ESP8266 вже випущено незліченну кількість модулів з послідовним інтерфейсом, що дозволяє дуже легко підключати їх до Arduino. Серед таких модулів: ESP-01, ESP-02, ESP-03 і т.д. – загалом 12 різних модифікацій. Модулі відрізняються один від одного типами антен, розмірами, наявністю екрануючих корпусів, кількістю виведених на роз'єм портів загального призначення та ін. У наступному прикладі ми будемо використовувати модуль ESP-12 (рис. 3.3), хоча підключення модулів практично аналогічно і можна використовувати будь-який з них.

У відповідності з документацією на модуль, його живлення становить 3,3 В і напруга на входах теж не повинна перевищувати цього значення, а отже, для підключення лінії RX необхідно використовувати перетворювач

рівня, найпростішим варіантом якого може бути резисторний подільник напруги. Але, як показує практика, входи модуля толерантно співвідносяться з напругою 5 В і модуль функціонує нормально. Тому ми будемо живити модуль напругою 3,3 В, а виводи підключимо безпосередньо до Arduino.

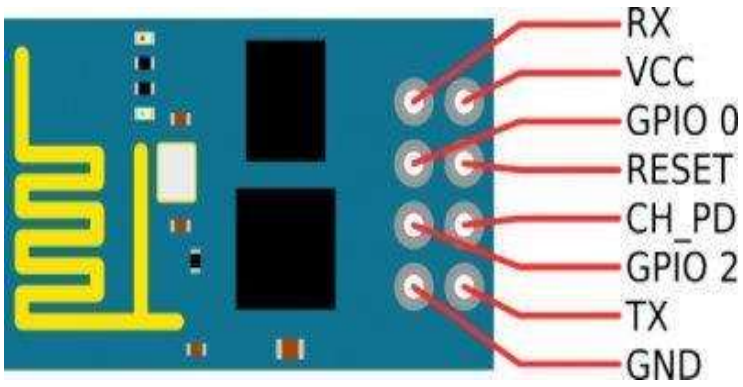


Рисунок 3.3 – Розпіновка модуля ESP-12

Як вже було сказано, керується Wi-Fi-модуль за послідовним інтерфейсом, а керування відбувається за допомогою відправки в нього AT-команд. Термін «AT-команда» походить від набору команд, розроблених компанією Hayes для модему власної розробки. Набір команд складався з серій коротких текстових рядків, які об'єднувалися разом, щоб сформувати повні команди операцій, таких як набір номера, початок з'єднання або зміна параметрів підключення. Для того щоб модем розпізнавав ці команди, вони мали бути записані у специфічній формі, і кожна команда завжди починалася літерами «AT» або «at». Такі AT-команди до цього часу є популярним методом керування різним устаткуванням зв'язку.

Кожна команда модуля (рядок символів, що починається на префікс AT і завершується символом кінця рядка – CR, код ASCII-13) може містити один з чотирьох типів команд:

– тест (формат – AT+CMD=?);

– запит (формат – AT+CMD?);

– установка (формат – AT+CMD=parameter);

– виконання (формат – AT+CMD).

Базові команди модуля:

– AT – перевірка на запуск модуля. Якщо все в порядку, на цю команду модуль має відповісти «OK»;

– AT+RST – перезавантаження модуля. Після застосування цієї команди модуль перезавантажується і відповідає словом «ready». Для перезавантаження модуля може знадобитися час до однієї секунди;

– ATE – керування повтором прийнятих символів команд (відлуння). Команда буває корисна для перевірки, чи приймає модуль дані за послідовним інтерфейсом. ATE0 забороняє відлуння, ATE1 дозволяє його.

Wi-Fi-команди модуля

AT+CWMODE – вибір режиму роботи Wi-Fi. Формат команди:

AT+CWMODE=mode,

де mode – режим роботи:

1 – режим робочої станції (клієнта), яка може підключатися до точки доступу (основний режим роботи);

2 – режим точки доступу, до якої можуть підключатися робочі станції;

3 – суміщений режим. Якщо вибір режиму пройшов успішно, модуль відповідає «ОК», а в разі помилки – «ERROR».

AT+CWJAP – підключення до точки доступу під час роботи в режимі робочої станції. Формат команди – AT+CWJAP=ssid,pwd, де ssid – ідентифікатор точки доступу, pwd – її пароль. На цю команду модуль також відповідає «ОК» або «ERROR». Для підключення до точки доступу потрібно кілька секунд.

Команди, пов'язані безпосередньо з передачею даних

AT+CIPMUX – встановлення режиму з'єднання (одиночне або множинне). Формат команди:

AT+CIPMUX=mode,

де mode – режим з'єднання (0 – одиночне, 1 – множинне).

AT+CIPSTART – встановлення зв'язку з віддаленим сервером або створення локального порту і початок з'єднання. Формат команди:

AT+CIPSTART=type, addr, port,

де type – тип з'єднання («TCP» або «UDP»);

addr – IP-адреса точки підключення (так само, як і тип підключення, вказується в лапках);

port – порт віддаленої точки підключення (також у лапках).

Існує ще один спосіб виклику даної команди для створення множинних з'єднань, але для вирішення нашої задачі він не знадобиться. Відповідь модуля на цю команду – також «ОК» або «ERROR».

AT+CIPSEND – передача даних на віддалену точку. Формат команди:

AT+CIPSEND=length,

де length – кількість переданих на віддалену точку даних.

Виконуючи цю команду, модуль відповідає запрошенням виду «>» і чекає переданих даних. Після отримання заданої в команді кількості даних модуль переходить назад у режим АТ-команд. Дана команда також має кілька варіантів використання, але ми використовуватимемо тільки варіант з жорстко заданою довжиною повідомлення в одиночному режимі (так званий режим «normal»).

AT+CIPCLOSE – закриття з'єднання з віддаленим сервером. Обов'язкова для виконання після відправки даних на сервер.

3.1.3 Датчик температури та вологості DHT11

Датчик температури та вологості (рис. 3.4) складається з двох частин: ємнісного датчика температури і гігрометра [30].

Рисунок 3.4 – Датчик температури та вологості DHT11

Перший використовується для вимірювання температури, другий – для вологості повітря. Чіп, який знаходиться всередині, може виконувати аналого-цифрові перетворення і видавати цифровий сигнал, який зчитується за допомогою мікроконтролера.

Характеристики датчика:

– споживаний струм – 2,5 мА (максимальне значення під час перетворення даних);

– вимірює вологість у діапазоні від 20% до 80%. Похибка може складати до 5%;

– застосовується під час вимірювання температури в інтервалі від 0 до 50 градусів (точність – 2%);

– живлення – від 3 В до 5 В;

– один вимір в одиницю часу (секунду), тобто, частота становить 1 Гц.

Підключення: VCC до +5, GND – до землі, третій контакт – до будь-якого вільного піна на платі Arduino [30]. Номер піна потрібно буде потім вказати у скетчі.

Перед написанням скетчу необхідно переконатися, що встановлена бібліотека для роботи з датчиками вологості і температури DHT.h.

3.1.4 Створення акаунту на сервері Thingspeak

Для того щоб мати можливість відправляти дані з датчиків на сервер проєкту, необхідно виконати такі кроки [28].

1. Зареєструйте безкоштовний акаунт https://thingspeak.com/. Натисніть кнопку «Get Started For Free» (рис. 3.5).

Рисунок 3.5 – Початок реєстрації на сервері Thingspeak

2. Тепер заповніть форму, яка з'явилася, поставте позначку і створіть акаунт (рис. 3.6).

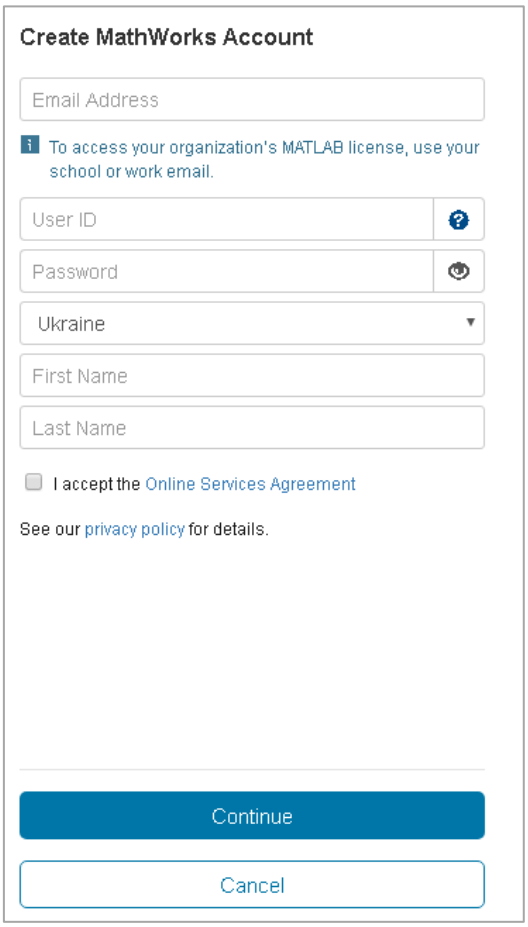


Рисунок 3.6 – Створення акаунту

3. Перейдіть у Channels, а далі в Create New Channel (рис. 3.7).

Рисунок 3.7 – Перехід у розділ «Create New Channel»

У даному розділі необхідно заповнити:

– Name – назва каналу;

– Description – опис проєкта.

У графі Field1 введемо ім'я першої змінної: Temperature (Температура).

У Field2 введемо ім'я другої змінної: Humidity (Вологість), попередньо потрібно поставити позначку за нею, як і у першій змінній.

Інші параметри залишаємо за замовчуванням і натискаємо зелену кнопку «SaveChannel» (рис. 3.8).

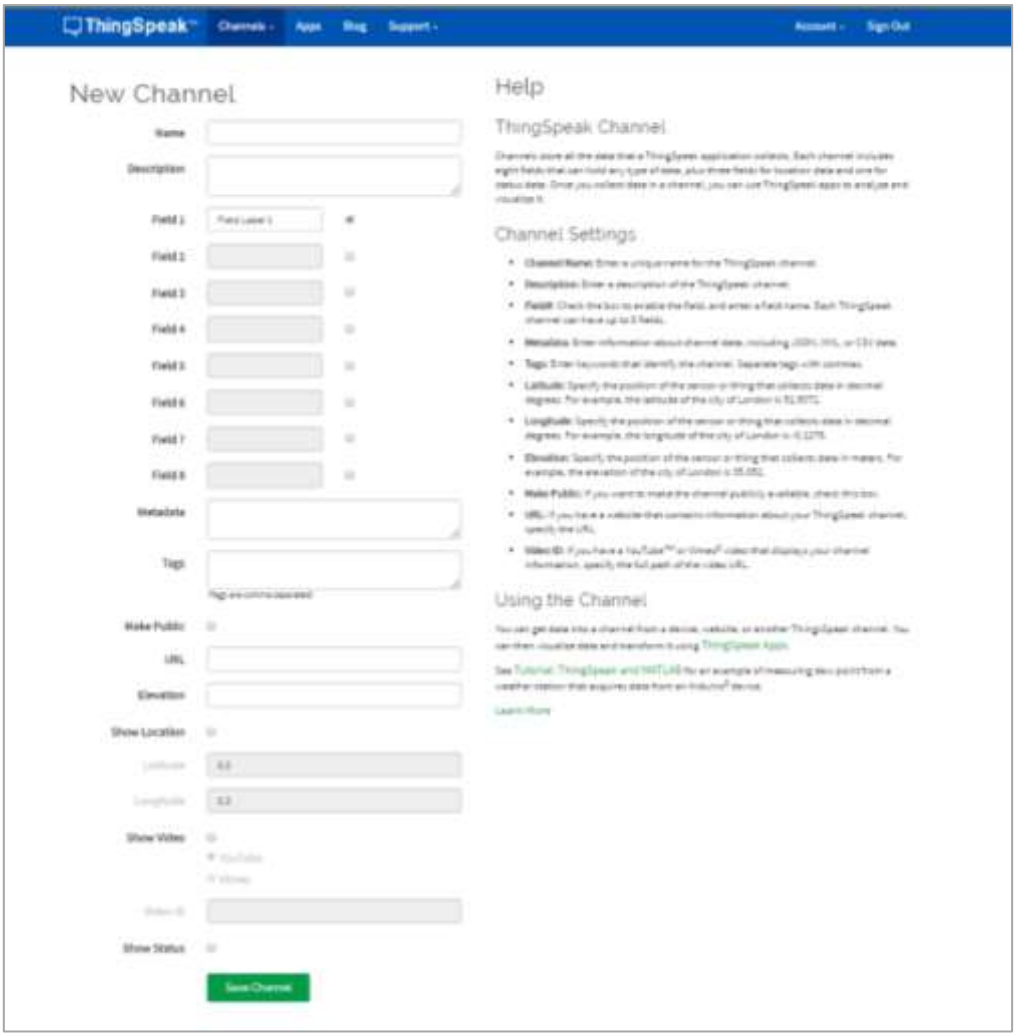


Рисунок 3.8 – Заповнення полів нового каналу передачі даних

4. Перейдіть в API Keys і отримайте свій унікальний KEY (рис. 3.9).

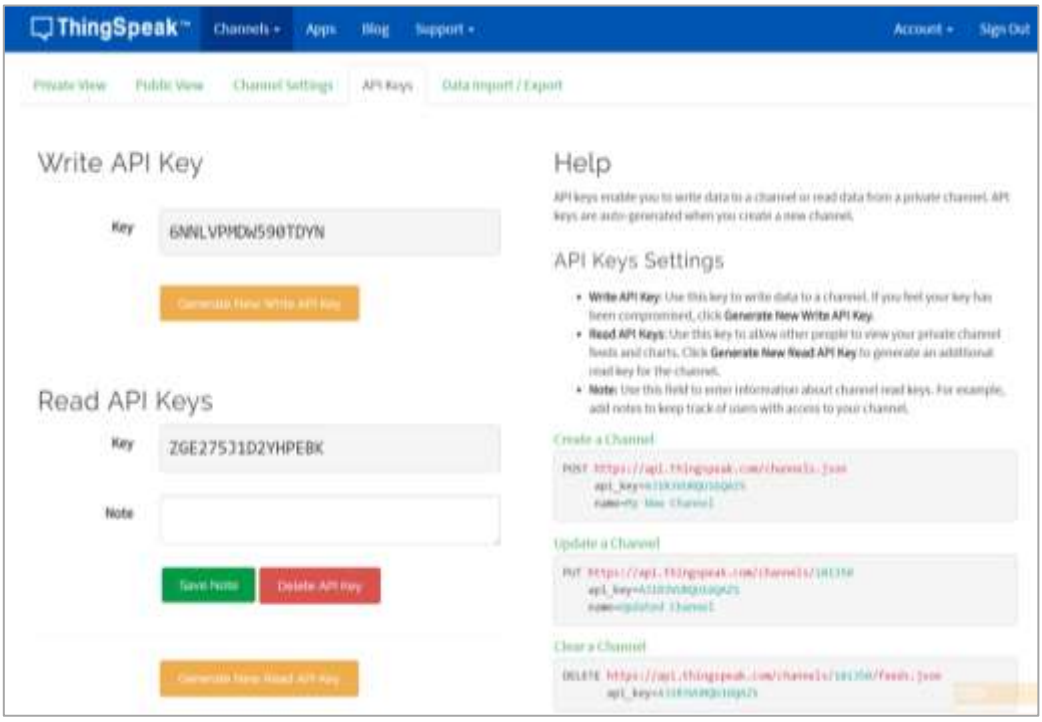


Рисунок 3.9 – Отримання унікального KEY

5. Проведіть перевірку, вставивши в рядок вашого браузера такий код (рис. 3.10):

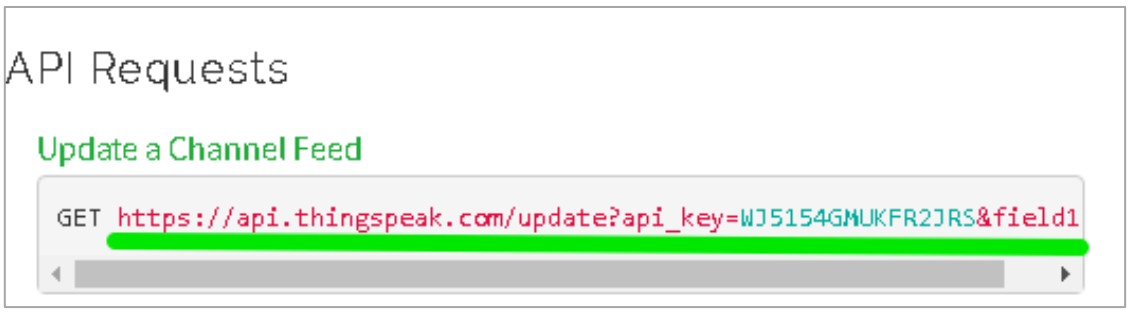


Рисунок 3.10 – Перевірка правильності запиту

6. Перегляд результатів (рис. 3.11).

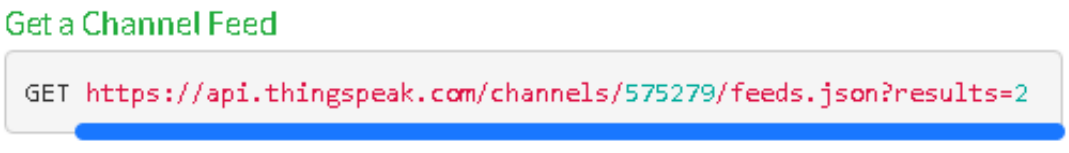


Рисунок 3.11 – Рядок для перегляду результатів

Внаслідок проведених операцій отримали зареєстрований API для отримання даних з датчиків метеостанції.

3.1.5 Приклад створення метеостанції з відправкою даних про температуру та вологість навколишнього середовища на віддалений сервер проєкту «ThingSpeak»

По-перше, необхідно підключити Wi-Fi-модуль до ардуіно (рис. 3.12) [29, 30]. Підключення як датчика вологості та температури, так і Wi-Fi-модуля, стандартне і окремого пояснення не вимагає. Wi-Fi-модуль використовує підключення за послідовним портом, тому зручніше використовувати послідовний порт програми Arduino IDE. Приклад підключення Wi-Fi-модуля (датчик вологості та температури підключіть самостійно).

Необхідно звернути увагу, що допустимий діапазон напруги живлення модуля ESP8266 від 3,0 до 3,6 вольт. Подача підвищеної напруги живлення на модуль гарантовано призведе до виходу ESP8266 з ладу.

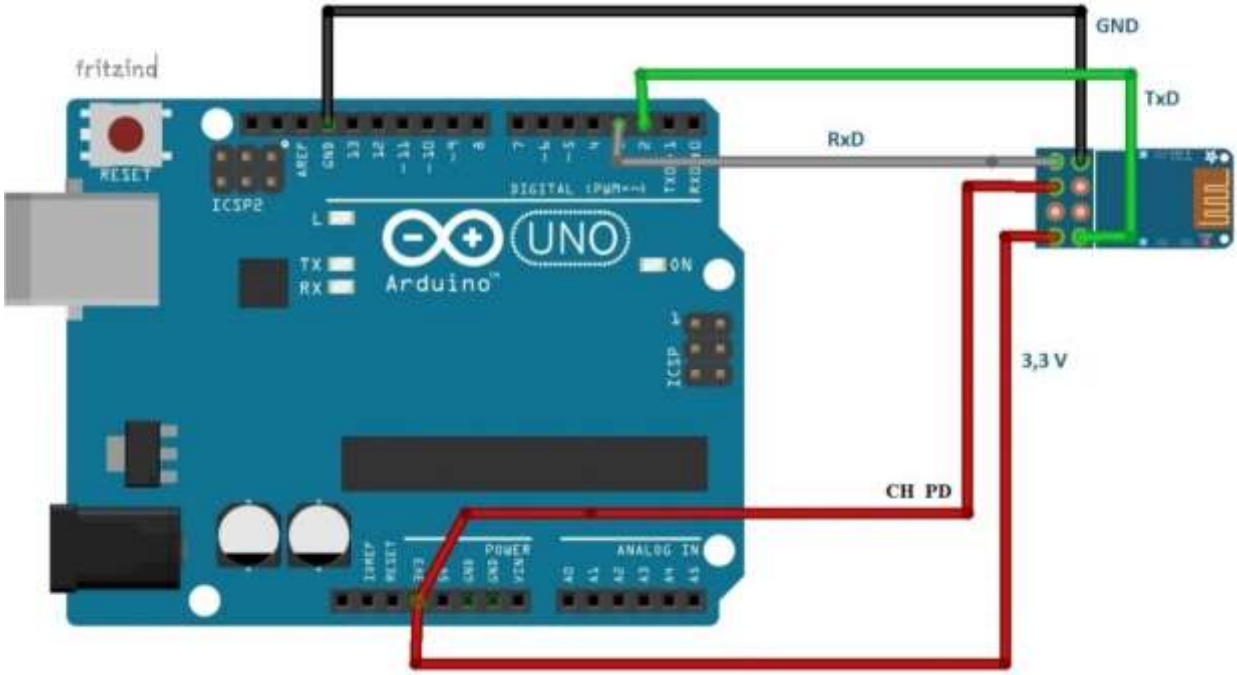


Рисунок 3.12 – Схема підключення WiFi-модуля до Arduino

Перевіримо роботу АТ-команд і реакцію модуля на них, використовуючи модифікований скетч SoftwareSerialExample, що входить до складу бібліотеки SoftwareSerial.h:

```
#include <SoftwareSerial.h>
// підключаємо Wi-Fi-модульна 2 (Rx) і 3 (Tx) піни
SoftwareSerial espSerial (2, 3);
void setup() {
  Serial.begin(9600);
```

```
    espSerial.begin(9600); /* якщо модуль не відповідає, то змініть швидкість на 74880 або 115200 */
}

void loop()
{
  if (espSerial.available())
  {
    Serial.write(espSerial.read());
  }
  if (Serial.available())
  {
    espSerial.write(Serial.read());
  }
}
```

Результат виконання програми показано на рисунку 3.13.

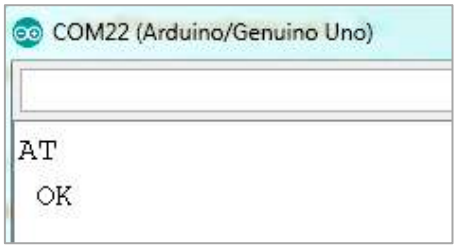


Рисунок 3.13 – Результат виконання програми

У випадку, якщо модуль не відповідає на команди, його потрібно перепрошити за допомогою спеціальної програми NodeMCU.

Крім NodeMCU flasher можливо ще знадобиться LuaLoader – клієнт для роботи з прошивкою. Після завантаження програми для прошивки отримаємо такий інтерфейс (рис. 3.14).

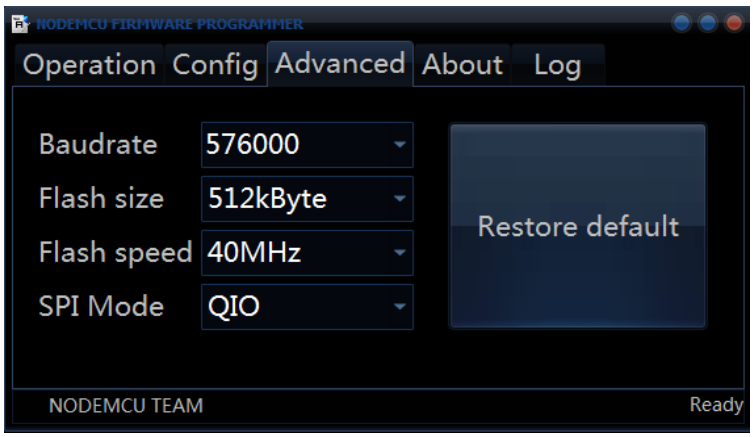


Рисунок 3.14 – Початковий інтерфейс NodeMCU

Вимикаємо чіп, під’єднаємо GPIO0 до землі, вмикаємо живлення та натискаємо на «Restore default». В результаті маємо отримати такий вигляд інтерфейсу програми (рис. 3.15).

Рисунок 3.15 – Завантаження прошивки

Якщо нічого не відбувається і поля AP MAC/STA MAC порожні, перевірте ще раз, щоб GPIO0 був на «землі».

Якщо прошивка почалася, але «зависла» – подивіться дані в закладці Log. Програма може давати збої під час прошивки зазначеного чіпа, але при цьому, без проблем прошиє цей самий чіп під іншою назвою і швидкістю.

Якщо чіп у зазначеному варіанті не має перемикання на 3,3 В, щоб ним скористатися, потрібно відкрити пластиковий корпус і перепаяти провід з 5 В на 3,3 В. Є також варіанти з п'ятьма виходами: 3.3, 5, TX, RX, Gnd.

Після запуску модуля створіть основну програму. Підключення датчика подано на схемі (рис. 3.16).

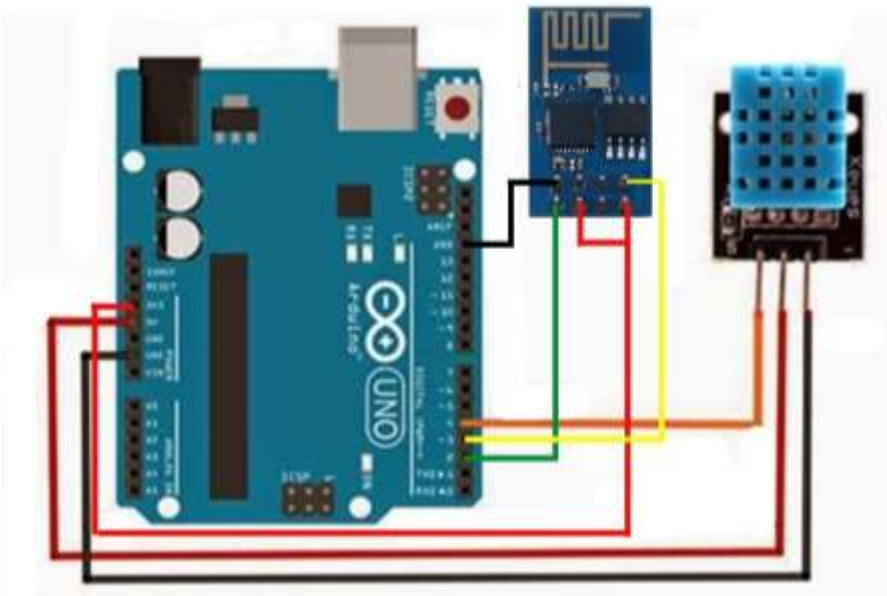


Рисунок 3.16 – Підключення датчика до Arduino

Для датчика вологості та температури необхідно встановити бібліотеку «Adafruit Unified Sensor».

Текст основної програми:

```
#include<SoftwareSerial.h>
#include<DHT.h>
// --- Вводимо назву і пароль вайфай точки доступу  ---
#define SSID «NEONNET»
#define PASS «NEONNET»
#define SERVER «184.106.153.149»
//--- Функції для оновлення даних температури та вологості ---
//--- Замість коду 6NNLVPMDW590TDYN вставте свій унікальний ключ
String GET = «GET /update?key=6NNLVPMDW590TDYN&field1=«;
String GET1 = «GET /update?key=6NNLVPMDW590TDYN&field2=«;
#define DHTPIN 4
#define DHTTYPE DHT11
SoftwareSerial espSerial(2, 3);
DHT sensor(DHTPIN, DHTTYPE);
//--- Функція формування команди відправки температури на сервер ---
void updateTemp(String tenmpF){
String cmd = «AT+CIPSTART=\»TCP\»,\»«;
cmd += SERVER;
cmd += «\»,80»;
sendDebug(cmd);
delay(2000);
if(espSerial.find(«Error»)){
Serial.print(«RECEIVED: Error»);
return;
}
cmd = GET;
cmd += tenmpF;
cmd += «\r\n»;
espSerial.print(«AT+CIPSEND=«);
espSerial.println(cmd.length());
if(espSerial.find(«>«)){
Serial.print(«>«);
Serial.print(cmd);
espSerial.print(cmd);
}else{
sendDebug(«AT+CIPCLOSE»);
}
if(espSerial.find(«OK»)){
Serial.println(«RECEIVED: OK»);
```

```
}else{
Serial.println(«RECEIVED: Error»);
}
}
//--- Функція формування команди відправки вологості на сервер ---
void updateHhumidity(String humidityF){
String cmd = «AT+CIPSTART=\»TCP\»,\»«;
cmd += SERVER;
cmd += «\»,80»;
sendDebug(cmd);
delay(2000);
if(espSerial.find(«Error»)){
Serial.print(«RECEIVED: Error»);
return;
}
cmd = GET1;
cmd += humidityF;
cmd += «\r\n»;
espSerial.print(«AT+CIPSEND=«);
espSerial.println(cmd.length());
if(espSerial.find(«>«)){
Serial.print(«>«);
Serial.print(cmd);
espSerial.print(cmd);
}else{
sendDebug(«AT+CIPCLOSE»);
}
if(espSerial.find(«OK»)){
Serial.println(«RECEIVED: OK»);
}else{
Serial.println(«RECEIVED: Error»);
}
}
//--- Функція відправки даних на сервер ---
void sendDebug(String cmd){
Serial.print(«SEND: «);
Serial.println(cmd);
espSerial.println(cmd);
}
//--- Функція підключення до Wi-Fi-точки ---
boolean connectWi-Fi()
{
  espSerial.println(«AT+CWMODE=1»);
```

```
  delay(2000);
  String cmd = «AT+CWJAP=\»«;
  cmd += SSID;
  cmd += «\»,\»«;
  cmd += PASS;
  cmd += «\»«;
  Serial.println(cmd);
  delay(100);
  espSerial.println(cmd);
  delay(5000);
  if (espSerial.find(«OK»))
  {
    Serial.write(«OK, Connected to Wi-Fi.»);
    return true;
  }
  else
  {
    Serial.write(«Can not connect to the Wi-Fi.»);
return false;
  }
}
void setup()
{
  //--- Налаштування послідовних портів ---
espSerial.begin(115200);
  espSerial.setTimeout(5000);
  Serial.begin(9600);
  Serial.setTimeout(5000);
  Serial.println(«ESP8266+DHT11 Meteo»);
//--- Скидання і перевірка працездатності ESP8266 ---
espSerial.println(«AT»);
    delay(5000);
  if  (espSerial.find(«OK»))
  {
    Serial.write(«ESP8266 not ready.»);
    Serial.println();
    delay(1000);
    connectWi-Fi();
   }
 delay(1000);
//--- Підключення до точки доступу (5 спроб) ---
boolean connected = false;
  for (int i = 0; i <5; i++)
```

```
  {
    if (connectWi-Fi())
    {
      connected = true;
      break;
}
  }
  if (!connected) {
    while (1);
  }
  delay(5000);
  //--- Налаштування датчика температури та вологості---
sensor.begin();
   float temperature;
    float humidity;
    humidity = sensor.readHumidity();
    temperature = sensor.readTemperature();
// Перевірка правильності читання температури та вологості
if (isnan(humidity) || isnan(temperature))
    {
      Serial.write(«Failed to read from DHT sensor!»);
return;
    }
}
void loop()
{float temperature;
float humidity;
humidity = sensor.readHumidity();
temperature = sensor.readTemperature();
char buffer[10];
String tempF = dtostrf(temperature, 4, 1, buffer);
updateTemp(tempF);
delay(60000);
String humidityF = dtostrf(humidity, 4, 1, buffer);
updateHhumidity(humidityF);
delay(60000);
}
```

Тепер можна запрограмувати і запустити Arduino, при цьому не забувши поміняти налаштування точки доступу і свій унікальний ключ у програмі (виділені сірим кольором). Результат виконання програми показано на рисунку 3.17.

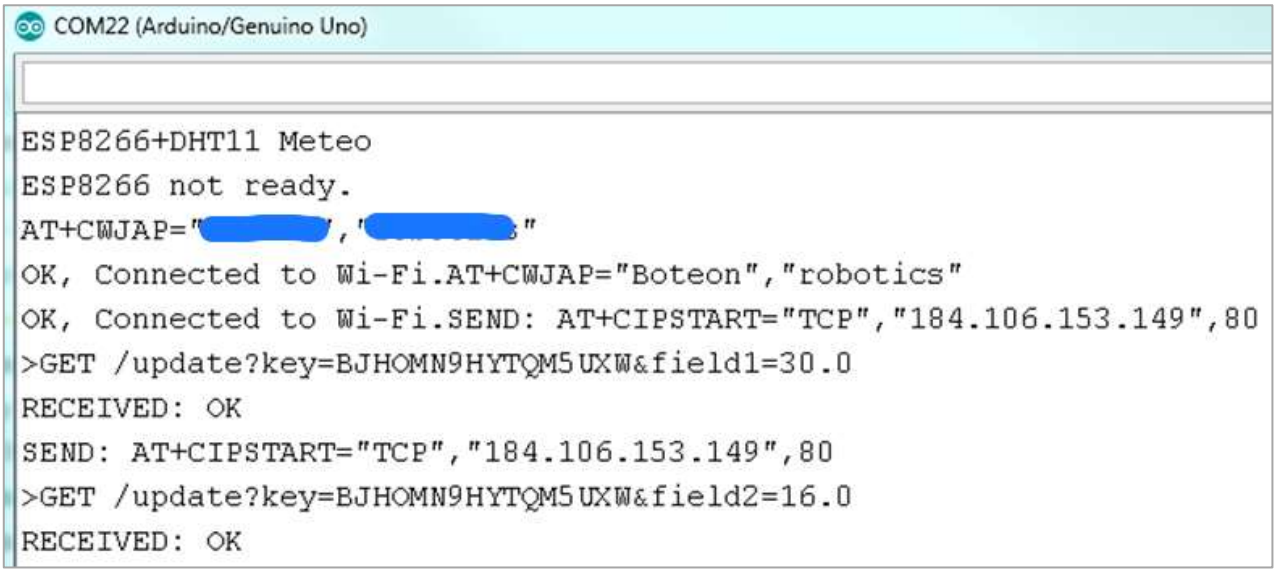


Рисунок 3.17 – Результат виконання програми

Після перевірки передачі і читання даних з сервера можна відсилати дані температури та вологості за допомогою модуля ESP8266 в Інтернет. Результати візуалізації отриманих даних показані на рисунку 3.18.

Рисунок 3.18 – Результати візуалізації отриманих даних на сервері

3.1.6 Тест для самоперевірки

1. Команда AT+RST виконує функцію...
   1) закриття з'єднання з віддаленим сервером;
   2) перезавантаження модуля;
   3) перевірки на запуск модуля;
   4) керування повтором прийнятих символів команд.
2. Команда AT+CWJAP виконує функцію...
   1) перезавантаження модуля;
   2) вибору режиму роботи Wi-Fi;

3) перевірки на запуск модуля;
4) підключення до точки доступу під час роботи в режимі робочої станції.

3. Команда AT+CIPCLOSE виконує функцію...
   1) закриття з'єднання з віддаленим сервером;
   2) керування повтором прийнятих символів команд;
   3) перевірки на запуск модуля;
   4) перезавантаження модуля.
4. Команда ATE виконує функцію...
   1) керування повтором прийнятих символів команд;
   2) перевірки на запуск модуля;
   3) перезавантаження модуля;
   4) закриття з'єднання з віддаленим сервером.
5. Команда AT+CWMODE виконує функцію...
   1) закриття з'єднання з віддаленим сервером;
   2) вибір режиму роботи Wi-Fi;
   3) перевірки на запуск модуля;
   4) керування повтором прийнятих символів команд.
6. Команда AT виконує функцію...
   1) вибір режиму роботи Wi-Fi;
   2) перезавантаження модуля;
   3) перевірки на запуск модуля;
   4) керування повтором прийнятих символів команд.

Відповіді на тест

1	2	3	4	5	6
2)	4)	1)	1)	2)	3)

3.2 Керування сервоприводом, використовуючи технологію Wi-Fi

У даному розділі розглянемо приклад керування сервомотором за допомогою веб-браузера, використовуючи Arduino і модуль Wi-Fi ESP8266 [29]. ESP8266 встановить з'єднання між сервомеханізмом і веб-браузером через IP-адресу, а потім, переміщаючи слайдер на веб-сторінці, сервопривод виконуватиме наші команди. Використовуючи матеріал цього прикладу, можна керувати, наприклад, сервоприводом жалюзі на вікнах будинку з будь-якої точки світу через Інтернет.

3.2.1 Створення WEB-сторінки

Для розробки WEB-сторінки, по-перше, необхідно створити порожній файл «servo.html» та додати в нього такий текст:

```
<!DOCTYPE html >
<html>
<head>
<title>Урок</title>
<script src="jquery.js"></script>
<script src="rangeslider.js"></script>
<link rel="stylesheet"  href="rangeslider.css">
</head>

<body>
    <center>
<h1 style="color: blue"> <b> Керування сервомотором за допомогою модуля Wi-Fi  </b> </h1>
    </center>
<h4> <b> Введіть IP адресу модуля, скопіювавши її з Монітору порту </b> </h4>
<div style="margin: 0; width:500px; height:80px;">
    <FORM NAME="form" ACTION="" METHOD="GET">
        IP-адреса:
        <INPUT TYPE="text" NAME="inputbox" VALUE="IP">
    </FORM>
</div>
<h3> Для запуску сервоприводу покрутіть повзунок </h3>
<input type="range" min="20" max="170" step="1" value="95"  />
<p id="servoangle"></p>
</body>

<script>
    $('input[type="range"]').rangeslider({
        polyfill: false,

        rangeClass: 'rangeslider',
        disabledClass: 'rangeslider--disabled',
        horizontalClass: 'rangeslider--horizontal',
        verticalClass: 'rangeslider--vertical',
        fillClass: 'rangeslider__fill',
        handleClass: 'rangeslider__handle',

        // Callback function
        onInit: function() {},
```

```
        // Callback function
        onSlide: function(position, value) {
           // alert('ok');
           servo1(this.value);
        },

        // Callback function
        onSlideEnd: function(position, value) {}
    });

$.ajaxSetup({timeout:1000});

function servo1(angle)
{
    $('#servoangle').text('Angle = ' + angle);
    TextVar = $('input[name=inputbox]').val();
    ArduinoVar = "http://" + TextVar + ":80";
    $.get( ArduinoVar, { "sr1": angle })    ;
    {Connection: close};
}
</script>
</html>
```

Далі помістимо файли jQuery і rangeslider (два файли) в ту саму папку, де й файл servo.html. jQuery – це бібліотека Java Script, яка має багато функцій JS і здійснює маніпуляції з DOM (Document Object Model), обробку подій і використання Ajax для виконання дій на сайті без оновлення веб-сторінки. Rangeslider – бібліотека для повзунка і його графічного зовнішнього вигляду. Завантажити бібліотеки можна за посиланням: https://rangeslider.js.org/ , або https://api.jquerymobile.com/rangeslider/

3.2.2 Створення програми для Arduino

Далі слід завантажити код в Arduino:

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial esp(2, 3);
#define DEBUG true // Відображення повідомлення ESP8266 на моніторі
#define servopin 9
Servo ser;
int current_pos = 170;
int v = 10;
```

```
int minpos = 20;
int maxpos = 160;
void setup()
{
  ser.attach(servopin);
  ser.write(maxpos);
  ser.detach();
  Serial.begin(115200);
  esp.begin(115200);
  sendData("AT+RST\r\n", 2000, DEBUG); //Рестарт модуля
  sendData("AT+CWMODE=1\r\n", 1000, DEBUG); //Встановити модуль в режимі клієнта
  sendData("AT+CWJAP=\"WI-FI\",\"PASSWORD\"\r\n", 2000, DEBUG); //Підключення до Wi-Fi мережі
  while(!esp.find("OK")) {  //Очікування підключення
  }
  sendData("AT+CIFSR\r\n", 1000, DEBUG);           //Виведення IP-адреси на Монітор порту
  sendData("AT+CIPMUX=1\r\n", 1000, DEBUG);         //Використання декількох з'єднань
  sendData("AT+CIPSERVER=1,80\r\n", 1000, DEBUG); //Запуск веб-сторінки на порту 80
}

void loop()
{
  if (esp.available())     //Перевірка даних на ESP8266
  {
    if (esp.find("+IPD,")) //Якщо є нова команда
    {
      String msg;
      esp.find("?");//Поки не буде знайдена команда
      msg = esp.readStringUntil(' '); //Читання повідомлення
      String command = msg.substring(0, 3);          //Команда повідомляється в перших 3-х символах "sr1"
      String valueStr = msg.substring(4);      //Наступні 3 символи повідомляють необхідний кут
      int value = valueStr.toInt();  //Перетворити в ціле число
      if (DEBUG) {
        Serial.println(command);
        Serial.println(value);
      }
      delay(100);
```

```
      //Повернути сервопривод в потрібний кут
      if(command == "sr1") {
         //limit input angle
         if (value >= maxpos) {
           value = maxpos;
         }
         if (value <= minpos) {
           value = minpos;
         }
         ser.attach(servopin); //Підключити сервопривод
         while(current_pos != value) {
           if (current_pos > value) {
             current_pos -= 1;
             ser.write(current_pos);
             delay(100/v);
           }
           if (current_pos < value) {
             current_pos += 1;
             ser.write(current_pos);
             delay(100/v);
           }
         }
         ser.detach();
      }
  }
  }
  }
String sendData(String command, const int timeout, boolean debug)
{
  String response = "";
  esp.print(command);
  long int time = millis();
  while ( (time + timeout) > millis())
  {
    while (esp.available())
    {
      char c = esp.read();
      response += c;
    }
  }
  if (debug)
  {
    Serial.print(response);
```

```
  }
  return response;
}
```

У рядку:

```
sendData("AT+CWJAP=\"WI-FI\",\"PASSWORD\"\r\n", 2000, DEBUG);
```

необхідно замінити ім'я Wi-Fi і пароль на власні.

Відкриємо монітор порту та виставимо швидкість обміну даними: 115200 бод. На екрані монітора має бути відображена інформація про модуль. Серед виведеного тексту потрібно знайти IP-адресу Wi-Fi-модуля та скопіювати її в буфер обміну (рис. 3.19).

Рисунок 3.19 – Потрібна адреса Wi-Fi-модуля

Наступним кроком необхідно завантажити файл servo.html, та в поле IP-адреси вставити з буфера обміну скопійовану адресу (рис. 3.20).

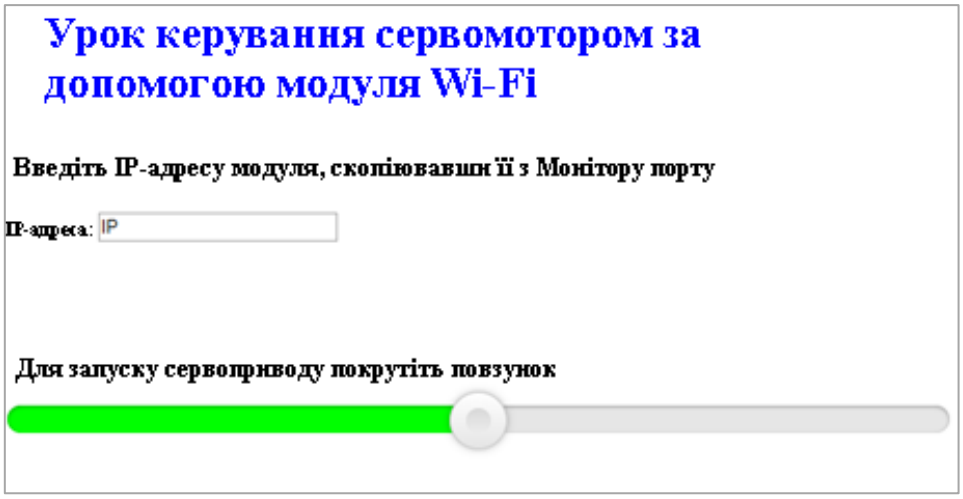


Рисунок 3.20 – Інтерфейс сторінки servo.html

Тепер, коли буде переміщено повзунок, сервомотор рухатиметься відповідно слайдера.

3.2.3 Тест для самоперевірки

1. AT-команда перевірки запуску:
   1) AT+RST;
   2) AT+GSLP ;

3) AT+UART;
4) AT.

2. AT-команда перезавантаження:
   1) AT+RST;
   2) AT+CWLAP;
   3) AT+UART;
   4) AT+CWJAP.
3. Команда запиту інформації про налаштований режим роботи Wi-Fi:
   1) AT+RST;
   2) AT+SLEEP;
   3) AT+CWMODE;
   4) AT+CIPAP.
4. Команда підключення до точки доступу:
   1) AT+CWLAP;
   2) AT+SLEEP;
   3) AT+CWMODE;
   4) AT+CWJAP.
5. Команда отримання локальної адреси IP:
   1. AT+CIPMUX;
   2. AT+CIFSR;
   3. AT+CWMODE;
   4. AT+CWJAP.
6. Команда дозволу режиму множинних з'єднань:
   1) AT+CIPMUX;
   2) AT+CIFSR;
   3) AT+CWMODE;
   4) AT+CWJAP.

Відповіді на тест

1	2	3	4	5	6
4)	1)	4)	4)	2)	1)

3.3 Підключення Ethernet модуля до Arduino

Головна функція цифрової метеостанції – вимір температури. Відповідно, коли це передбачено конструкцією, вимір можна робити всередині і/або зовні

приміщення. Діапазон вимірюваних температур метеостанцій варіюється зазвичай від +50 до -70 градусів Цельсія.

3.3.1 Загальні відомості про мережу Ethernet

Найпоширенішим стандартом, який використовується в провідних локальних і глобальних мережах, є Ethernet. Це – ціле сімейство технологій пакетної передачі даних для комп'ютерних мереж. Назва «Ethernet» (у перекладі «ефірна мережа») відображає початковий принцип роботи цієї технології: вся інформація, яка передана одним вузлом, водночас приймається всіма іншими (тобто є певна схожість з радіомовленням). Зараз практично завжди підключення пристроїв до мережі відбувається через комутатори (switch), так що кадри, які відправляються одним вузлом, доходять лише до адресата (виняток становлять передачі на широкомовну адресу), що підвищує швидкість роботи і безпеку мережі.

Кожен мережний пристрій володіє унікальною фізичною адресою, яка називається MAC-адреса і складається з шести байтів виду, наприклад, 0F-1E-2D-3C-4B-5A. Теоретично фізична адреса пристрою не має збігатися з адресою жодного з мережних пристроїв у всьому світі.

Крім того, кожен пристрій, підключений до мережі, має унікальну для цієї мережі мережну адресу, яка складається з чотирьох байтів (для стандарту IP4), наприклад, 192.168.0.1. Ця електронна адреса називається IP-адресою. Для однозначної ідентифікації, яка надійшла мережею інформації до IP-адреси, додається так званий «порт», кожен з яких обслуговується відповідним застосунком, запущеним на мережному пристрої. Тоді мережна адреса пристрою розширюється і виглядає, наприклад, так: 192.168.0.1:80.

Усередині мережі існує два види мережних пристроїв: сервери і клієнти. Сервери обслуговують запити від клієнтів, що надходять до них, і формують відповіді. Клієнти ж, у свою чергу, такі запити формують і користуються відповідями сервера. Реалізацією даної взаємодії на мережних пристроях займаються спеціальні програми, також звані серверами і клієнтами.

3.3.2 Мережний модуль Ethernet

Для підключення Arduino до локальної мережі у прикладі будемо використовувати Ethernet мережний модуль, який можна бачити на рис. 3.21. Цей модуль заснований на чіпі – мережному контролері W5100, який має внутрішній буфер на 16 Кбайт [31, 32]. Це означає, що в своїй оперативній пам'яті він може зберігати 16 Кбайт інформації для прийому й видачі.

Рисунок 3.21 – Ethernet мережний модуль

Модуль призначений для реалізації мережних застосунків: віддаленого керування системою, мережного доступу, публікації даних [32].

Для використання мережного модуля Ethernet його потрібно підключити до Arduino через інтерфейс, розташований на платі. Далі потрібно створити і записати у пам'ять контролера програму керування проєктом.

Коли на плату буде подано живлення, має увімкнутися світлодіод індикації живлення. Після вищеописаних дій до інтерфейсу мережного модуля можна підключати мережний кабель. Якщо зв'язок на інтерфейсі нормальний, мають увімкнутися світлодіоди індикації прийому–відправки пакетів (прийому – зелений, відправки – помаранчевий).

Мережний модуль Ethernet Arduino має два інтерфейси – SPI-інтерфейс для підключення Arduino контролера або іншого мікропроцесорного керуючого пристрою та інтерфейс RJ45 для підключення мережного кабелю [31].

SPI-інтерфейс для підключення Arduino контролера має 10 виводів, які позначені CLKOUT, WOL, SI, CS, VCC, INT, SO, SCK, RESET, GND.

Інтерфейс RJ45 має 8 виводів для підключення мережного кабелю. Максимальна швидкість передачі даних 100 Мбіт / с. Безпосередньо сам роз'єм кріпиться до плати за допомогою спеціальних пластикових штирів-фіксаторів.

Живлення мережного модуля здійснюється від Arduino контролера або іншого мікропроцесорного керуючого пристрою, а також може здійснюватися від зовнішніх джерел живлення (блоків живлення, батарей). Виводи живлення VCC і GND.

Характеристики модуля:

– вбудований контролер для перетворення даних у стандарт Ethernet-Microchip ENC28J60;

– SPI-інтерфейс з виводами CLKOUT, WOL, SI, CS, VCC, INT, SO, SCK, RESET, GND;

– інтерфейс RJ45 зі швидкістю передачі даних до 100 Мбіт / с;

– напруга живлення: 3,3 В.

3.3.3 Практична реалізація

Підключення мережного модуля Ethernet до Arduino відбувається за допомого інтерфейсу SPI.

На рисунку 3.22 показано приклад виведення пінів інтерфейсу SPI Arduino Uno.

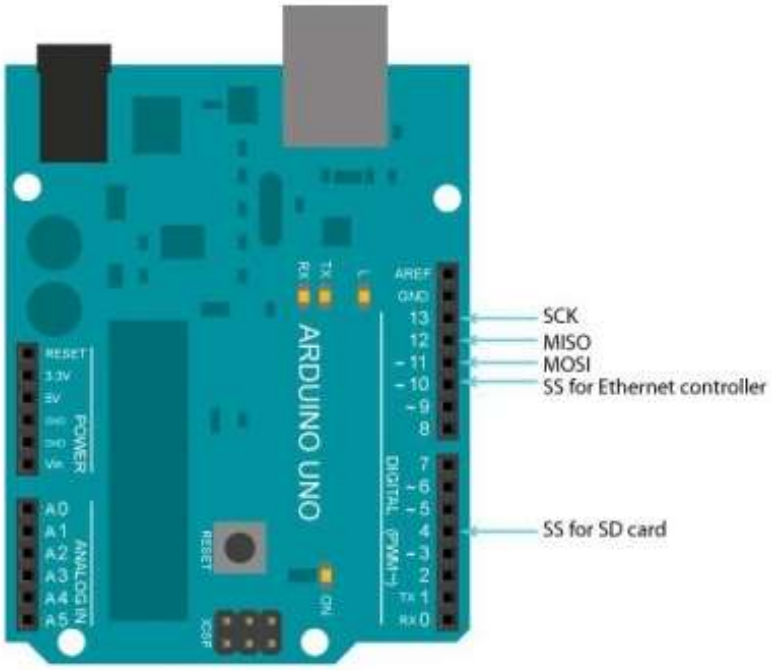


Рисунок 3.22 – Приклад виведення пінів інтерфейсу SPI Arduino Uno

Послідовний периферійний інтерфейс (SPI) – це синхронний протокол послідовної передачі даних, який використовується для зв'язку мікроконтролера з одним або декількома периферійними пристроями. Інтерфейс SPI відрізняється відносно високою швидкістю і призначений для зв'язку близько розташованих пристроїв. Він також може використовуватися для взаємодії двох мікроконтролерів.

Згідно з протоколом SPI, один із взаємодіючих пристроїв (зазвичай мікроконтролер) завжди є провідним і контролює ведені периферійні пристрої. Як правило, всі взаємодіючі пристрої об'єднані трьома загальними лініями:

– MISO (Master In Slave Out) – лінія для передачі даних від провідного пристрою (Slave) до ведучого (Master);

– MOSI (Master Out Slave In) – лінія для передачі даних від ведучого пристрою (Master) до провідного (Slave);

– SCK (Serial Clock) – тактові імпульси, які генеруються провідним пристроєм (Master) для синхронізації процесу передачі даних.

Крім перерахованих, на кожен пристрій відводиться окрема лінія:

– SS (Slave Select) – вивід, який присутній на кожному провідному пристрої. Він призначений для активізації Майстром того чи іншого периферійного пристрою.

Схема підключення модуля Ethernet до Arduino показана на рисунку 3.23.

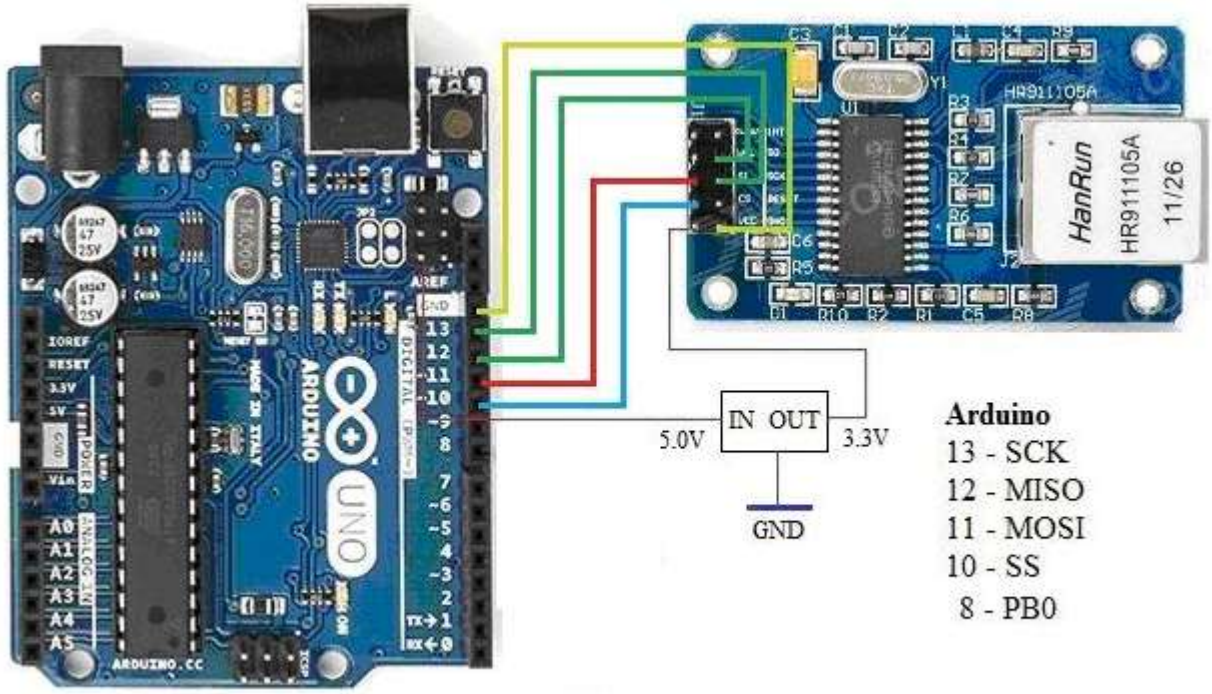


Рисунок 3.23 – Схема підключення модуля Ethernet до Arduino

Необхідно звернути увагу, що 10 контактний модуль Ethernet ENC28J60 Enternet HR911105A живиться від роз'єму 3,3 В, а 12 контактний модуль від 5 В. Крім того, позначення роз'ємів на HanRun HR911105A можуть бути змінені виробником, наприклад, ST – замість SO.

Після складання електричної схеми необхідно завантажити бібліотеку для роботи з HanRun HR911105A. Бібліотеки слугують для полегшення коду. Це можуть бути драйвери до додаткового обладнання або часто використовувані функції. У програмі Arduino IDE вже є набір стандартних бібліотек, які часто використовуються. В цьому випадку потрібно завантажити нові бібліотеки «Ethercard».

Лістинг програми:

```
#include <SPI.h>
#include <Ethernet.h>
#include <math.h>

byte mac[]     = { 0x90, 0xA2, 0xDA, 0x0D, 0x83, 0x9D };
IPAddress  ip(192,  168,  0,  101);//Вкажіть  вашу  локальну  IP-
адресу!!!!!
EthernetServer server(80);

// Функція переведення в Градуси
double Thermister(int RawADC) {
  double Temp;
  Temp = log(((10240000/RawADC) - 10000));
  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp
* Temp ))* Temp );
  Temp = Temp - 273.15;   // Кельвін в Градуси
  return Temp;
}

void setup()
{
    //Старт
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());

}
void loop()
{
  EthernetClient client = server.available();
  if (client) {//Якщо клієнт підключений
    Serial.println("Connected to your website...");
    while(client.connected()) {
      if(client.available()) {
        client.print("t=");
        client.print(float(Thermister(analogRead(0))), 1);
        client.print(" C");
        client.println();
        client.stop();
        client.flush();
```

```
        }
      }
      client.stop();
      Serial.println("Client disconnected");
    }
}
```

Результат виконання програми показано на рисунку 3.24.

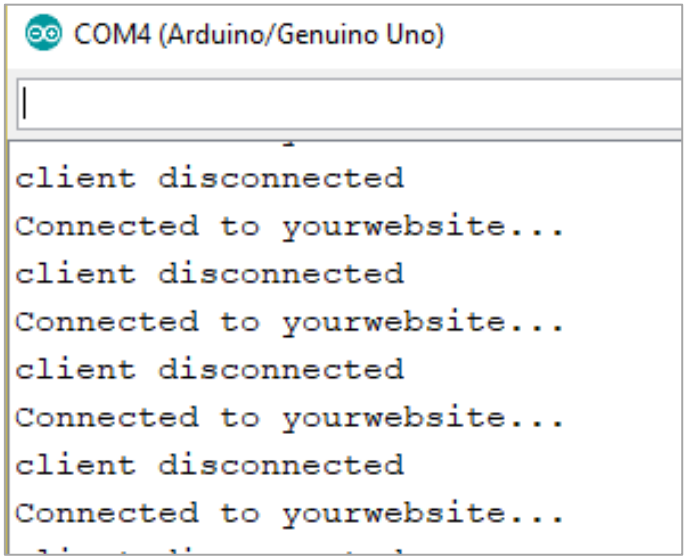


Рисунок 3.24 – Результат виконання програми

Після відкриття в браузері сторінки за адресою «192.168.0.101» ми маємо побачити виміряне значення температури.

3.3.4 Тест для самоперевірки

1. Ethernet – це…
   1) сімейство технологій для пошуку даних;
   2) технології для відправки даних за допомогою комп'ютерних мереж;
   3) технології для прийому даних за допомогою комп'ютерних мереж;
   4) сімейство технологій пакетної передачі даних для комп'ютерних мереж.
2. Скільки байт займає MAC-адреса?
   1) 2;
   2) 4;
   3) 6;
   4) 8.

3. Скільки байт займає мережна адреса?
   1) 2;
   2) 4;
   3) 8;
   4) 16.
4. Адреса виду 192.168.0.1 називається…
   1) IP-адресою;
   2) MAC-адресою;
   3) TCP-адресою;
   4) PIP-адресою.
5. Сервери обслуговують...
   1) запити, які надходять до них від інших серверів;
   2) формування запиту і відправлення клієнтові;
   3) запити, які надходять до них від клієнтів і формують відповіді;
   4) формування запиту для інших серверів.
6. Чому дорівнює об'єм внутрішнього буфера мережного контролера W5100?
   1) 2 Кбайт;
   2) 4 Кбайт;
   3) 8 Кбайт;
   4) 16 Кбайт.

Відповіді на тест

1	2	3	4	5	6
4)	3)	2)	1)	3)	4)

3.4 Відправка даних до серверу через Ethernet

3.4.1 Опис моделі взаємодії клієнт–сервер

Як правило, комп'ютери і програми, які входять до складу інформаційної системи, не є рівноправними [32]. Деякі з них володіють ресурсами (файлова система, процесор, принтер, база даних тощо), інші мають можливість звертатися до цих ресурсів. Комп'ютер (або програму), що керує ресурсом, називають сервером цього ресурсу (файл-сервер, сервер бази даних, обчислювальний сервер...). Клієнт і сервер будь-якого ресурсу можуть знаходитися як у рамках однієї обчислювальної системи, так і на різних комп'ютерах, пов'язаних мережею.

Основний принцип технології «клієнт–сервер» полягає в поділі функцій застосунку на три групи:

– введення і відображення даних (взаємодія з користувачем);

– прикладні функції, характерні для даної предметної області;

– функції керування ресурсами (файловою системою, базою даних тощо).

Тому в будь-якому застосунку виділяються такі компоненти:

– компонент подання даних;

– прикладний компонент;

– компонент керування ресурсом.

Зв'язок між компонентами здійснюється за певними правилами, які називають «протокол взаємодії».

Історично першою з'явилася модель розподіленого подання даних, яка реалізовувалася на універсальній ЕОМ з підключеними до неї неінтелектуальними терміналами. Керування даними та взаємодія з користувачем при цьому поєднувалися в одній програмі, на термінал передавалася тільки «картинка», сформована на центральному комп'ютері.

Потім, з появою персональних комп'ютерів (ПК) і локальних мереж, були реалізовані моделі доступу до віддаленої бази даних (БД). Деякий час базовою для мереж ПК була архітектура файлового сервера. При цьому один з комп'ютерів є файловим сервером, на клієнтах виконуються застосунки, в яких поєднані компоненти уявлення та прикладний: система керування базами даних (СКБД) і прикладна програма. Протокол обміну при цьому являє собою набір низькорівневих викликів операцій файлової системи. Така архітектура реалізована, як правило, за допомогою персональних, має очевидні недоліки – високий мережний трафік і відсутність уніфікованого доступу до ресурсів.

З появою перших спеціалізованих серверів баз даних з'явилася можливість іншої реалізації моделі доступу до віддаленої бази даних. У цьому випадку ядро СУБД функціонує на сервері, протокол обміну забезпечується за допомогою мови SQL. Такий підхід порівняно з файловим сервером веде до зменшення завантаження мережі й уніфікації інтерфейсу «клієнт–сервер» [27]. Втім мережний трафік залишається досить високим, крім того, як і раніше неможливо задовільний адміністрування застосунків, оскільки в одній програмі поєднуються різні функції.

Пізніше була розроблена концепція активного сервера (рис. 3.25), який використовував механізм збережених процедур. Це дозволило частину прикладного компонента перенести на сервер (модель розподіленого

застосунку). Процедури зберігаються в словнику бази даних, розподіляються між декількома клієнтами і виконуються на тому самому комп'ютері, що і SQL-сервер. Переваги такого підходу: можливе централізоване адміністрування прикладних функцій, значно знижується мережний трафік (тому що передаються не SQL-запити, а виклики збережених процедур). Недолік – обмеженість коштів розробки збережених процедур порівняно з мовами загального призначення.

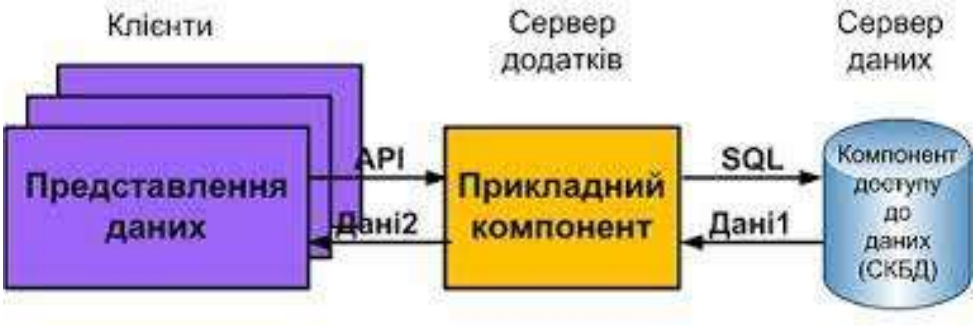


Рисунок 3.25 – Архітектура «клієнт–сервер»

На практиці зазвичай використовується змішаний підхід:

– найпростіші прикладні функції виконуються збереженими процедурами на сервері;

– більш складні реалізуються на клієнті безпосередньо в прикладній програмі.

Зараз ряд постачальників комерційних СУБД оголосило про плани реалізації механізмів виконання збережених процедур з використанням мови Java. Це відповідає концепції «тонкого клієнта», функцією якого залишається тільки відображення даних (модель віддаленого подання даних).

3.4.2 Бібліотека Ethernet.h

Ethernet дуже широко використовується в Arduino-проєктах у всьому світі, і тому автори Arduino IDE вважали за необхідне включити бібліотеки для роботи з мережею безпосередньо до складу середовища розробки.

Ця бібліотека містить три основні класи – Ethernet, EthernetServer, EthernetClient. При цьому слід зауважити, що клас Ethernet дозволяє використовувати свої функції без створення відповідного об'єкта, в той час як класи Server і Client вимагають попереднього створення об'єкта – сервера або клієнта.

Клас Ethernet

Функція void begin (unsigned int[] mac, unsigned int[] ip, unsigned int[] gateway, unsigned int[] mask) приймає від двох (одного, в разі використання адреси, яка автоматично присвоюється маршрутизатором) до чотирьох параметрів і призначена для ініціалізації та запуску Ethernet-шілда. Всі параметри передаються у функцію як масиви і являють собою (за порядком передачі параметрів у функцію) MAC-адресу (масив з 6 байтів), IP-адресу (масив з 4 байтів), необов'язкову адресу шлюзу (масив з 4 байтів, за замовчуванням такий самий, як IP-адреса, тільки з 1 в останньому байті) і маску підмережі (масив з 4 байтів, за замовчуванням – 255, 255, 255, 0).

Викликається функція begin класу Ethernet так:

```
#include <Ethernet.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 0, 101);
void setup()
{
  Ethernet.begin(mac, ip);
}
```

Функція int[] localIP() дозволяє дізнатися поточну IP-адресу, присвоєну шілду, у вигляді масиву з 4-х байтів. Особливо корисна ця функція, якщо шілд автоматично отримує IP-адресу від маршрутизатора (технологія DHCP).

Клас EthernetServer і серверні функції

Як вже було сказано, даний клас вимагає обов'язкового створення свого об'єкта. Робота з класом починається з використання конструктора.

EthernetServer EthernetServer(unsigned int port) – функція-конструктор сервера. Приймає один параметр – номер порту, за яким сервер прийматиме запити. Функція повертає об'єкт типу Server, для якого в подальшому можна використовувати методи сервера.

void begin() – запускає раніше створений об'єкт сервера на прийом вхідних з'єднань, після чого він починає приймати запити на відповідному порту. Слід зазначити, що запускати сервер можна тільки після запуску самого Ethernet-шілда.

Як приклад наведемо короткий фрагмент коду:

```
EthernetServer server(80);
void setup()
```

```
{
  Ethernet.begin(mac, ip);
  server.begin();
}
```

EthernetClient available() – функція, яка перевіряє наявність запиту на підключення до створеного нами сервера і повертає об'єкт типу Client, з якого в подальшому можна зчитувати дані запиту.

Застосувати функцію можна, наприклад, так:

```
void loop()
{
  EthernetClient client = server.available();
if (client) {
    server.write(client.read());
}
}
```

З'єднання з клієнтом підтримується постійно – навіть після виходу з блоку коду, де був побудований об'єкт Client. Щоб відхилити з'єднання, необхідно використовувати функцію EthernetClient.stop().

Для передачі даних підключеному клієнту сервер підтримує кілька функцій:

– void write(unsigned int) – передає клієнту один байт;

– void write(char[]) – передає клієнту масив символів (рядок);

– void write(unsigned int[], unsigned int) – передає клієнту масив символів із зазначенням його довжини.

Крім того, сервер підтримує більш звичні нам функції відправки даних – void print() і void println(). Працюють функції так само, як, наприклад, функції виведення інформації в послідовний порт, і можуть передавати числа, символи, рядки та іншу інформацію як послідовність символів. Наприклад, число 123 буде передано як три символи: '1', '2', '3'.

Клас EthernetClient і клієнтські функції

EthernetClient (unsigned int[]) і EthernetClient (unsigned int[], unsigned int) – функції-конструктори, які повертають об'єкт класу Client, для підключення до заданого сервера. Приймає функція два параметри, перший з яких – масив байтів IP-адреси сервера, до якого потрібно підключитися, другий, необов'язковий, – номер порту для підключення.

int connect() – функція підключає створений клієнт до IP-адреси і порту, заданому через нього конструктор. Функція повертає параметр, який вказує на факт підключення. Значення, яке повертається 1, вказує на успішне підключення, а від'ємні значення (-1, -2 і т.д.) – на виникнення під час підключення тих чи інших помилок. Наведений нижче приклад, коли здійснюється спроба підключення до сервера (як такий сервер цілком можна використовувати один з серверів Google з IP-адресою 64.233.161.101) і з успішним підключенням відправляє серверу дані з пошуковим запитом:

```
EthernetClient client(server, 80);
void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
  delay(1000);
  Serial.println("Connecting...");
  if  (client.connect())
{
    Serial.println("Connected");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  } else
 {
    Serial.println("Connection failed");   }   }
```

boolean connected() – визначає, чи підключений клієнт до віддаленого сервера. Значення, яке повертається, true, вказує на наявність підключення, false – на його відсутність. Функцію корисно використовувати перед кожною спробою передачі даних на сервер, адже він міг відключити наш клієнт з тієї чи іншої причини.

Для відправки даних сервера клієнт також використовує функції write, print і println. Їх використання не відрізняється від стандартного.

unsigned int available() – ця функція повертає кількість байтів даних, надісланих віддаленим сервером нашого клієнта. Якщо значення, яке повертається функцією, більше 0, то в буфері прийому ще є необроблені дані від сервера. Використання функції повністю аналогічно використанню однойменної функції послідовного порту Serial.

int read() – функція читає наступний байт даних з буфера прийому. Якщо в буфері даних немає, функція поверне значення -1. Щоб за необхідності

очистити буфер прийому до читання всіх даних, які знаходяться в ньому, можна скористатися функцією void flush().

3.4.3 Основи HTML

У цьому прикладі плата Arduino працюватиме як веб-сервер і повертатиме клієнту-браузеру на комп'ютері текст веб-сторінки. Для того щоб хоча б приблизно уявляти собі, що таке веб-сторінка, стисло розглянемо основи мови гіпертекстової розмітки HTML.

Будь-яка веб-сторінка являє собою звичайний текстовий файл, в якому, з використанням спеціальної мови зазначено, які елементи матиме веб-сторінка, де вони розташовуватимуться і як виглядатимуть. Крім того, більш складні веб-сторінки можуть містити в собі програми-скрипти мовами JavaScript, Perl, PHP, а зовнішній їх вигляд може описуватися в інших файлах за допомогою мови CSS. У нашому уроці ми будемо використовувати найпростіші HTML-сторінки без скриптів і будь-якого оформлення і тому розглянемо зараз лише найпростіші HTML-конструкції.

Простіше кажучи, HTML не є мовою програмування, вона є мовою розмітки гіпертекстових документів. Іншими словами, вона відповідає за розташування в документі текстів, посилань, рисунків, таблиць та інших елементів, призначених для життя в мережі Інтернет. Змусити цю мову щось порахувати неможливо, в ній немає логічних функцій, проте вона у змозі красиво й легко показати будь-яку інформацію. Читається ця мова за допомогою програм, так званих браузерами, які розуміють стандартні команди HTML-мови і, обробляючи їх, виводять на монітор комп'ютера документи в тому вигляді, в якому хоче подати їх розробник документа.

Команди мови HTML називають дескрипторами або, навіть частіше, тегами.

Простіше за все розглядати теги мови на простому прикладі. Припустимо, що ми підготували на комп'ютері текстовий документ з таким змістом:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>HTML-сторінка</title>
    </head>
<body>
    <p>Привіт, світ!!!</p>
    <br>
```

```
    <p>Це моя перша HTML-сторінка!</p>
</body>
</html>
```

Те, що вміщено між знаками <> – і є теги. Теги невидимі читачеві, але видні браузеру, і кожен з них дає браузеру певну команду.

Найперший тег <! DOCTYPE HTML> повідомляє браузеру, що йому передана HTML-сторінка і обробляти її потрібно відповідним чином.

<html> – завжди починає веб-сторінку.

<head> – відкриває «шапку» сторінки і слугує для початку блоку, в якому розміщуються різні службові дані, які необхідні браузеру для аналізу сторінки. Наприклад, там знаходяться теги <title> і </ title>, всередині яких міститься рядок, який браузер відображає на вкладці поточної сторінки і в заголовку свого вікна.

Теги <body> обрамляють основний блок сторінки і саме в них розташовується весь корисний вміст сторінки.

Теги <html>, <head> і <body> мають бути присутніми в кожній правильній веб-сторінці і саме з них складається найпростіший її шаблон.

Також у нашій сторінці присутні теги <p>, призначені для розміщення на сторінці параграфів з текстом, а також тег
 вставляють у сторінку перенесення рядка.

Як видно, в нашій сторінці є як парні теги – один тег без косої риски (відкриває), другий – з косою рискою (закриває), так і поодинокі. Парні теги формують на сторінці блоки і дозволяють вибудувати необхідну структуру сторінки. Використання парних тегів виглядає так:

```
<Тег «Зовнішній блок»>
     <Тег «Внутрішній блок»>
          <Тег «Ще більш внутрішній блок»>
               зміст
          </ Тег «Ще більш внутрішній блок>
     </ Тег «Внутрішній блок»>
</ Тег «Зовнішній блок»>
```

Якщо спробувати завантажити в браузер створену нами сторінку, він відобразить два параграфи з текстом, розділених переведенням рядка, а заголовок сторінки прийме заданий сторінкою вигляд.

Цього введення в HTML буде цілком достатньо, щоб ми змогли сформувати нашим Arduino-сервером просту сторінку і відправити її у браузер.

При цьому слід зазначити, що, крім самого тексту сторінки, під час запиту сервер повертає так званий «HTTP-заголовок», який виглядає так:

```
HTTP / 1.1 200 OK
Content-Type: text / html
Connection: close
Refresh: 5
Заголовок відповіді сервера
```

3.4.4 Практична реалізація

У даному прикладі Arduino виступає в ролі сервера, який очікує вхідні з'єднання через вказаний порт, найчастіше це 80 порт. Розглянемо приклад, коли як клієнт виступатиме ПК, а як сервер Arduino [32]. Відправимо запит до Arduino, і після того, як він його обробить, згенерує для нас HTML-сторінку.

Спочатку необхідно підключити Ethernet до Arduino, до модуля підключити мережний кабель, а також скласти найпростішу демонстраційну схему – змінний резистор підключити до входу A0 плати (рис. 3.26).

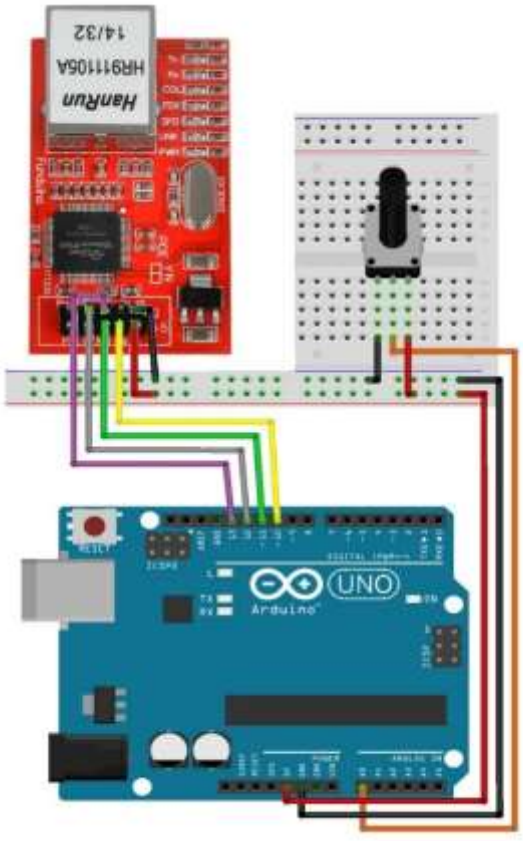


Рисунок 3.26 – Схема підключення компонентів до Arduino

Другим кроком необхідно з'ясувати структуру нашої мережі, щоб правильно поставити IP-адресу модуля. Для цього найпростіше вивчити

властивості проводового мережного підключення на будь-якому комп'ютері, увімкненому в локальну мережу (рис. 3.27).

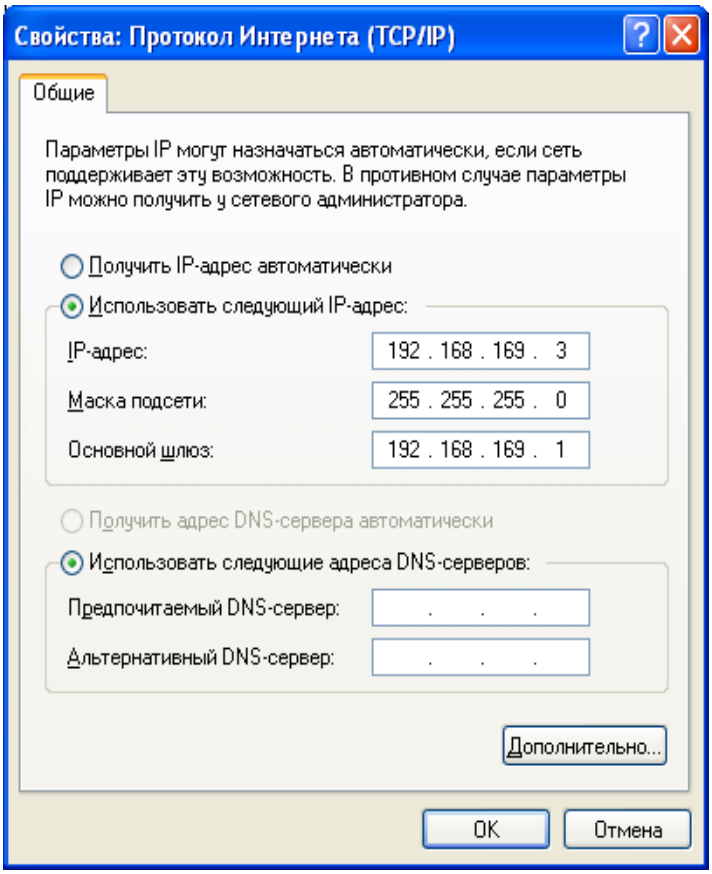


Рисунок 3.27 – Відомості про мережне з’єднання в Windows

У прикладі IP-адреса мережі має вигляд 192.168.169.X, а отже, адреса модуля матиме вигляд, наприклад, 192.168.169.100. Звичайно ж, існує небезпека, що ця адреса вже зайнята будь-яким пристроєм у мережі, якщо таких пристроїв багато. Тоді доведеться обрати іншу адресу або звернутися до адміністратора, щоб з'ясувати вільні адреси.

Напишемо код своєї сторінки, на якій міститиметься інформація про кількість вільної оперативної пам'яті (RAM) Ардуіно:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Перший проект</title>
    </head>
<body>
    <p>Вільно пам’яті: </p>
</body>
</html>
```

У разі успішного виконання запиту клієнту надсилається вказаний вище заголовок:

```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // MAC-адреса
IPAddress ip(192, 168, 3, 20);//Вкажіть вашу локальну IP-адресу
EthernetServer server(80);
void setup()
{ //Старт
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
}
void loop()
{
  EthernetClient client = server.available();
  if (client)
  {
    // Перевіряємо чи підключений клієнт до сервера
    while (client.connected())
    {
      if (client.available()) {
        // Виводимо HTML - сторінку
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.println("<!DOCTYPE html>");
        client.println("<html>");
        client.println("<head>");
        client.println("<meta charset=\"UTF-8\">");
        client.println("<title>Home</title>");
        client.println("</head>");
        client.println("<body>");
        client.println("<h1>Home Server</h1>");
        client.println("<p>Вільно пам’яті: ");
        client.println(freeRam());
        Serial.println("FREE RAM: ");
        Serial.println(freeRam());
        client.println("</body>");
        client.println("</html>");
        client.stop();
        client.flush();
```

```
          }
        }
        client.stop();
        Serial.println("Client disconnected");
      }
    }
    int freeRam () {
      extern int __heap_start, *__brkval;  int v;  return (int) &v -
(__brkval == 0 ? (int) &__heap_start : (int) __brkval);
    }
```

Після завантаження скетчу в Arduino в адресному рядку браузера можна набрати IP-адресу нашої Arduino і спостерігати веб-сторінку, яка формується сервером (рис. 3.28).

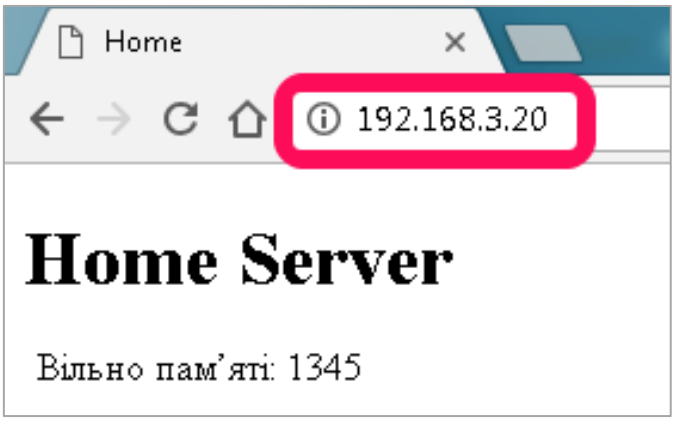


Рисунок 3.28 – Сторінка, створена сервером

Розглянемо приклад виведення на веб-сторінку інформації про значення температури і вологості, використовуючи датчик DHT11, який підключений до 9-го піна Arduino [33]:

```
#include <SPI.h>
#include <Ethernet.h>
#include "DHT.h"
#define DHTPIN 9 // номер піна, до якого приєднаний датчик
DHT dht(DHTPIN, DHT11);

byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // MAC-адреса
IPAddress ip(192, 168, 3, 20);// Вкажіть вашу локальну IP-адресу
EthernetServer server(80);
float h, t;
void setup()
{ //Старт
```

```
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  dht.begin();

}
void loop()
{
  h = dht.readHumidity();
  t = dht.readTemperature();

  EthernetClient client = server.available();
  if (client)
  {
    // Перевіряємо чи підключений клієнт до сервера
    while (client.connected())
    {
      if (client.available()) {

        // Виводимо HTML – сторінку
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();

        client.println("<!DOCTYPE html>");
        client.println("<html>");
        client.println("<head>");
        client.println("<meta charset=\"UTF-8\">");
        client.println("<title>DHT1</title>");
        client.println("</head>");
        client.println("<body>");
        client.println("<h1>Передача даних з Arduino на HTML-
сторінку</h1>");
        client.println("<h2>");
        client.println("<center>");
        client.println ("Вологість: ");
        client.print(h);
        client.print(" %");

        client.println("<hr>");
        client.println("Температура");
        client.print(t);
        client.println(" C");
        client.println("</center>");
```

```
        client.println("</h2>");

        client.println("</body>");
        client.println("</html>");
        client.stop();
        client.flush();
      }
    }
    client.stop();
    Serial.println("Client disconnected");
  }
}
```

В результаті отримаємо такий вигляд WEB-сторінки від серверу (рис. 3.29).

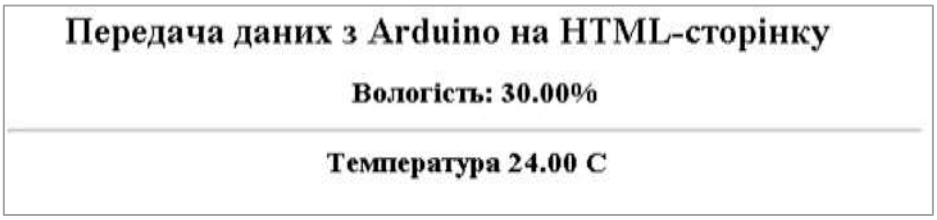


Рисунок 3.29 – Вигляд WEB-сторінки від серверу

3.4.5 Тест для самоперевірки

1. Тег <p> призначений для...
   1) перенесення тексту на новий рядок;
   2) підведення риски під текстом;
   3) для розміщення на сторінці параграфів і абзаців з тестом;
   4) форматування тексту за шириною.
2. Функція void begin класу Ethernet призначена для...
   1) ініціалізації і запуску Arduino;
   2) ініціалізації і запуску Ethernet-шілда;
   3) ініціалізації і запуску серверу;
   4) ініціалізації і запуску передачі даних клієнту.
3. Функції void write призначена для...
   1) передачі даних підключеному клієнту;
   2) передачі даних підключеному серверу;
   3) ініціалізації і запуску серверу;
   4) ініціалізації і запуску клієнта.

4. void print() і void println() – це функції…
    1) функції читання;
    2) формування запиту;
    3) відправки даних;
    4) створення форм.
5. Функція int connect()…
    1) визначає, чи підключений клієнт до віддаленого серверу;
    2) підключає створений клієнт до IP-адреси і порту;
    3) підключає створений клієнт до MAC-адреси і порту;
    4) передає дані підключеному клієнту.
6. Функція boolean connected()…
    1) визначає, чи підключений клієнт до віддаленого серверу;
    2) передає дані підключеному клієнту;
    3) підключає створений клієнт до IP-адреси і порту;
    4) підключає створений клієнт до MAC-адреси і порту.

Відповіді на тест

1	2	3	4	5	6
3)	2)	1)	3)	2)	1)

3.5 Керування пристроями через Ethernet

3.5.1 Загальні відомості про керуючі сигнали

Під час використання цифрового сигналу сенсор у будь-який момент часу видає на сигнальний провід або 0 В, або напругу свого живлення – 5 В. Проміжних значень немає. Для того щоб абстрагуватися від конкретних значень напруги, які не важливі в процесі обробки цифрових сигналів, існують поняття логічного нуля (LOW) і логічної одиниці (HIGH). 0 В – це логічний нуль, напруга живлення – це логічна одиниця.

На Arduino Uno є 14 цифрових входів, кожен з яких може бути використаний для підключення такого датчика.

Є прості сенсори, у яких лише два стани: чорний/білий, ліво/право і т.д. Їх дуже легко підключити і зчитувати показники: сенсори передають свій сигнал безперервно, а значення на сигнальному проводі безпосередньо відповідає їх показникам. Такий простий протокол називається бінарним.

Існують також сенсори з цифровим сигналом, які вимірюють безліч градацій певної фізичної величини на зразок відстані або температури. Але для передачі своїх даних з використанням лише двох значень кожен

такий сенсор визначає власний протокол. У ньому описується, які послідовності нулів і одиниць, з якими затримками, як несуть у собі дані, що передаються. Приймаюча сторона, така як Arduino, має реалізувати алгоритм, який буде зчитувати показання відповідно до протоколу. Протокол у кожного сенсора свій, він описується в його документації.

Якщо говорити про цифрові сенсори з бінарним протоколом, то зчитувати дані з нього – дуже легко.

Оскільки цифрові контакти можуть бути як входами, так і виходами, то для початку потрібно сконфігурувати контакт, до якого підключений сенсор у режим введення. Це потрібно зробити один раз, тому функція setup – підходяще для цього місце. Для конфігурації режиму використовується стандартна функція pinMode. Так, наприклад, якщо ви підключили сенсор до контакту 9, код конфігурації виглядатиме так:

```
void setup()
{
    pinMode(9, INPUT);
}
```

Потім, щоб зчитати стан у будь-який момент часу, існує стандартна функція digitalRead. Продовжуючи приклад, щоб отримати стан сенсора у змінну value, достатньо виконати:

```
int value = digitalRead(9);
```

Вхідна напруга до 2 В проєктується на цілочисельне значення 0, що відповідає значенню константи LOW; напруга більше 3 В проєктується на цілочисельне значення 1, що відповідає значенню константи HIGH. Напруга від 2 до 3 В проєктується на 0 або 1 випадковим чином, але це не є проблемою, оскільки цифрові сенсори не повинні видавати такий сигнал.

Таким чином, програма, яка раз на секунду зчитує показники цифрового сенсора з двома станами, який підключений до контакту 9, і надсилає їх на комп'ютер, може виглядати так:

```
digitalSensorRead.pde
#define SENSOR_PIN 9
void setup()
{
    pinMode(SENSOR_PIN, INPUT);
Serial.begin(9600);
```

```
}
void loop()
{
    delay(1000);
    int val = digitalRead(SENSOR_PIN);
Serial.println(val);
}
```

3.5.2 Керування цифровими пінами Arduino з HTML-сторінки

Цей приклад багато в чому повторюватиме завдання з попереднього підрозділу. При цьому інформація про сигнал передаватиметься серверу прямо зі сторінки сайту.

Схема поєднання компонентів наведена на рисунку 3.30.

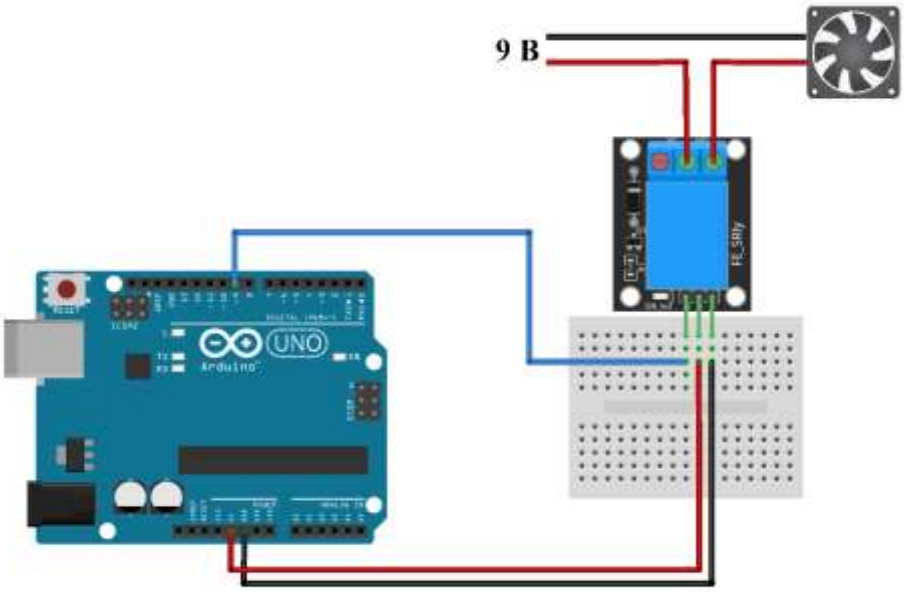


Рисунок 3.30 – Підключення зовнішнього реле

На даній схемі показано тільки підключення зовнішнього реле. Ethernet модуль підключається аналогічно з попереднім прикладом.

Код програми для Arduino:

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // MAC-адреса
IPAddress ip(192, 168, 3, 20);//Вкажіть вашу локальну IP-адресу
EthernetServer server(80);

int rele = 3; // номер піна, до якого приєднане реле
int pinState = 0;
```

```
String getData = "";
boolean startGet = false;
void setup()
{ //Старт
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  pinMode(rele, OUTPUT);
}
void loop()
{
  // очікування підключення клієнтів
  EthernetClient client = server.available();
  if (client)
  {
    boolean currentLineIsBlank = true;
    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
        if (startGet == true) // дані після '?'
          getData += c;
        if (c == '?') // початок збору даних після '?'
          startGet = true;
        if (c == '\n' && currentLineIsBlank) // закінчення отримання
        {
          if (getData.length() < 1) // запит без get-даних
          {
            pinState = 0;
          }
          else
          {
            pinState = int(getData[5]) - 48;
          }
          // відправка заголовків клієнту
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          // формування сторінки відповіді
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
```

```
                client.println("<title>Реле</title>");
                client.println("<meta charset=\"UTF-8\">");
                client.println("<h2>Керування реле</h2>");
                client.println("<form method='get'>");
                client.println("<h3>");

                client.print("<div>");
               client.print("Реле <br>Вимкнути<input type='radio' name='rele'
value=0 onclick='document.getElementById(\"submit\").click();' ");
                if (pinState == 0)
                  client.print("checked");
                client.println(">");
                client.print("<input  type='radio'  name='rele'  value=1
onclick='document.getElementById(\"submit\").click();' ");
                if (pinState == 1)
                  client.print("checked");
                client.println("> Увімкнути");
                client.println("</div>");

                client.println("<input     type='submit'     id='submit'
style='visibility:hidden;' value='Refresh'>");
                client.println("</h3>");
                client.println("</form>");
                client.println("</html>");
                break;
              }
              if (c == '\n')
              {
                currentLineIsBlank = true;
              }
              else if (c != '\r')
              {
                currentLineIsBlank = false;
              }
            }
          }
        }
        // затримка для отримання клієнтом даних
        delay(1);
        // закрити з'єднання
        client.stop();

        digitalWrite(rele, pinState);
```

```
    startGet = false;
    getData = "";
  }
```

Після завантаження програми у пристрій введіть в адресному рядку браузера адресу Arduino. Тепер з сайту віддалено можна керувати цифровими сигналами, тобто подавати HIGH або LOW на пін реле. Приклад WEB-інтерфейсу показано на рисунку 3.31.

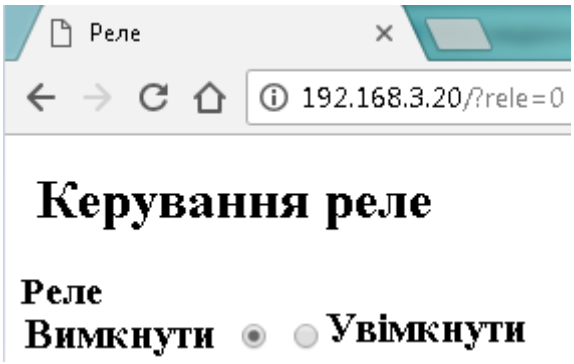


Рисунок 3.31 – Приклад WEB-інтерфейсу для керування реле

3.5.3 Тест для самоперевірки

1. Протокол безперервної передачі відповідних значень на сигнальному проводі називається...
   1) бінарним;
   2) двійковим;
   3) дестиїчним;
   4) унарним.
2. Функція localIP() дозволяє дізнатися…
   1) поточну MAC-адресу, присвоєну шілду;
   2) поточну IP-адресу, присвоєну шілду;
   3) кількість байтів даних з буфера прийому;
   4) кількість байтів даних, надісланих віддаленим сервером.
3. Функція EthernetClient available() дозволяє дізнатися …
   1) підключення створеного клієнта до IP-адреси і порту;
   2) поточну IP-адресу, присвоєну шілду;
   3) кількість байтів даних з буфера прийому;
   4) наявність запиту на підключення до створеного серверу.
4. Connect()– це функція…
   1) підключення клієнта до IP-адреси і порту;
   2) читає наступний байт даних з буфера прийому;

3) повертає кількість байтів даних з буфера прийому;
4) підключення запиту на підключення до створеного серверу.

5. Функція unsigned int available()…
   1) підключення клієнта до IP-адреси і порту;
   2) підключення запиту на підключення до створеного серверу;
   3) повертає поточну IP-адресу, присвоєну шілду;
   4) повертає кількість байтів даних з буфера прийому.
6. Функція int read()…
   1) повертає кількість байтів даних з буфера прийому;
   2) підключення клієнта до IP-адреси і порту;
   3) підключення запиту на підключення до створеного серверу;
   4) читає наступний байт даних з буфера прийому.

Відповіді на тест

1	2	3	4	5	6
1)	2)	4)	1)	4)	4)

3.6 Відправка даних за допомогою ThingTweet

3.6.1 Реєстрація в Twitter

Для подальших дій необхідний акаунт у Twitter [34]. Якщо у вас його немає, то зареєструйтеся в цій соціальній мережі:

– відкрийте сторінку http://twitter.com і знайдіть на ній поля реєстрації або перейдіть безпосередньо на сторінку реєстрації https://twitter.com/signup;

– введіть свої ім'я та прізвище, номер телефону і пароль;

– натисніть Зареєструватися;

– вам надійде SMS-повідомлення з кодом, щоб перевірити ваш номер телефону. Введіть цей код підтвердження у відповідне поле. Детальніше про прив'язку телефонного номера до облікового запису можна прочитати тут;

– після натискання кнопки Реєстрація можна обрати ім'я користувача (ім'я користувача – це унікальний ідентифікатор у Twitter). Введіть власне ім'я користувача або оберіть одне із запропонованих. Вас попередять, якщо обране ім'я користувача зайнято;

– уважно перевірте введені ім'я та прізвище, номер телефону, пароль та ім'я користувача;

– натисніть Створити обліковий запис. Вам може бути запропоновано ввести текст з картинки, щоб підтвердити, що ви є людиною.

3.6.2 Застосунок ThingTweet сервісу ThingSpeak

Застосунок ThingTweet сервісу ThingSpeak дозволяє пристроям надіслати повідомлення у профіль Twitter, використовуючи API ThingSpeak [34]. При цьому сервіс ThingSpeak діє для Twitter як проксі-сервер – щоб пристрої відправляли повідомлення в Twitter без необхідності реалізовувати відкриту аутентифікацію (OAuth).

Авторизуйтеся в сервісі ThingSpeak та оберіть пункт меню Apps – відкриється вікно вибору застосунків (рис. 3.32).

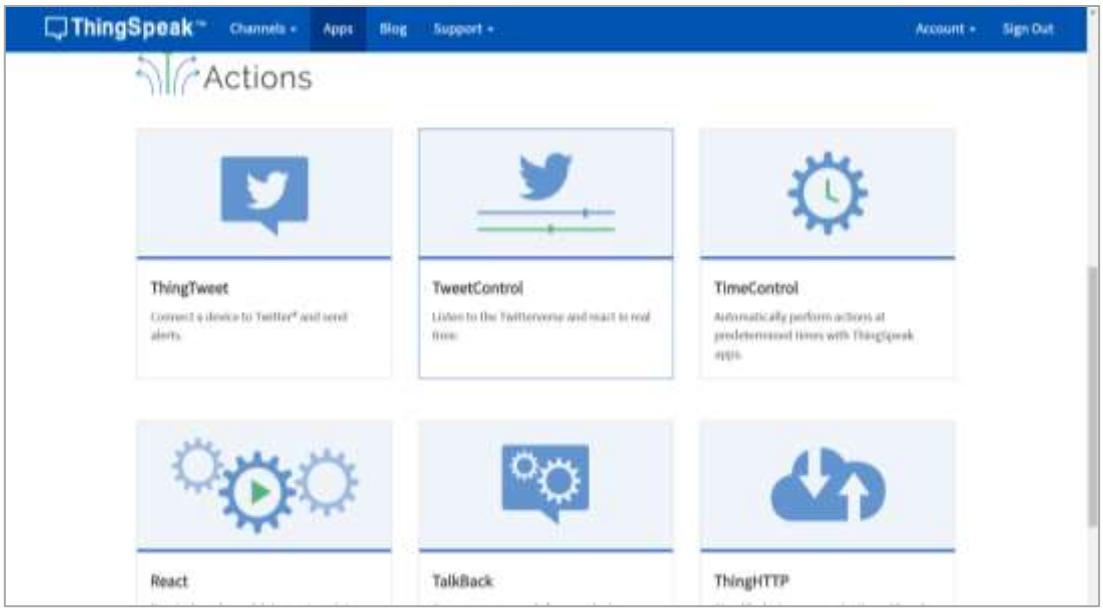


Рисунок 3.32 – Вікно вибору застосунку в сервісі ThingSpeak

Оберіть програму ThingTweet і ознайомтеся з інформацією сторінки з посиланням переходу у ваш акаунт Twitter (рис. 3.33).

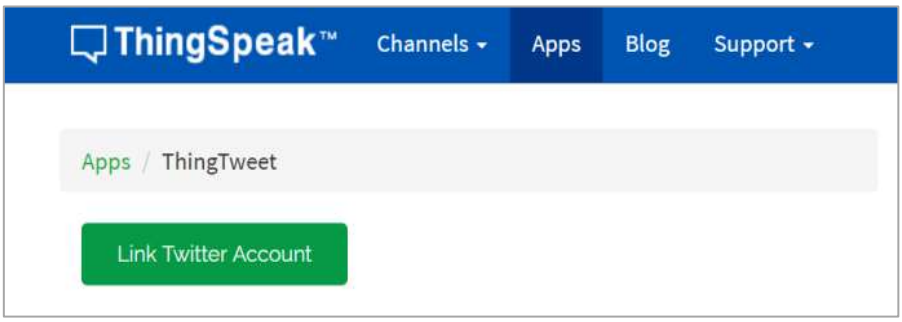


Рисунок 3.33 – Вікно переходу в акаунт Twitter

У наступному вікні (вже на сторінці Twitter) необхідно дозволити програмі ThingTweet доступ до даних вашого облікового запису Twitter, натиснувши на кнопку Авторизувати (рис. 3.34).

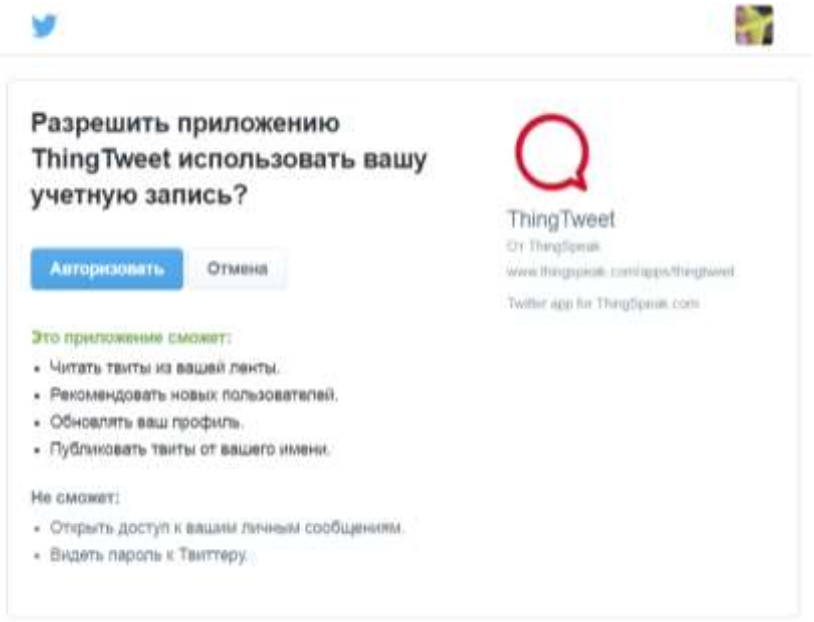


Рисунок 3.34 – Надання застосунком ThingTweet доступу до акаунту Twitter

Після того, як ви надасте застосунку ThingTweet доступ до даних вашого облікового запису в Twitter, Twitter перенаправить вас назад у ThingSpeak, де застосунок ThingTweet згенерує ключ API (рис. 3.35).

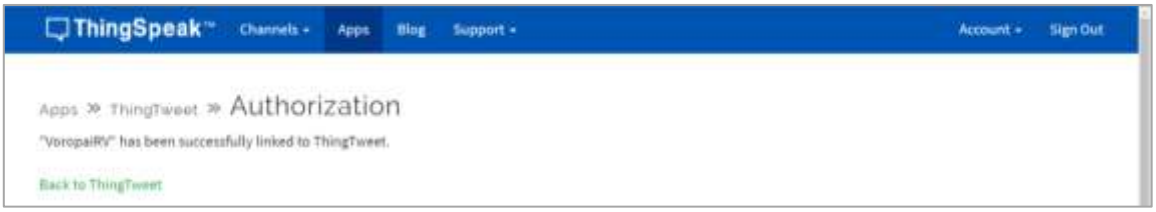


Рисунок 3.35 – Доступ з застосунком до облікового запису Twitter надано

Тепер, коли ви станете надсилати HTTP POST з ключем ThingTweet API (API Key), ваше повідомлення буде передано в Twitter.

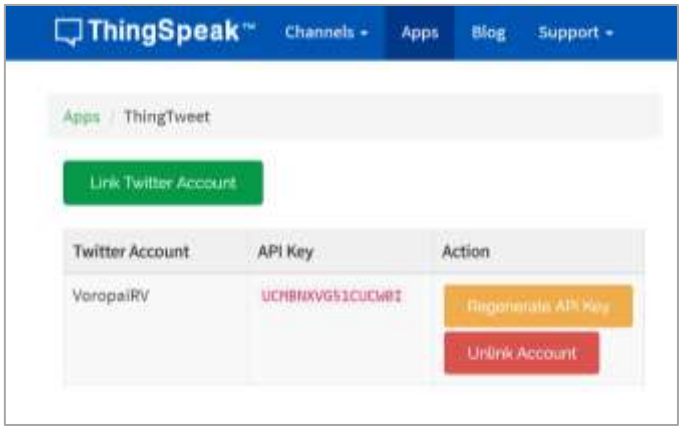


Рисунок 3.36 – Генерація API-ключів для перенаправлення даних у Twitter

3.6.3 Відправка даних у Twitter з Arduino

Для прикладу, необхідно організувати відправку з періодичністю в 10 хвилин даних з застосунком ThingSpeak, який перенаправить їх у Twitter. Для цього один раз на 10 хвилин будемо відправляти з плати Arduino, яка поєднана з Ethernet Shield, в сервіс ThingSpeak запит HTTP POST такого змісту:

```
POST /apps/thingtweet/1/statuses/update HTTP/1.1
Host: api.thingspeak.com
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length:
<length>api_key=<thingtweet_api_key>&status==<message>
```

де:

– <length> — кількість символів у повідомленні;

– <thingtweet_api_key> — отриманий API-ключ JUPP0TQ752M2AOEP;

– <message> — повідомлення.

Інформацію про сигнал передаватимемо серверу кожні 10 хвилин. Якщо немає спрацьовувань, інформація не передається.

Схема макета наведена на рисунку 3.37.

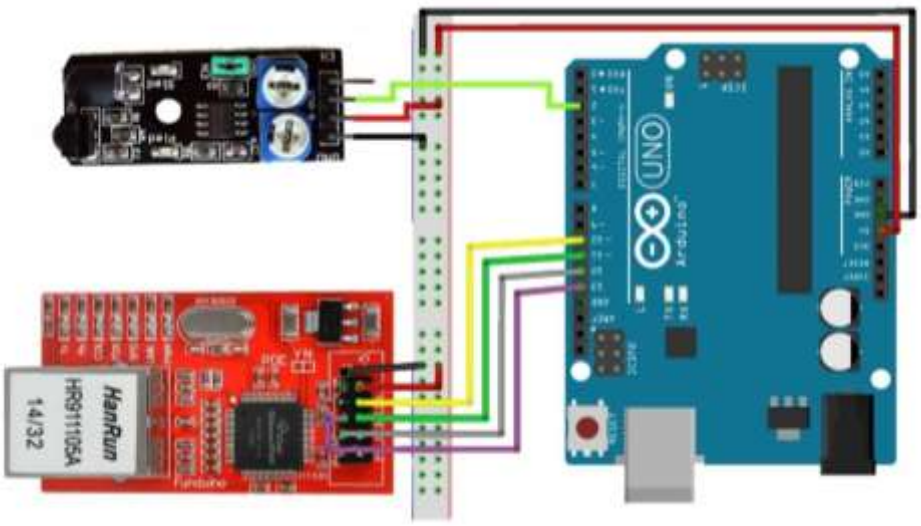


Рисунок 3.37 – Схема макета для відправлення повідомлень

Вихідний код програми для Arduino:

```
#include<SPI.h>
#include <Ethernet.h>
  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
  byteip[] = { 192, 168, 0, 110 };
constintIRpin = 2; // пін для підключення сенсора
```

```
int valuel; // для зберігання значення
    unsigned long timevisitors = 0;
    int signal=0;
    unsigned long time10minute=500000;
// ThingSpeak Settings
char thingSpeakAddress[] = «api.thingspeak.com»;
String thingtweetAPIKey = «UCMBNXVG51CUCW0I»;
// службові змінні
long lastConnectionTime =0;
boolean lastConnected = false;
int failedCounter =0;
// ініціалізація Arduino Ethernet Client
EthernetClient client;
void setup()
{Serial.begin(9600); // запуск послідовного порту
startEthernet();
  delay(1000);}
void loop()
{  valuel=digitalRead(IRpin);
  if(valuel)
    {
      timevisitors=millis();
      while(digitalRead(IRpin)) {};
  if(millis()-timevisitors>300)
    {
      Serial.println(«passage!!!»);
      signal=signal+1;
      Serial.print(«signal obstacle avoidance sensor=«);
      Serial.println(signal) ;
      }
    }
      delay(200);
  if (millis () -time10minute>600000) {
      String  message=«last  10.  minute  «+String(signal)+»  signal
obstacle avoidance sensor»;
      updateTwitterStatus(message);
      time10minute=millis();
    }
// друк відповіді від сервера в монітор
  while (client.available())
    {
      char c = client.read();
      Serial.print(c);
```

```
      }
  // відключення від ThingSpeak
     if (!client.connected() && lastConnected)
      {
        Serial.println(«...disconnected»);
        Serial.println();
        client.stop();
      }
  // рестарт мережевого з'єднання
    if (failedCounter> 3 ) {startEthernet();}
  lastConnected = client.connected();
  }
  // усереднення декількох значень для згладжування
  void updateTwitterStatus(String tsData)
  {
    if (client.connect(thingSpeakAddress, 80))
    {
      // Create HTTP POST Data
      tsData = «api_key=«+thingtweetAPIKey+»&status=«+tsData;
      client.print(«POST /apps/thingtweet/1/statuses/update HTTP/1.1\n»);
      client.print(«Host: api.thingspeak.com\n»);
      client.print(«Connection: close\n»);
      client.print(«Content-Type: application/x-www-form-urlencoded\n»);
      client.print(«Content-Length: «);
      client.print(tsData.length());
      client.print(«\n\n»);
      client.print(tsData);
      lastConnectionTime = millis();
      if (client.connected())
      {
        Serial.println(«Connecting to ThingSpeak...»);
        Serial.println();
        failedCounter = 0;
      }
      else
      {
        failedCounter++;
        Serial.println(«Connection to ThingSpeak failed
(«+String(failedCounter, DEC)+»)»);
        Serial.println();
      }
    }
    else
```

```
    {
      failedCounter++;
      Serial.println(«Connection to ThingSpeak Failed
(«+String(failedCounter, DEC)+»)»);
      Serial.println();
      lastConnectionTime = millis();
    }
  }
  void startEthernet()
  {
    client.stop();
    Serial.println(«Connecting Arduino to network...»);
    Serial.println();
    delay(1000);
    // Connect to network amd obtain an IP address using DHCP
    if (Ethernet.begin(mac) == 0)
    {
      Serial.println(«DHCP Failed, reset Arduino to try again»);
      Serial.println();
    } else
    {
      Serial.println(«Arduino connected to network using DHCP»);
  Serial.println();
    }
  delay(1000);}
```

Завантажте скетч у плату, підключіть послідовний порт, дочекайтеся з'єднання з сервером. Вплиньте на датчик, щоб він спрацював. У порт виведеться рядок спрацьовування і номер спрацьовування. Кожні 10 хвилин плата передаватиме значення на сервер, їх можна буде побачити в акаунті Twitter.

Результат роботи програми показано на рисунку 3.38.

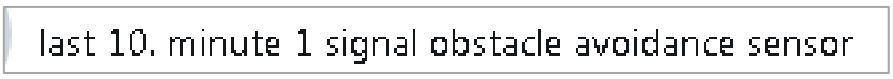


Рисунок 3.38 – Результат відправки повідомлення

3.6.4 Тест для самоперевірки

1. Команда <length> у відправці запиту позначає...
   1) передачу тексту;
   2) текст повідомлення;

3) довжину API-ключа;
4) кількість символів у повідомленні.

2. Команда <thingtweet_api_key> у відправці запиту позначає...
   1) передачу тексту;
   2) текст повідомлення;
   3) довжину API-ключа;
   4) кількість символів у повідомленні.
3. Команда <message> у відправці запиту позначає...
   1) довжину API-ключа;
   2) кількість символів у повідомленні;
   3) передачу тексту;
   4) текст повідомлення.
4. Команда failedCounter++ позначає...
   1) нарощування змінної на 1;
   2) зменшення змінної failedCounter на 1;
   3) обнулення змінної failedCounter;
   4) ініціалізацію змінної failedCounter.
5. SCK підключається для...
   1) передачі даних від веденого пристрою до провідного;
   2) передачі даних від провідного пристрою до веденого;
   3) активізації Майстром того чи іншого периферійного пристрою;
   4) синхронізації процесу передачі даних.
6. MISO підключається для...
   1) синхронізації процесу передачі даних;
   2) передачі даних від провідного пристрою до веденого;
   3) передачі даних від веденого пристрою до провідного;
   4) активізації Майстром того чи іншого периферійного пристрою.

Відповіді на тест

1	2	3	4	5	6
4)	3)	4)	3)	4)	3)

3.7 Система контролю доступу з використанням технології RFID

3.7.1 Принцип дії радіочастотної ідентифікації RFID

Типова система RFID (Radio Frequency IDentification) [35] показана на рисунку 3.39.

Вона складається з:

– радіочастотної мітки або транспондера (англ. – Tag, Transponder);

– зчитувача інформації (Reader);

– пристрою для обробки інформації – комп’ютера.

Рисунок 3.39 – Типова система RFID

Мітка і зчитувач пов'язуються між собою радіочастотним каналом (рисунок 3.40):

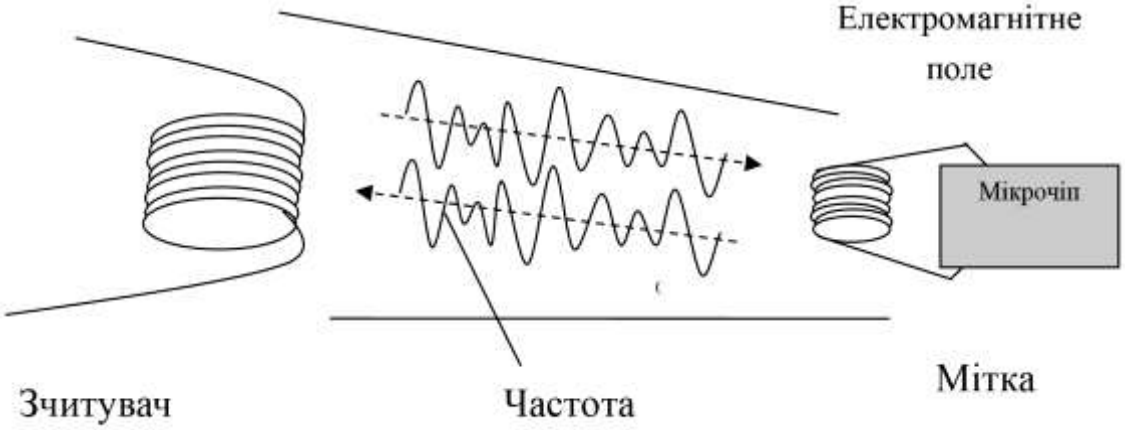


Рисунок 3.40 – Зв’язок між міткою та зчитувачем

Зчитувач містить передавач і антену, за допомогою яких випромінюється електромагнітне поле певної частоти. Радіочастотні мітки, що потрапили в зону дії поля, що зчитує, «відповідають» власним сигналом, який містить корисну інформацію (наприклад, код товару) на тій самій або іншій частоті. Сигнал вловлюється антеною зчитувача, корисна інформація розшифровується й передається в комп'ютер для обробки.

3.7.2 Активні й пасивні мітки

Радіочастотна мітка зазвичай містить у собі приймач, передавач, антену й блок пам'яті для зберігання інформації [35]. Приймач, передавач і пам'ять

конструктивно виконуються у вигляді окремої мікросхеми (чіпа), тому зовні здається, що радіочастотна мітка складається лише з двох частин: багатовиткової антени й чіпа. Іноді до складу конструкції мітки включається джерело живлення (наприклад, літієва батарейка).

Мітки з джерелами живлення називаються активними (Active). Дальність зчитування активних міток не залежить від енергії зчитувача.

Пасивні мітки (Passive) не мають власного джерела живлення, а необхідну для роботи енергію одержують від електромагнітного сигналу, що надходить від зчитувача. Дальність читання пасивних міток залежить від енергії зчитувача.

Перевагою активних міток порівняно з пасивними є значно більша (не менше, ніж у 2–3 рази) дальність зчитування інформації й висока припустима швидкість руху активної мітки відносно зчитувача.

Перевагою пасивних міток є практично необмежений термін їх служби (не вимагають заміни батарейок). Недолік пасивних міток у необхідності використання потужних пристроїв зчитування інформації, що мають відповідні джерела живлення.

3.7.3 Способи запису інформації на мітки

Інформацію в пристрій пам'яті радіочастотної мітки можна занести різними способами. Спосіб запису інформації залежить від конструктивних особливостей мітки. Залежно від цього розрізняють такі типи міток:

Read Only-мітки, які працюють тільки на зчитування інформації. Необхідні для зберігання дані заносяться на згадку мітки виготовником і не змінюються в процесі експлуатації.

WORM-мітки ('Write Once Read Many") для однократного запису й багаторазового зчитування інформації. Вони надходять від виготовника без будь-яких даних користувача в пристрої пам'яті. Необхідна інформація записується самим користувачем, але тільки один раз. У разі необхідності змінити дані потрібна нова мітка.

R/W-мітки ('Read/Write") багаторазового запису й багаторазового зчитування інформації.

3.7.4 Діапазони частот

Частоти електромагнітного випромінювання зчитувача й зворотного сигналу, переданого міткою, значно впливають на характеристики роботи радіочастотної системи в цілому. Як правило, чим вище діапазон робочих

частот системи RFID, тим більша дальність, на яких зчитується інформація з радіочастотних міток (табл. 3.1).

Таблиця 3.1 – Діапазон робочих частот системи RFID

Діапазон частот	Характеристики системи	Приклади застосування
Низькі 100 – 500 кГц	Мала дальність зчитування, низька вартість міток.	Контроль доступу. Ідентифікація тварин. Системи інвентаризації.
Проміжні 10 – 15 МГц	Середня дальність зчитування.	Контроль доступу. Смарт-карти.
Високі 850 – 950 МГц 2,4–5,0 ГГц	Більша дальність і швидкість зчитування, необхідне точне націлювання зчитувача, висока вартість міток.	Спостереження за перевезенням вантажів залізницею. Системи одержання плати за користування дорогою з водіїв автомобілів.

Низькочастотні мітки мають вбудовані антени у вигляді багатоконтурних (кілька сотень) обмоток. Високочастотні мітки мають одноконтурні обмотки (диполь–антена).

Найменші розміри й вартість мають пасивні мітки класу Read Only (тільки читання) і малої дальності (відстань до зчитувача не більше 2-х метрів).

3.7.5 Класифікація радіочастотних систем

У класифікаційній схемі, що наводиться нижче (рис. 3.41), зверніть увагу на англомовну термінологію, яка використовується серед виробників і постачальників засобів RFID [36].

3.7.6 Переваги радіочастотних міток

Універсальною технологією в області автоматичної ідентифікації є штрихове кодування і найчастіше використовуються символіки EAN/UCC. Радіочастотна ідентифікація порівняно зі штриховим кодуванням має такі переваги:

– дані ідентифікаційної мітки можуть доповнюватися;

– на мітку можна записати набагато більше даних;

– дані на мітку заносяться значно швидше;

– дані на мітці можуть засекречуватися;

– радіочастотні мітки довговічніші;

– розташування мітки не має особливого значення для зчитувача;

– мітка краще захищена від впливу навколишнього середовища.

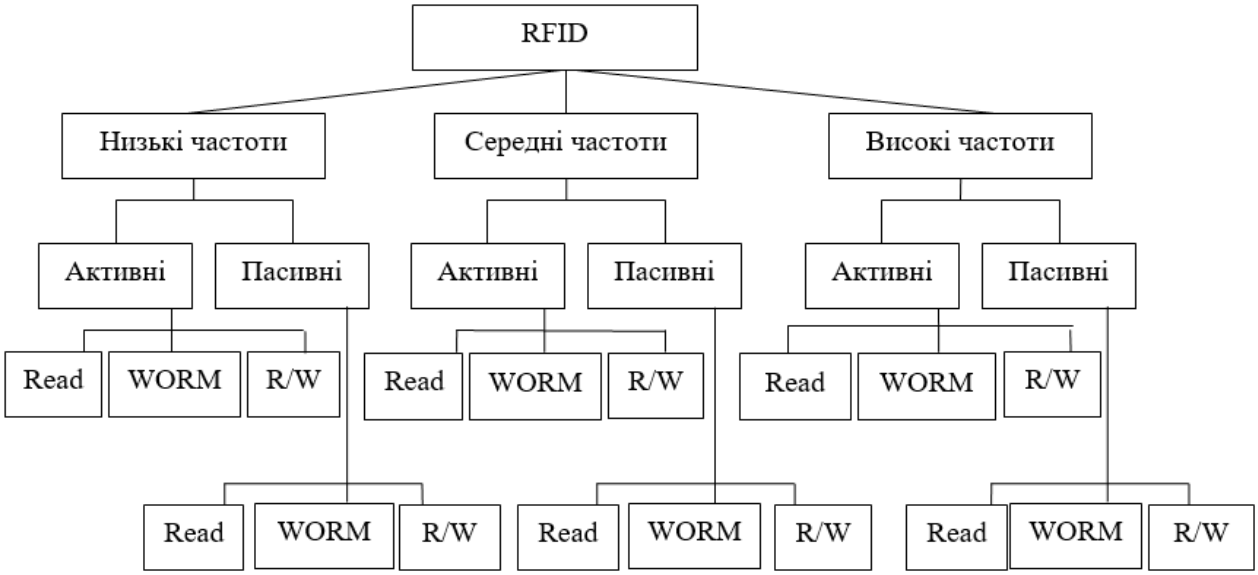


Рисунок 3.41 – Класифікація радіочастотних систем

Дані ідентифікаційної мітки можуть доповнюватися

У той час, як дані штрихового коду записуються тільки один раз (під час друкування), інформація, збережена радіочастотною міткою, може змінюватися, доповнюватися або навіть замінюватися на іншу за наявності відповідних умов. Це положення стосується тільки міток 'Read/Write" багаторазового запису й зчитування інформації.

На мітку можна записати набагато більше даних

Нещодавно розроблені двовимірні й матричні штрихові коди, здатні зберігати великий обсяг даних, однак їх практичне використання стримується необхідністю використання специфічних принтерів і пристроїв зчитування (сканерів). Звичайні штрихові коди можуть містити інформацію не більше 50 байт (знаків), причому для відтворення такого символу знадобиться площа розміром зі стандартний аркуш формату А4.

У свою чергу радіочастотна мітка може легко помістити 1000 байт на мікросхемі площею в 1 квадратний сантиметр. Не виникає серйозної технічної проблеми й у розміщенні інформації обсягом 10 000 байт [36].

Дані на мітку заносяться значно швидше

Для одержання штрихового коду зазвичай необхідно надрукувати його символ або безпосередньо на матеріалі пакування, або на паперовій етикетці. І печатка, і наклеювання липкої етикетки є або ручними, або механізованими

операціями. Радіочастотні мітки можна імплантувати в основу пакування на весь термін їх експлуатації. Самі дані про зміст пакування записуються винятково безконтактним способом за час, що не перевищує однієї секунди.

Дані на мітці можуть бути засекречені

Як і будь-який цифровий пристрій, радіочастотна мітка має можливості, що дозволяють закрити паролем операції запису й зчитування даних. Крім того, інформацію можна зашифрувати. В одній і тій самій мітці можна водночас зберігати закриті й відкриті дані. Це робить радіочастотну мітку ідеальним засобом, що захищає товари й матеріальні цінності від підробок і крадіжок.

Радіочастотні мітки довговічніші

У тих сферах застосування, де один і той самий маркірований об'єкт може використовуватися незліченну кількість разів (наприклад, у процесі ідентифікації пакунків або поверненої тари), радіочастотна мітка є ідеальним засобом ідентифікації, тому що може використовуватися 1 000 000 разів.

Розташування мітки не має особливого значення для зчитувача

З метою забезпечення автоматичного зчитування штрихового коду комітетами зі стандартів (у тому числі EAN International) розроблені правила розміщення символів штрихового коду на товарному й транспортному пакуванні. Для радіочастотних міток ці вимоги несуттєві. Єдине, що необхідно для зчитування інформації з радіочастотної мітки, – це її знаходження в зоні дії сканера RFID.

Мітка краще захищена від впливу навколишнього середовища

Радіочастотні мітки не потрібно розміщувати на зовнішній стороні пакування (об'єкта). Тому вони виявляються краще захищеними в умовах зберігання, обробки й транспортування логістичних одиниць. На відміну від штрихового коду, на них не впливають пил і бруд.

3.7.7 Недоліки радіочастотних міток

Радіочастотним міткам властиві й деякі недоліки:

– відносно висока вартість;

– неможливість розміщення під металевими й електропровідними поверхнями;

– взаємні колізії;

– чутливість до перешкод у вигляді електромагнітних полів;

– вплив на здоров'я людини.

Відносно висока вартість

Вартість пасивної радіочастотної мітки, що працює на середніх частотах 13,56 Мгц, становить (в ум. од.) :

– 30 центів у разі придбання 1 шт;

– 10 центів у разі придбання 100 шт;

– 2 центи у разі придбання понад 100 000 000 шт.

Отже, вартість радіочастотних міток значно перевищує вартість етикеток зі штриховим кодом на пакуванні товарів. Зображення символу штрихового коду EAN-13, що входить до загального оформлення пакування, практично нічого не коштує, у випадку використання етикетки, яка сама клеїться, її ціна лише 0,02 долара. Тому в наш час використання радіочастотних міток розміщення коду EAN-13 економічно не виправдано.

Разом з тим використання радіочастотних міток доцільно для захисту дорогих товарів від крадіжок або для забезпечення збереження виробів, переданих на гарантійне обслуговування. У сфері логістики й транспортування вантажів вартість радіочастотної мітки може виявитися незначною порівняно з вартістю вмісту контейнера. Тому великі супермаркети можуть почати використання RFID з застосування радіочастотних міток на пакувальних ящиках, пакунках і контейнерах.

Неможливість розміщення під металевими й електропровідними поверхнями

Радіочастотні мітки піддаються впливу металу (електромагнітне поле екранується струмопровідними поверхнями). Тому перед їх використанням у пакуваннях певного виду (наприклад, металевих контейнерах) його варто модернізувати. Це стосується й деяких типів пакування рідких харчових продуктів, загорнутих фольгою (зміст – тонкий лист металу). Відомі випадки маркування мітками RFID пакувань із взуттям. При цьому в умовах вологості шкіра взуття ставала електропровідною й робота системи RFID погіршувалася.

Взаємні колізії

У багатьох випадках у поле дії зчитувача може водночас потрапити кілька радіочастотних міток. Це можна зробити навмисно, наприклад, у магазині під час проходу через пункт контролю. Добре контрольне устаткування має не тільки виявляти радіочастотні мітки, але й чітко ідентифікувати кількість однотипних міток, щоб, заплативши тільки за один виріб, було неможливо водночас винести інші того самого виду. Така технологія існує. Звичайно, складно ідентифікувати й підрахувати

кількість міток кожного типу, які водночас потрапили в поле дії зчитувача, не пропустивши жодної з них. У зчитувачах, що мають і характеристики, реалізовано спеціальний алгоритм антиколізії. Хоча технології антиколізії успішно продемонстровані в лабораторних умовах, на практиці вони досить мало застосовуються у зв'язку з тим, що їх реалізація призводить до значного збільшення часу зчитування.

Проблема існує й вимагає свого вирішення особливо у сфері постачання. Найпростіше рішення полягає у використанні єдиної радіочастотної мітки на пакуванні кожного рівня. Наприклад, на транспортному пакуванні (контейнері) розміщується одна мітка, до пам’яті якої записуються дані про всі товари, розміщені в упаковці.

Системи радіочастотної ідентифікації можуть бути чутливі до перешкод у вигляді електромагнітних полів від підключених комп'ютерів (моніторів). Тому необхідно ретельно проаналізувати умови, в яких експлуатуватиметься система RFID.

3.7.8 Опис компонентів макета

Кожна RFID-система включає в себе *зчитувач/рідер* і *RFID-мітку* (рис. 3.42), в якій зберігаються дані [37]. *Мітки складаються з двох частин* – інтегральної схеми та антени. Інтегральна схема дозволяє зберігати й обробляти дані, антена – приймати і передавати інформацію.

Рисунок 3.42 – RFID-мітка

Всі RFID-системи можна розділити за довжиною дії:

– ближньої ідентифікації – відстань не більше 20 см;

– середньої ідентифікації – відстань від 20 см до 5 м;

– дальньої ідентифікації – максимум 300 м.

З точки зору частот можна виділити:

– системи, які працюють у низькочастотному діапазоні (125 кГц, 134 кГц);

– системи, які працюють у середньочастотному діапазоні (13,56 МГц);

– системи, які працюють у високочастотному діапазоні (800 МГц–2, 4 ГГц).

По суті RFID-мітки – це маркери з унікальним кодом і можливістю зберігати певну кількість інформації. Якщо такі маркери прикріпити до відомих об'єктів, то, знаючи їх ID, можна визначити, з яким об'єктом у даний момент ми працюємо (рис. 3.43).

Рисунок 3.43 – Комплект RFID

Даний вид маркерів починає працювати, коли його підносять до пристрою читання [37]. Оскільки маркер не містить елементів живлення, то він його бере від пристрою читання, випрямляючи наведений на внутрішню антену сигнал. Споживання маркера мале, і цієї потужності йому вистачить.

Модуль зчитування володіє такими характеристиками:

– заснований на мікросхемі MFRC522;

– напруга живлення: 3.3 V;

– споживаний струм: 13– 26 mA;

– робоча частота: 13.56 MHz;

– дальність зчитування: 0 ~ 60 мм;

– інтерфейс: SPI, максимальна швидкість передачі 10 Мбіт/с;

– розмір: 40 мм х 60 мм;

– читання і запис RFID-міток.

Підключення модуля до Arduino наведено на рисунку 3.44.

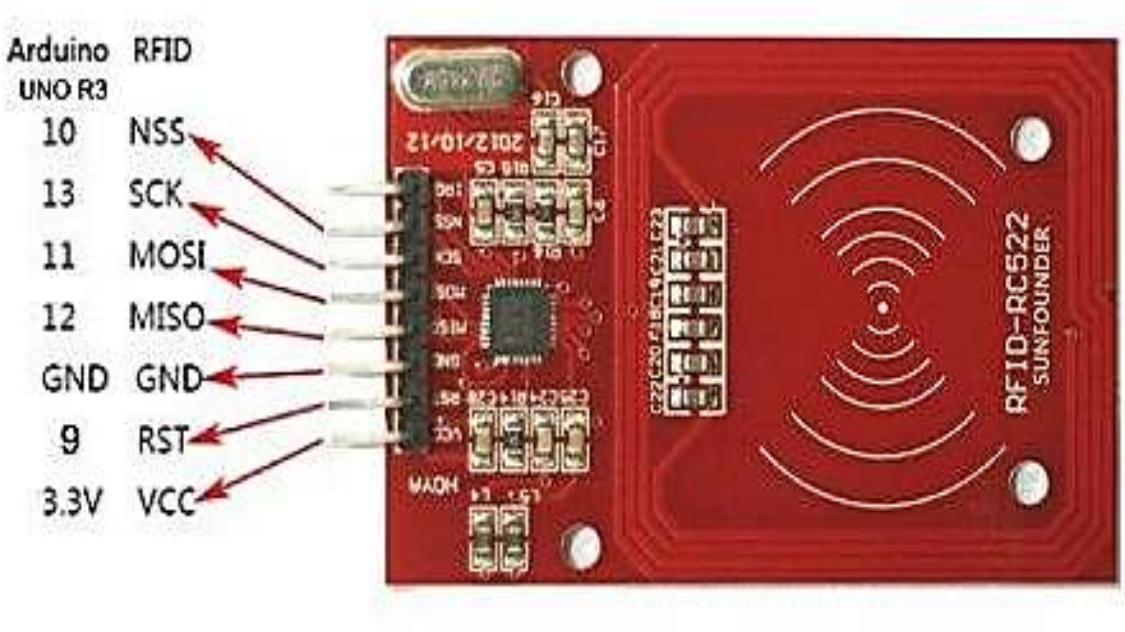


Рисунок 3.44 – Підключення модуля до Arduino

3.7.9 Робота з бібліотеками rfid.h і SPI.h

Для полегшення роботи з RFID-модулем існує бібліотека rfid.h. Бібліотека містить такі функції:

– RFID nameRfid(SS_PIN, RST_PIN) – створення об'єкта типу RFID;

– init() – ініціалізація RFID-модуля. Виконується функції setup();

– isCard() – повертає 1, якщо мітка була виявлена, і 0, якщо мітка була не виявлена;

– readCardSerial() – визначає серійний номер чіпа. Повертає 1, якщо номер був визначений, в іншому випадку повертає 0;

– halt() – обчислює контрольну суму, додає ознаку закінчення серійного номера. Не має значення, яке повертається.

Для зв'язку RFID-мітки використовується протокол SPI, також використовується бібліотека SPI.h і її функція SPI.begin() – ініціалізація SPI-інтерфейсу.

3.7.10 Практична реалізація системи контролю доступу

Напишемо програму, що використовує RFID-модуль для виявлення мітки та відправлення на COM-порт її коду. У випадку наступних виявлень даної мітки в порт виводиться точка.

Підключимо RFID-модуль до Arduino за такою схемою (рис. 3.45).

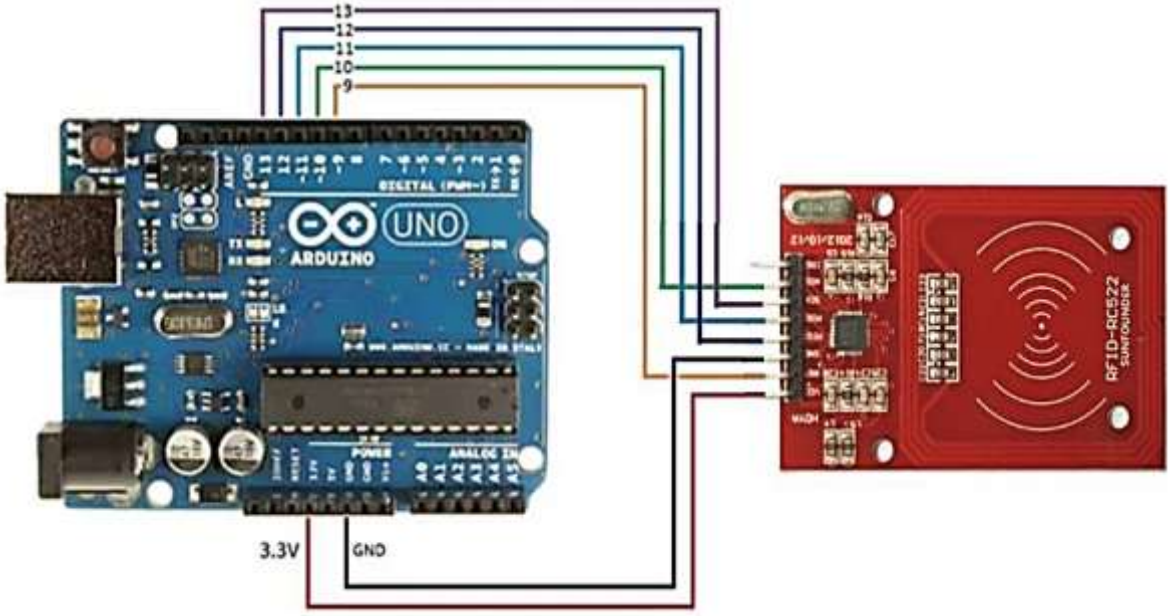


Рисунок 3.45 – Схема системи контрою доступу

Для роботи макета завантажимо в Arduino такий код:

```
#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

RFID rfid(SS_PIN, RST_PIN);

// Setup variables:
int serNum0;
int serNum1;
int serNum2;
int serNum3;
int serNum4;
void setup()
{
Serial.begin(9600);
SPI.begin();
rfid.init();
}
void loop()
{
if (rfid.isCard())
{
if (rfid.readCardSerial())
```

```
{
if (rfid.serNum[0] != serNum0
&& rfid.serNum[1] != serNum1
&& rfid.serNum[2] != serNum2
&& rfid.serNum[3] != serNum3
&& rfid.serNum[4] != serNum4
)
{
/* With a new cardnumber, show it. */
Serial.println(« «);
Serial.println(«Card found»);
serNum0 = rfid.serNum[0];
serNum1 = rfid.serNum[1];
serNum2 = rfid.serNum[2];
serNum3 = rfid.serNum[3];
serNum4 = rfid.serNum[4];
//Serial.println(« «);
Serial.println(«Cardnumber:»);
Serial.print(«Dec: «);
Serial.print(rfid.serNum[0], DEC);
Serial.print(«, «);
Serial.print(rfid.serNum[1], DEC);
Serial.print(«, «);
Serial.print(rfid.serNum[2], DEC);
Serial.print(«, «);
Serial.print(rfid.serNum[3], DEC);
Serial.print(«, «);
Serial.print(rfid.serNum[4], DEC);
Serial.println(« «);
Serial.print(«Hex: «);
Serial.print(rfid.serNum[0], HEX);
Serial.print(«, «);
Serial.print(rfid.serNum[1], HEX);
Serial.print(«, «);
Serial.print(rfid.serNum[2], HEX);
Serial.print(«, «);
Serial.print(rfid.serNum[3], HEX);
Serial.print(«, «);
Serial.print(rfid.serNum[4], HEX);
Serial.println(« «);
}
else
{
```

```
/* If we have the same ID, just write a dot. */
Serial.print(«.»);
}
}
}
rfid.halt();
}
```

3.7.11 Використання RFID для відкриття дверей у приміщення

Напишімо програму, яка у випадку виявлення мітки з певним кодом відкриває двері Розумного будинку і його шлагбаум. При цьому має видаватися звуковий сигнал.

Додамо до схеми, зібраної в минулому завданні, сервоприводи і бузер (рис. 3.46).

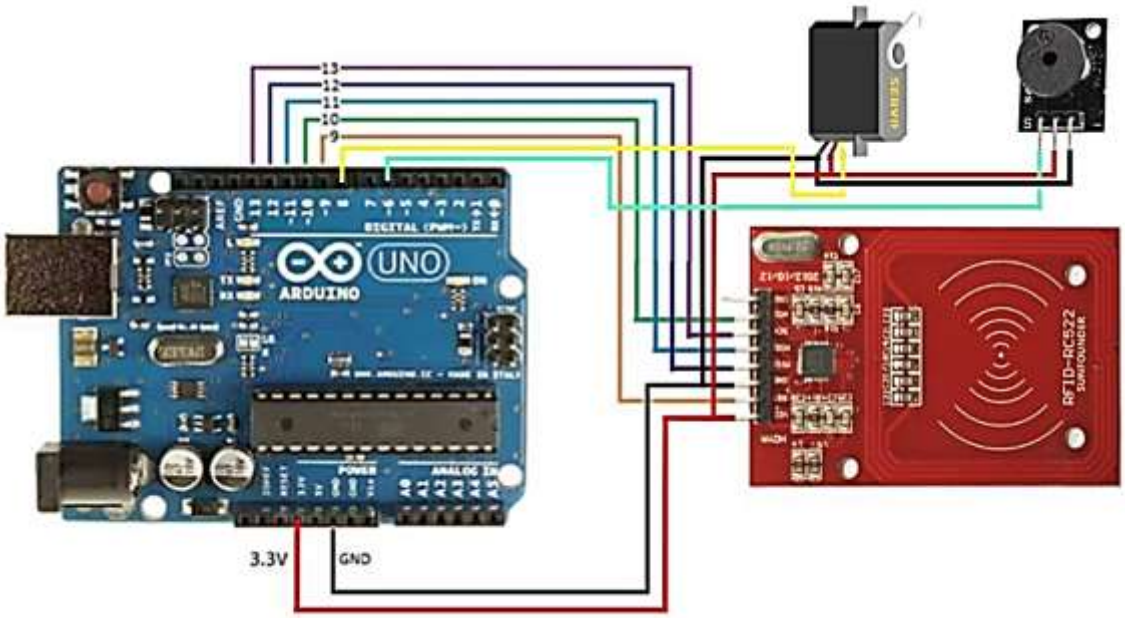


Рисунок 3.46 – Схема макета для управління електричним замком

Для роботи макета завантажимо в Arduino такий код:

```
#include<SPI.h>
#include <RFID.h>
#include <Servo.h>
#define SS_PIN 10
#define RST_PIN 9
#defineSOUND 6
#define locked 0 //положення сервоприводу «закрито»
#define unlocked 90 //положення сервоприводу «відчинено»
RFID rfid(SS_PIN, RST_PIN);
```

```
Servo lock;
unsigned char reading_card[5]; //for reading card
unsigned char master[5] = {102, 183, 130, 141, 222}; // allowed card
unsigned char i;
void setup()
{
Serial.begin(9600);
SPI.begin();
rfid.init();
pinMode(SOUND, OUTPUT);
lock.attach(8);
}
void loop()
{
if (rfid.isCard())
{
if (rfid.readCardSerial())
{ /* Reading card */
Serial.println(« «);
Serial.println(«Card found»);
Serial.println(«Cardnumber:»);
for (i = 0; i <5; i ++)
{
Serial.print(rfid.serNum[i]);
Serial.print(« «);
reading_card[i] = rfid.serNum[i];
}
Serial.println(); //verification
for (i = 0; i <5; i ++)
{
if (reading_card[i]!= master[i])
{
break;
}   }
if (i == 5)
{
allow();
}
else
{
denied();
}   }   }
rfid.halt();
```

```
}
void allow()
{
Serial.println(«Access accept!»);
lock.write(unlocked);
tone(SOUND, 1300, 200);
}
void denied()
{
Serial.println(«Access denied!»);
lock.write(locked);
tone(SOUND, 4300, 200);
}
```

Таким чином, ми розглянули будову RFID-модуля, ознайомилися з основними його характеристиками і схемою підключення. Розглянули бібліотеку rfid.h, яка використовується для полегшення роботи з RFID-модулем. Виконавши практичну частину, навчилися програмувати RFID-модуль у середовищі Arduino IDE для відкриття шлагбаума та дверей Розумного будинку.

3.7.12 Тест для самоперевірки

1. Кожна RFID-система включає в себе...
    1) інтегральну і антенну схему;
    2) зчитувач і мітку;
    3) пульт і мітку;
    4) пульт і зчитувач.
2. Інтегральна схема RFID-мітки дозволяє...
    1) зберігати і приймати дані;
    2) зберігати й обробляти дані;
    3) перевіряти дані;
    4) приймати і передавати дані.
3. Антенна схема RFID-мітки дозволяє...
    1) приймати і передавати дані;
    2) перевіряти дані;
    3) зберігати й обробляти дані;
    4) зберігати і приймати дані.
4. Команда readCardSerial() виконує функцію...
    1) визначення серійного номера чіпа;
    2) ініціалізації SPI-інтерфейсу;

3) додавання ознаки закінчення серійного номера;
4) ініціалізації RFID-модуля.

5. Команда SPI.begin() виконує функцію...
   1) визначення серійного номера чіпа;
   2) виявлення мітки;
   3) додавання ознаки закінчення серійного номера;
   4) ініціалізації SPI-інтерфейсу.
6. Команда isCard() виконує функцію...
   1) визначення серійного номера чіпа;
   2) виявлення мітки;
   3) ініціалізації SPI-інтерфейсу;
   4) додавання ознаки закінчення серійного номера.

Відповіді на тест

1	2	3	4	5	6
2)	2)	1)	1)	4)	2)

3.8 Організація роботи Arduino з шиною I2C

3.8.1 Відомості про шину I2C

Інтерфейс I2C був спочатку розроблений фірмою Philips наприкінці 1970-х років спеціально для того, щоб забезпечувати спосіб підключення периферійних пристроїв до мікропроцесорів, який не вимагав би використання традиційних шин адреси, даних і керування, а крім того, дозволяв би декільком мікропроцесорам працювати з одними й тими самими периферійними пристроями (multimastering) [38]. Philips запатентував назву інтерфейсу і до 2007 у різних виробників цей мікроконтролер мав свою назву. Наприклад, у мікроконтролерах ATmega, на яких будується Arduino, цей інтерфейс зветься 2-wire Serial Interface (двопроводовий послідовний інтерфейс).

Дані передаються двома проводами – провід даних і провід тактів. Є провідний (master) і ведений (slave), такти генерує master, ведений лише підтакує під час прийому байта. Лише на одній двопроводовій шині може бути до 127 пристроїв.

Передача/Прийом сигналів здійснюється притисканням лінії в 0, в одиничку встановлюється сама за рахунок резисторів, які підтягують. Їх ставити завжди обов'язково. Резистори на 10к оптимальні. Чим більший резистор, тим довше лінія відновлюється в одиницю (йде перезарядка

паразитної ємності між проводами) і тим сильніше завалюються фронти імпульсів, а відтак швидкість передачі спадає. Саме тому у I2C швидкість передачі набагато нижча, ніж у SPI. Зазвичай I2C працює або на швидкості 10 кбіт/с – у повільному режимі, або на 100 кбіт/с у швидкому. Але в реальності можна плавно змінювати швидкість аж до нуля.

У жодному разі не можна перемикати вивід мікроконтролера в OUT і смикати провід на +5. Може статися коротке замикання, яке спалить або контролер, або який-небудь девайс на шині.

Інтерфейс I2C використовує лише дві лінії – вони називаються SCL (Serial Clock) і SDA (Serial Data). Перша призначена для передачі синхроімпульсів (вони формуються тим пристроєм, який зараз передає дані), а друга – для передачі самих даних і команд, які керують цим процесом. Обидві лінії мають відкритий колектор (як і в багатьох інших випадках, коли необхідно, щоб до однієї лінії могло підключатися кілька різних пристроїв), тому вимагають підключення резисторів, які «підтягують», опором 1–10 кОм (рис. 3.47).

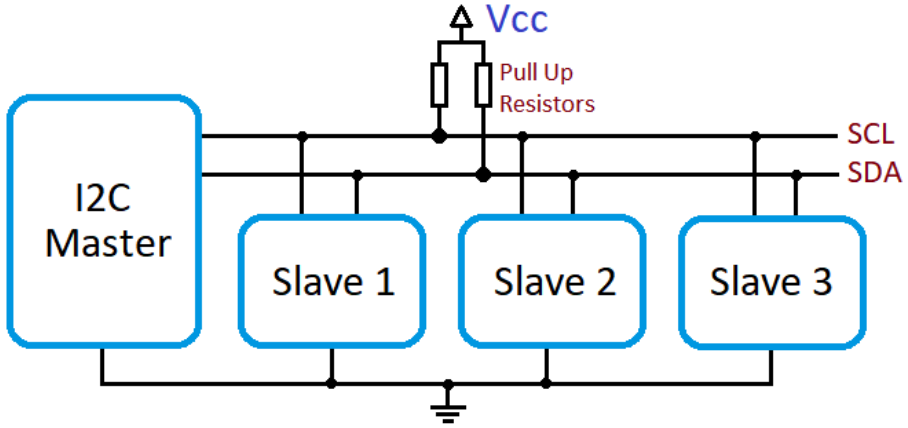


Рисунок 3.47 – Структурна схема пристрою керування з використанням I2C

Поки жоден пристрій не почав передачу даних, за допомогою підтягувальних резисторів на обох лініях шини I2C діє напруга високого рівня. Якщо будь-який пристрій збирається почати передачу даних, він спочатку перевіряє, чи вільна шина. Адже в кожен момент часу провідним на шині може бути тільки один пристрій. Напруга високого рівня на лінії SCL показує, що шина поки вільна.

Перед початком процесу передачі задавач (майстер) встановлює напругу низького рівня спочатку на лінії SDA, а потім на лінії SCL (рис. 3.48). У процесі передачі даних такий стан ліній неможливий, оскільки сигнал на лінії SDA не повинен змінюватися під час дії тактового імпульсу на лінії SCL.

У деяких випадках біт підтвердження передається високим рівнем сигналу, навіть якщо прийом пройшов успішно. Це показує, що обмін закінчений і передавач (зазвичай є або провідним пристроєм, або задавачем, який не повинен сам починати операцію обміну) може підготуватися до отримання наступного запиту. Цей режим використовується, коли мікроконтролер запрошує дані у будь-якого периферійного пристрою. В цьому випадку мікроконтролер є приймачем даних. Якщо замість біта підтвердження мікроконтролер виставить сигнал високого рівня, то ведений пристрій «зрозуміє», що наступну порцію даних пересилати не потрібно.

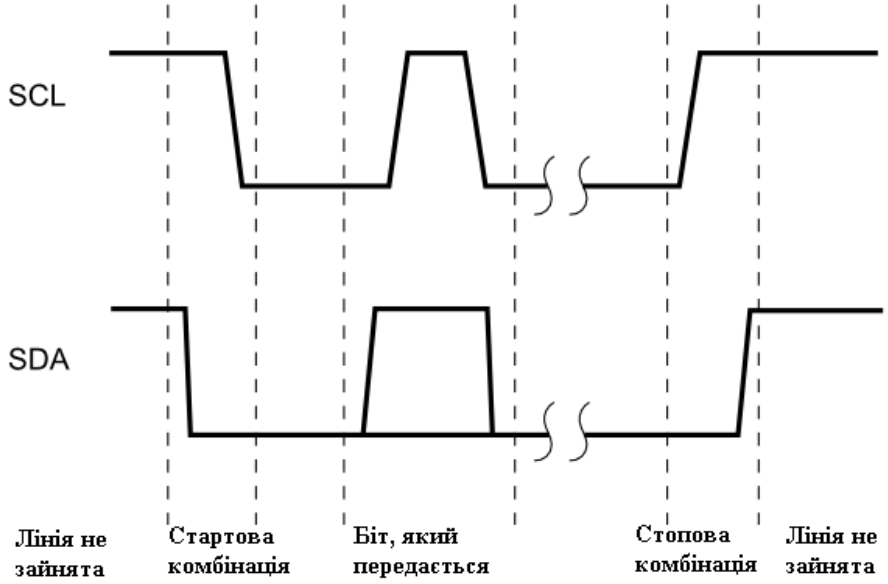


Рисунок 3.48 – Передача даних по інтерфейсу I2C

На рисунку 3.49 показаний формат команд, які використовуються для керування процесом передачі даних по інтерфейсу I2C.

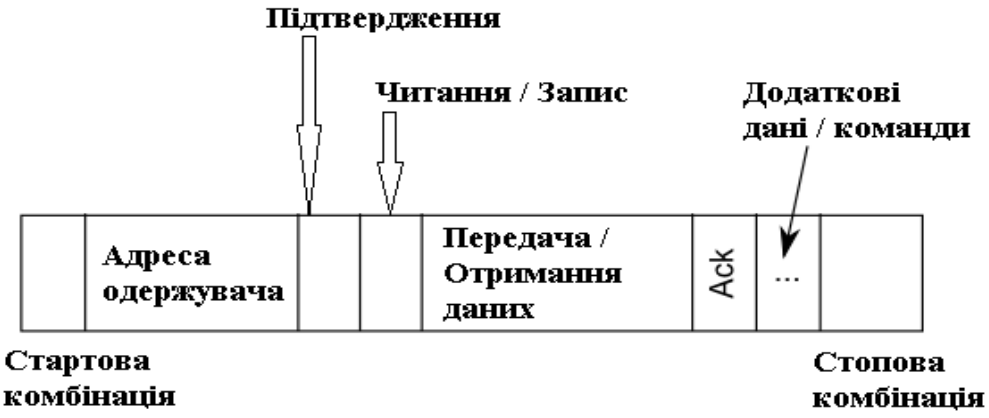


Рисунок 3.49 – Формат команд, які використовуються для керування процесом передачі даних по інтерфейсу I2C

Адреса одержувача задається сімома бітами. Старші чотири біти адреси визначають тип пристрою, а решту три молодших біти вказують, для якого саме пристрою (з восьми можливих) цього типу призначена інформація, яка надсилається.

У деяких випадках потрібно дещо ускладнювати протокол обміну. Наприклад, під час читання інформації з пам'яті EEPROM (або запису даних у пам'ять) задавач повинен спочатку встановити стартову послідовність, щоб переслати адресу потрібної комірки пам'яті, а потім знов виконати стартову послідовність, щоб тепер вже зчитати дані з пам'яті (або записати їх).

Для того, щоб провідними на шині могли бути різні пристрої, необхідний будь-який протокол дозволу колізій (конфліктів). Колізія виникає, коли два пристрої, водночас перевіривши стан шини і виявивши, що вона поки вільна, починають передачу даних.

Конфлікти вирішуються завдяки тому, що на лінії з відкритим колектором подача сигналу високого рівня реалізується, насправді, простим відключенням активного пристрою (згадайте про резистори, які «підтягують»). У цьому випадку перемагає завжди той пристрій, який виставив сигнал низького рівня. Тоді інший пристрій, «побачивши», що рівень напруги, який діє на лінії, не збігається з тим, який він намагається встановити, «розуміє», що на шині активний ще один задавач, і на час відключається, щоб дати йому можливість безперешкодно закінчити обмін інформацією.

Реалізація інтерфейсу I2C за допомогою мікроконтролерів вельми проста. Однак через програмну його реалізацію важко досягти високих швидкостей передачі. Навіть максимальна швидкість стандартного режиму (100 Кбіт/с) може виявитися недосяжною.

Програмна реалізація інтерфейсу I2C проте є найкращим рішенням, якщо крім мікроконтролера на шині не може бути інших задавачів. Адже в цьому випадку не потрібно синхронізувати його роботу з будь-якими швидкими пристроями, в яких використовується апаратна реалізація цього інтерфейсу.

3.8.2 Бібліотека Wire для Arduino

Ця бібліотека дозволяє спілкуватися з іншими пристроями за протоколом I2C / TWI. На більшості платформ Arduino, SDA (лінія даних) знаходиться на аналоговому порту 4, і SCL (лінія тактування) на аналоговому порту 5.

Існують 7- і 8- бітні версії I2C адресації. 7 біт ідентифікують пристрій, а 8-й біт визначає, чи будуть на пристрій відправлені дані, або зчитані з нього. Дана бібліотека використовує 7-бітну адресацію. Якщо вихідний код містить

8-бітну адресацію, то знадобиться відкинути молодший біт (зсунувши значення на 1 біт вправо), використовуючи адреси між 0 і 127.

Функції бібліотеки описані в таблиці 3.2.

Таблиця 3.2 – Функції бібліотеки Wire

Функція	Опис	Параметри
Wire.begin () Wire.begin (address)	Ініціалізує бібліотеку і підключається в шині I2C як master або slave	*address*: 7-біт адреса слейва (опціонально); якщо не визначено, то пристрій – майстер
Wire.requestFrom (address, quantity)	Робить запит на байти з іншого пристрою. Байти можуть бути отримані за допомогою функцій available() і receive()	*address*: 7-бітна адреса пристрою, у якого запитуємо дані, *quantity*: кількість запитуваних байт
Wire.beginTransmission (address)	Почати передачу на I2C слейв пристрій з встановленою адресою. Згодом поставити в чергу байти за допомогою функції send() і передати їх викликом endTransmission()	*address*: 7-бітна адреса пристрою для передачі
Wire.endTransmission ()	Завершить передачу слейву, яка була розпочата командою beginTransmission() і передає чергу байт, які встановлені функцією send()	
Wire.send (value) Wire.send (string) Wire.send (data, quantity)	Відправляє дані зі слейва у відповідь на запит майстра або створює чергу для передачі від майстра до слейва (між викликами beginTransmission() і endTransmission())	*value*: байт для передачі (byte) *string*: рядок для передачі (char *) *data*: масив даних для передачі (byte *) *quantity*: число байт даних для передачі (byte)
Wire.available ()	Повертає число байт, які доступні для отримання функції receive(). Вона повинна бути викликана майстром після виклику requestFrom () або слейвом всередині onReceive ()	Повертає число байт доступних для читання
byte Wire.receive ()	Отримує байти, які були передані від слейва до майстра після виклику requestFrom або передані від майстра до слейва	Повертає наступний переданий байт
Wire.onReceive (handler)	Реєструє функцію, яка повинна бути викликана коли слейв отримає дані від майстра	*handler*: функція, яка викликається після отримання даних. Вона повинна мати чисельний параметр int і нічого не повертати. Наприклад: void myHandler (int numBytes)
Wire.onRequest (handler)	Реєструє функцію, яка буде викликана, коли майстер запросить дані у слейва	*handler*: Функція, яка буде викликана, не повинна мати параметрів і не повинна нічого не повертати, тобто: void myHandler ()

3.8.3 Організація передачі повідомлень між платами Arduino

Для виконання даного прикладу необхідно підключити Arduino Uno і Arduino Nano, як показано на рисунку 3.50 [39]. Визначимо такий розподіл ролей: Arduino Uno – майстер, Arduino Nano – слейв.

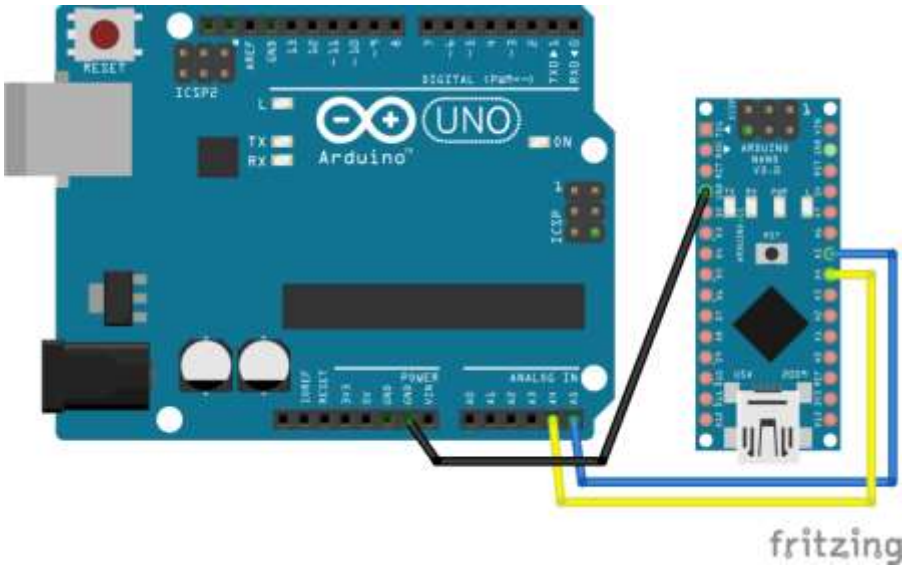


Рисунок 3.50 – Схема поєднання двох плат Arduino

Наступний крок – підключаємо Arduino Uno до комп'ютера. Виявляємо, на який послідовний порт вона визначилася. Зберігаємо скетч, назвавши його «master0».

Створюємо новий скетч та підключаємо Arduino Nano до комп'ютера. Виявляємо, на який послідовний порт вона визначилася, та зберігаємо скетч, назвавши його «slave0».

Код файлу «master0» містить такі інструкції:

```
#include <Wire.h>

void setup() {
 Serial.begin(9600);
 Wire.begin(); // підключення до шини I2C
}

void loop() {
 Wire.beginTransmission(8);  /* Почати передачу на I2C слейв з адресою 8  */
 Wire.write("Hello, slave!");
 Wire.endTransmission();    /* Завершити передачу */

 Wire.requestFrom(8, 14); /* Отримати відповідь від слейв з адресою 8, довжиною 14 символів (з урахуванням пробілів)*/
```

```
 while(Wire.available()){
    char c = Wire.read();
    Serial.print(c); // Вивести на монітор порту символи
 }
 Serial.println();

 delay(500); // Обов'язкова пауза між відправленням даних
}
```

Код файлу «slave0» містить такі інструкції:

```
#include <Wire.h>
int led = 8;
void setup() {
 Wire.begin(8);                /* Підключитися до I2c з адресою 8*/
 Wire.onReceive(receiveEvent);  /*  Функція  отримання  даних  від
майстра*/
 Serial.begin(9600);
 }

void loop() {
 delay(100);
}

void receiveEvent(int howMany) {
 while (0 <Wire.available()) {
    char c = Wire.read();      /* Отримання символів від майстра*/
    Serial.print(c);           /* Виведення отриманої інформації на
монітор порту*/
  }
 Serial.println();             /*Перехід на новий рядок*/
}

// Функція відповіді майстру
void requestEvent() {
 Wire.write("Hello, master!");  /*send string on request */
}
```

У налаштуваннях Arduino IDE обираємо послідовний порт Arduino Uno – відобразиться текст Hello, master!

У налаштуваннях іншого Arduino IDE обираємо послідовний порт Arduino Nano – відобразиться текст Hello, slave!

3.8.4 Керування пристроями, підключеними до однієї Arduino через іншу Arduino

Завдання №1

Засвітити світлодіод, підключений до 11 порту Arduino Nano, використовуючи Arduino Uno (використовуйте макетну плату).

Крок 1. Підключіть світлодіод на 11 порт Arduino Nano через резистор на 10 кОм.

Крок 2. Змініть код майстра:

```
#include <Wire.h>
void setup() {
 Serial.begin(9600);
 Wire.begin(); // підключення до шини I2C
}
void loop() {
 Wire.beginTransmission(8);
 Wire.write(1);
 Wire.endTransmission();
 delay(500); // Обов'язкова пауза між відправленням даних
}
```

Крок 3. Змініть код слейва:

```
#include <Wire.h>
int led = 8;
void setup() {
 Wire.begin(8);
 Wire.onReceive(receiveEvent);
 Serial.begin(9600);
 pinMode (led, OUTPUT);
}
void loop() {
 delay(100);
}

void receiveEvent(int howMany) {
 while (Wire.available()) {
     byte x = Wire.read();
     Serial.print (x);
     if(x == 1)
     digitalWrite(led, HIGH);
  }
```

```
  Serial.println();
}
```

Крок 4. Завантажте код.

В результаті виконання програми світлодіод має засвітитися.

Завдання №2

Вивести показники датчика температури і вологості, який підключений до майстра на екран, що підключений до слейва.

Крок 1. Підключіть датчик температури і вологості на 11-й порт Arduino Uno. Оскільки підключення екрану теж вимагає використання портів A5 і A5 (LCD працює на I2C), то використовуйте макетну плату для організації інтерфейсу I2C. Підключення екрану: SDA – A4; SCL – A5. Схема підключення двох Arduino та РК індикатора показана на рисунку 3.51.

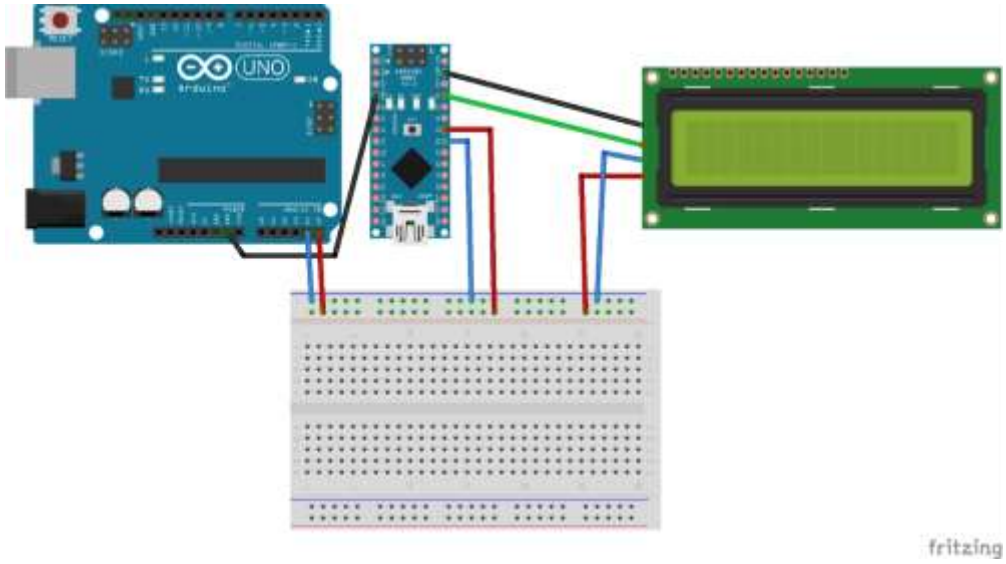


Рисунок 3.51 – Схема підключення двох Arduino та РК індикатора

Крок 2. LCD-дисплей теж працює на I2C-шині. Тому для передачі на нього даних необхідно дізнатися його адресу. Створіть новий скетч і запрограмуйте Arduino Nano:

```
#include <Wire.h>
oid setup() {
  Serial.begin (9600);
  Serial.println ("I2C scanner. Scanning...");
  byte count = 0;
  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
```

```
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0)
      {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      delay (1);  // maybe unneeded?
      } // end of good response
  } // end of for loop
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");
}  // end of setup
void loop() {}
```

Крок 3. Запрограмуйте Arduino Uno. Передавати потрібно два показники: температури й вологості. Для одночасної передачі використовуйте масив data. Зверніть увагу на рядок з відправкою показників:

```
#include <Wire.h>
#include <dht11.h> // підключаємо бібліотеку для датчика

byte DHT11_PIN = 11;
dht11 DHT; // повідомляємо на якому порту буде датчик
byte data [2]; // Масив для переміщення даних
void setup() {
 Serial.begin(9600); // підключаємо монітор порту
 Wire.begin();
}
// Функція моніторингу помилок датчика
void chek () {
  int chk;
  chk = DHT.read(DHT11_PIN);    // Читання даних
  switch (chk){
  case DHTLIB_OK:
    break;
  case DHTLIB_ERROR_CHECKSUM:
    Serial.println("Checksum error, \t");
    break;
```

```
    case DHTLIB_ERROR_TIMEOUT:
      Serial.println("Time out error, \t");
      break;
    default:
      Serial.println("Unknown error, \t");
      break;
    }
  }
  void loop() {
    chek (); // Перевіряємо датчик на помилки

    data[0] = DHT.humidity;
    data[1] = DHT.temperature;
    Wire.beginTransmission(8);
  // Відправляємо показники вологості і температури
    Wire.write (data, sizeof data);
    Wire.endTransmission();
    delay(500);
  }
```

Крок 4. Запрограмуйте Arduino Nano. Створіть масив для зчитування переданих з Arduino Uno показників датчика. Заповнюйте масив у циклі:

```
  #include <Wire.h>
  #include <LiquidCrystal_I2C.h> //Бібліотека для дисплея
  LiquidCrystal_I2C lcd(0x27,16,2);   // Задаємо адресу і розмірність
дисплея
  volatile byte buf [2]; //масив для зчитування показників з Arduino Uno
  byte degree[8] =         // Бітова маска символу градуса
  {
    B00111,
    B00101,
    B00111,
    B00000,
    B00000,
    B00000,
    B00000,
  };

  void setup() {
   Wire.begin(8);
   Wire.onReceive(receiveEvent);
   Serial.begin(9600);
```

```
  lcd.init();                      // Ініціалізація lcd
  lcd.backlight();                 // Включаємо підсвічування
  lcd.createChar(1, degree);       // Створюємо символ під номером 1
}

void receiveEvent(int howMany) {
// Зчитування показників в масив через цикл
for (byte i = 0; i < howMany; i++)
    {
    buf [i] = Wire.read ();
    }
}

void loop() {
  // Виводимо показники вологості і температури
  lcd.begin(16,2);
  lcd.setCursor(0,0); //Встановлення символу першого рядка
  lcd.print("Humidity ");
  lcd.print(buf[0]);
  lcd.setCursor(0,1); //Другий рядок
  lcd.print("Temperature ");
  lcd.print(buf[1]);
  delay (500);
}
```

В результаті роботи програми на індикаторі має відобразитись виміряна температура за допомогою Arduino Uno.

3.8.5 Тест для самоперевірки

1. У кожен момент часу провідним на шині може бути...
    1) 127 пристроїв;
    2) два пристрої;
    3) кілька пристроїв;
    4) тільки один пристрій.
2. Бібліотека Wire використовує...
    1) 7-бітну адресацію;
    2) 8-бітну адресацію;
    3) 32-бітну адресацію;
    4) 64-бітну адресацію.

3. SCL призначена для...
    1) передачі даних і команд;
    2) ініціалізації пристрою;
    3) передачі синхроімпульсів;
    4) запобігання колізій.
4. SDA призначена для...
    1) запобігання колізій;
    2) передачі синхроімпульсів;
    3) передачі даних і команд;
    4) ініціалізації пристрою.
5. Функція, яка підключає контролер у шині I2C як *master*...
    1) receiveEvent;
    2) Wire.begin ();
    3) Wire.begin (address);
    4) Wire.send (data, quantity).
6. Код реєстрації функції, яка має бути викликана, коли *slave* отримає дані від майстра...
    1) receiveEvent;
    2) Wire.begin ();
    3) Wire.onReceive (handler);
    4) Wire.onRequest (handler).

Відповіді на тест

1	2	3	4	5	6
4)	1)	3)	3)	2)	3)

3.9 Керування «Розумним будинком», використовуючи хмарний сервіс Blynk

3.9.1 Теоретичні відомості про платформу Blynk

Blynk (рис. 3.52) – це платформа для систем iOS і Android, за допомогою якої можна керувати Arduino, Raspberry Pi ESP8266 та іншими контролерами через Інтернет. Blynk не прив'язаний до якоїсь певної плати або модуля. Незалежно від типу підключення до мережі (Wi-Fi, Ethernet або ESP8266), Blynk забезпечить онлайн-доступ до пристрою.

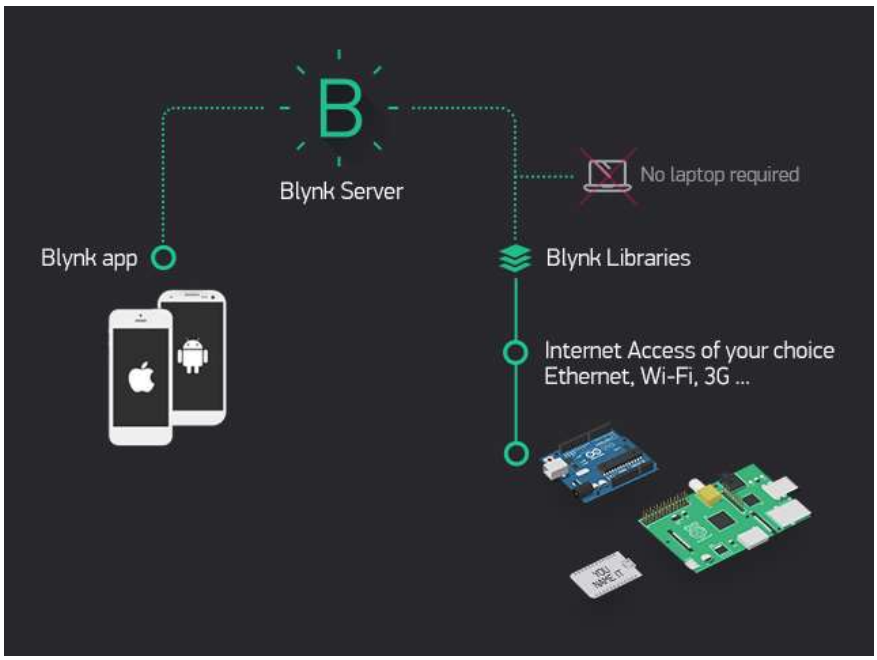


Рисунок 3.52 – Архітектура Blynk

Для використання Blynk необхідно:

а) завантажити застосунок Blynk для Android або IOS;

б) підключити Blynk і плату, використовуючи Auth Token:

1) створити новий обліковий запис у Blynk App;
2) створити новий проєкт. Потім обрати плату і з'єднання, яке будете використовувати;
3) після того, як проєкт буде створений, Blynk відправить Auth Token електронною поштою;
4) перевірте свою поштову скриньку і знайдіть Auth Token;

в) встановити бібліотеку Blynk в Arduino IDE, скачати її можна за посиланням: https://github.com/blynkkk/blynk-library/releases/tag/v0.5.4;

г) розробити першу програму, скачавши код за посиланням: https://examples.blynk.cc/?board=ESP8266&shield=ESP8266%20WiFi&example=GettingStarted%2FBlynkBlink;

д) підключити Auth Token:

1) у прикладі Blynk знайдіть рядок: char auth[] = "YourAuthToken";
2) змініть його за допомогою Auth Token (він має бути у вашій поштовій скриньці після створення проєкту в застосунку Blynk): Blynk): char auth[] = "53e4da8793764b6197fc44a673ce4e21";
3) запрограмуйте мікроконтролер. Відкрийте Монітор порту. Результат виконання прикладу програми показано на рис. 3.53.

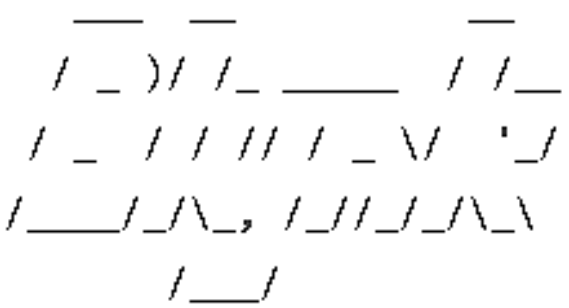


Рисунок 3.53 – Результат виконання прикладу програми

3.9.2 Керування компонентами «Розумного будинку», використовуючи RFID-модуль, технологію Wi-Fi і сервіс Blynk

Розробимо проєкт керування сервомотором, використовуючи RFID-модуль, технологію Wi-Fi і сервіс Blynk.

Крок 1. Підключіть до модуля Arduino:

– сервомотор до 2-го порту Arduino;

– RGB-світлодіод до 4, 5, 6 порту Arduino;

– модуль Wi-Fi на 0 і 1 порти (RX, TX);

– RFID-модуль на 9, 10, 11, 12, 13 порти.

Крок 2. Встановіть застосунок Blynk. Зареєструйтеся в ньому. Скопіюйте з пошти Auth Token. Створіть віджет кнопки в сервісі Blynk.

На рисунку 3.54 показано процес створення нового проєкту.

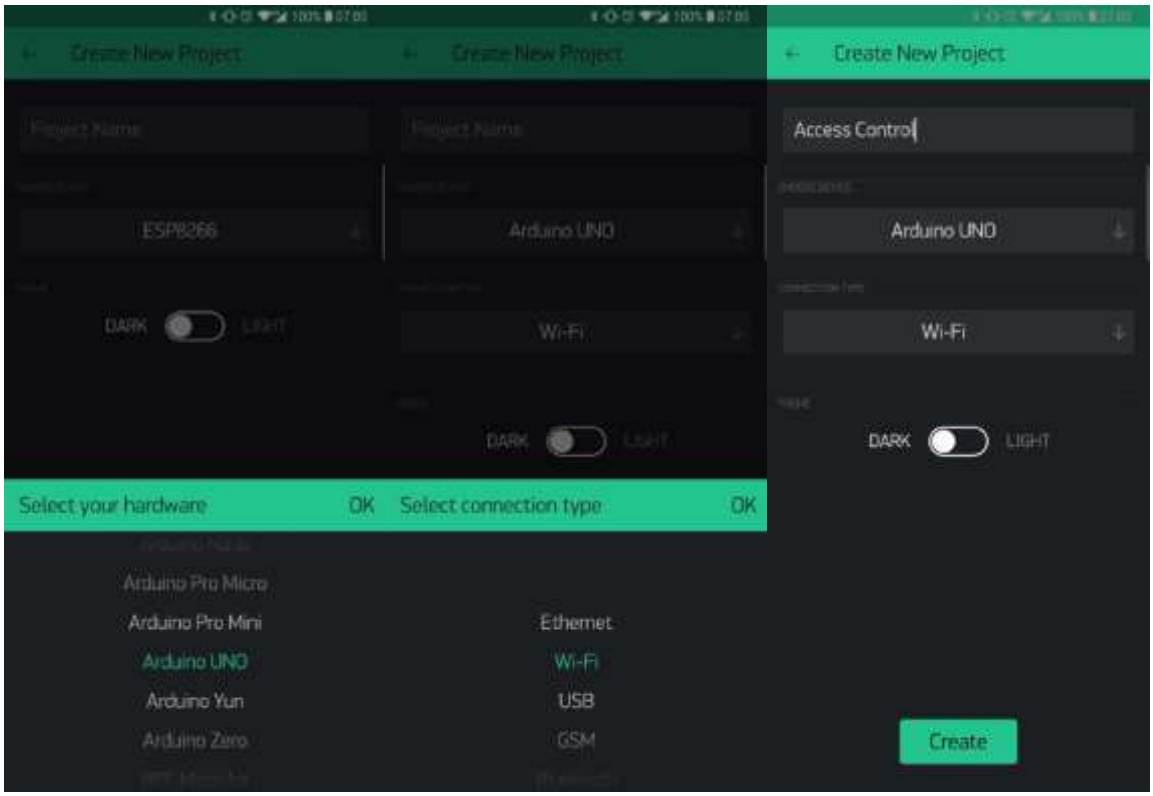


Рисунок 3.54 – Процес створення нового проєкту

На рисунку 3.55 показано приклад обирання необхідних компонентів

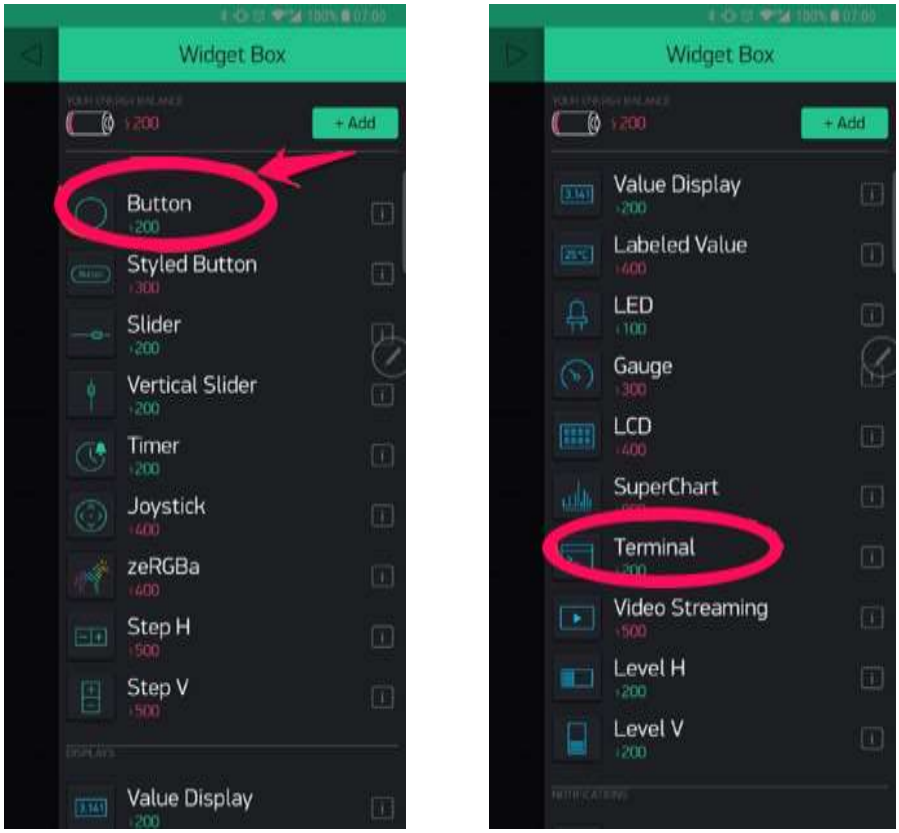


Рисунок 3.55 – Приклад обирання необхідних компонентів

На рисунку 3.56 наведено приклад налаштування терміналу.

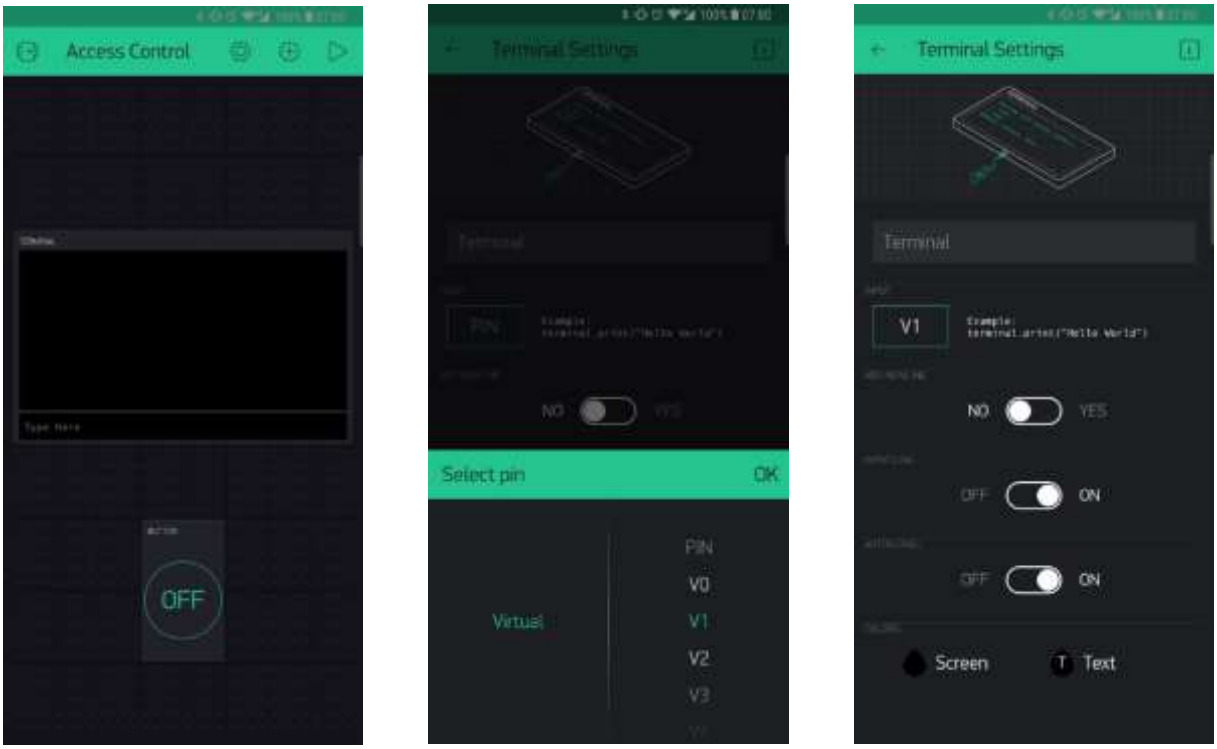


Рисунок 3.56 – Приклад налаштування терміналу

На рисунку 3.57 наведено приклад налаштування кнопки.

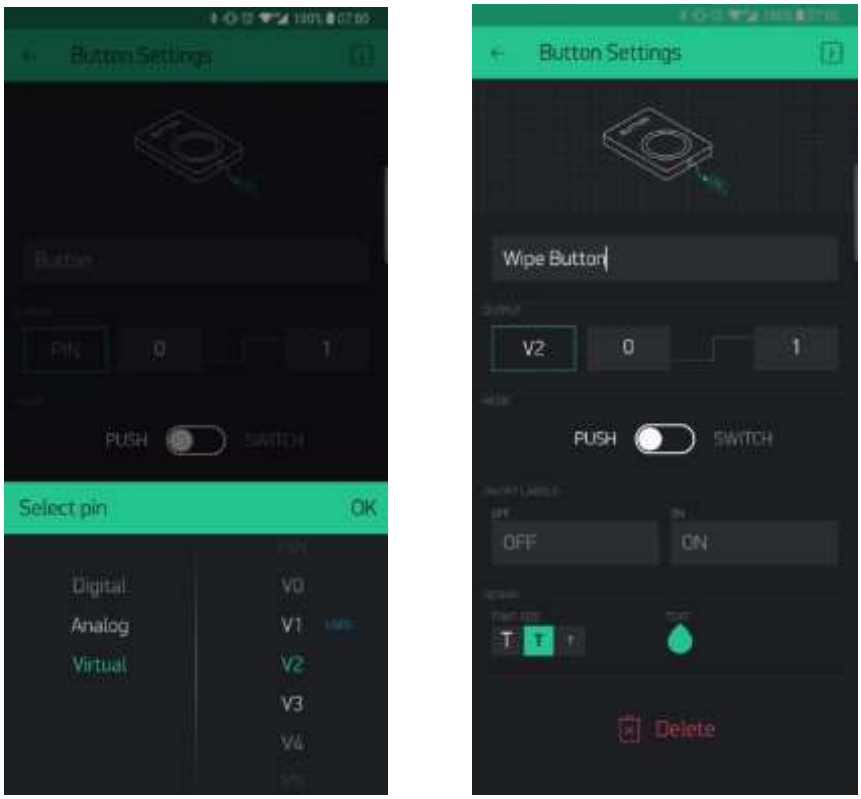


Рисунок 3.57 – Приклад налаштування кнопки

Натискаючи на кнопку «Play», вмикаємо термінал (рис. 3.58).

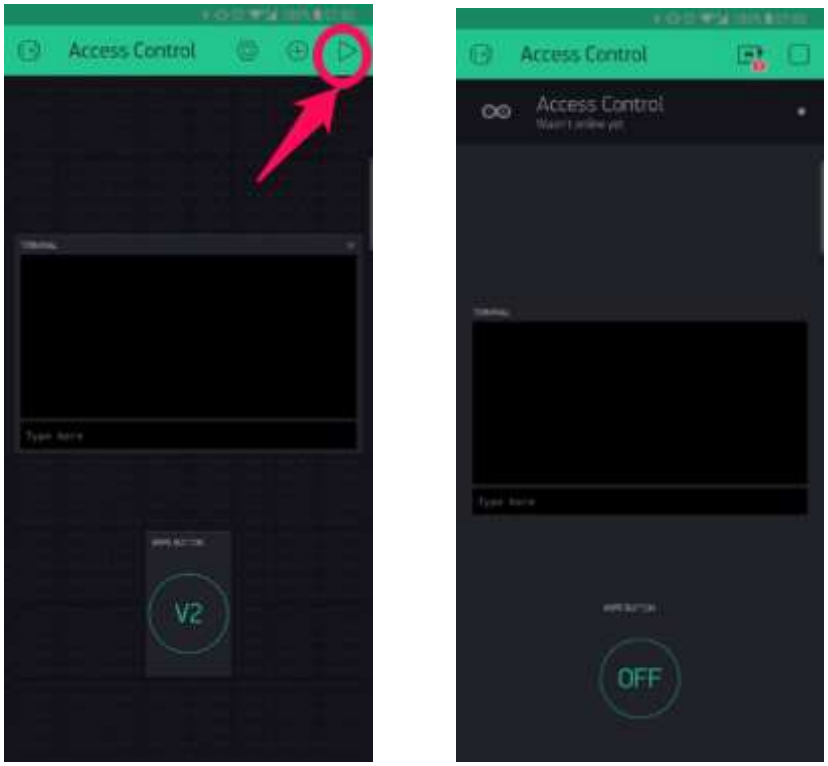


Рисунок 3.58 – Запуск програми

Для перевірки роботи застосунку потрібно завантажити вихідний код до Arduino із підключеним до нього модулем Wi-Fi. Повний текст коду наведено в додатку А.

Результат виконання програми показано на рисунку 3.59.

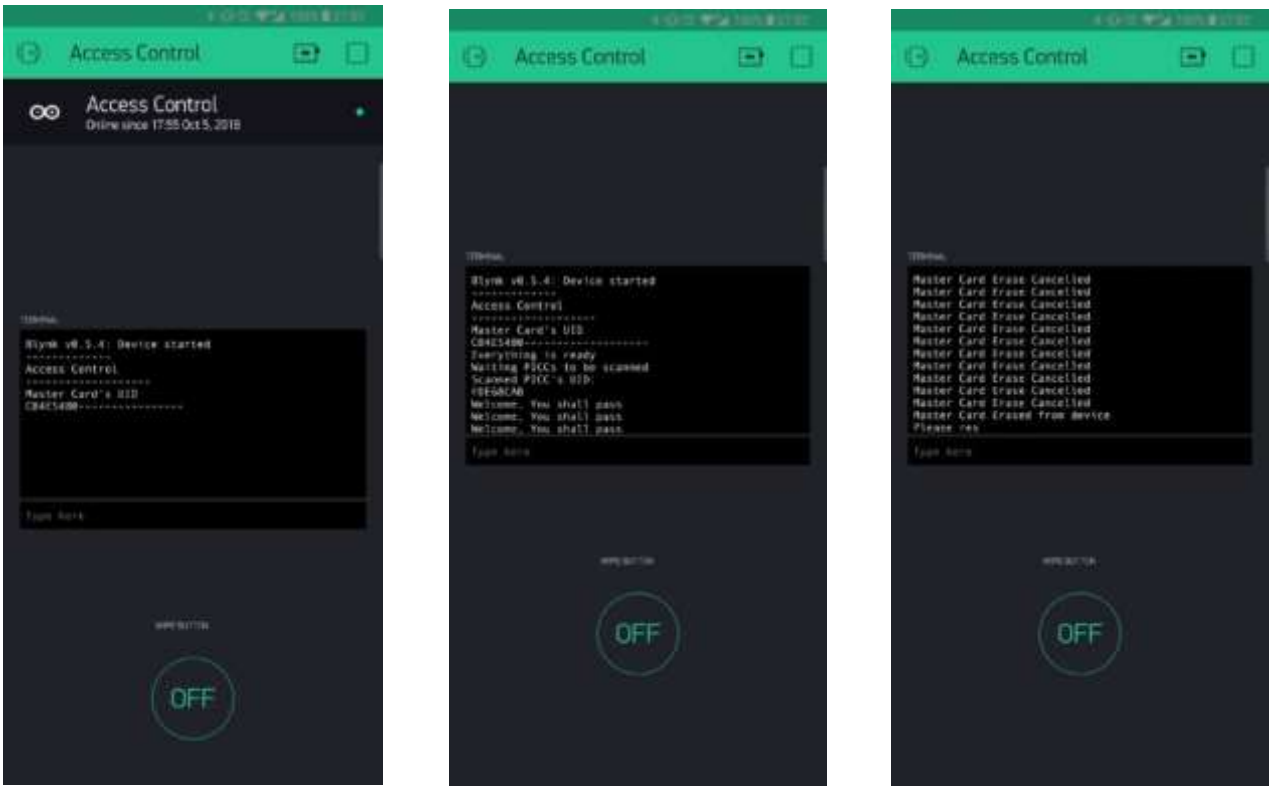


Рисунок 3.59 – Результат виконання програми

Протокол виконання команд на боці Arduino показано на рисунку 3.60.

Рисунок 3.60 – Протокол виконання команд на боці Arduino

На рисунку 3.61 показано приклад отримання даних від RFID-мітки

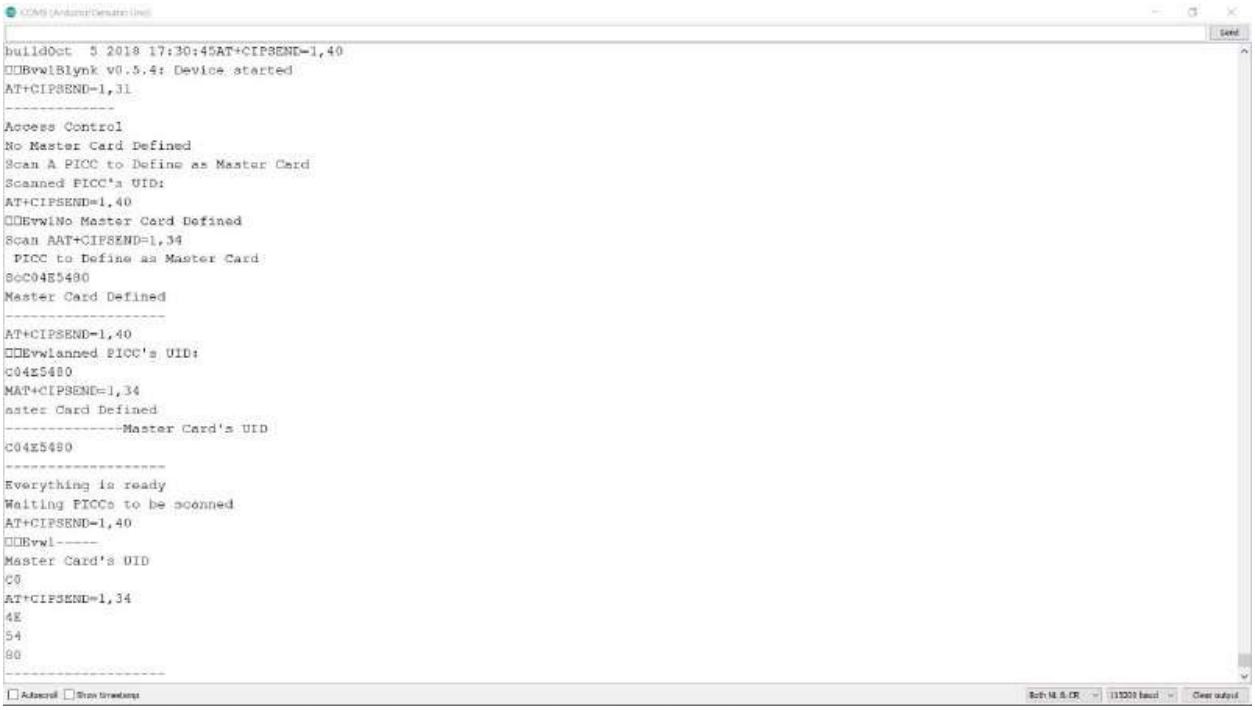


Рисунок 3.61 – Приклад отримання даних від RFID-мітки

3.9.3 Тест для самоперевірки

1. Функція підтвердження відправки в термінал Blynk…
   1) terminal.write();
   2) terminal.print();
   3) terminal.flush();
   4) terminal. read ().
2. Команда підключення віртуального терміналу до Virtual Pin V1 застосунку Blynk…
   1) BLYNK_WRITE(V1);
   2) terminal.print("You said:");
   3) terminal.write(param.getBuffer(), param.getLength());
   4) WidgetTerminal terminal(V1).
3. Команда підключення MFRC522 за SPI-протоколом…
   1) #include <SPI.h>;
   2) SPI.begin();
   3) SPI MOSI;
   4) SPI SCK.
4. Команда ініціалізації MFRC522…
   1) MFRC522 mfrc522(SS_PIN, RST_PIN);
   2) mfrc522.PCD_Init();

3) mfrc522.PICC_IsNewCardPresent();
4) mfrc522.PICC_ReadCardSerial().

5. Функція відображення конфігурації MFRC522 модуля…
   1) mfrc522.PCD_Init();
   2) MFRC522 mfrc522(SS_PIN, RST_PIN);
   3) ShowReaderDetails();
   4) mfrc522.uid.uidByte[i].
6. Виведення інформації про версії Blynk на віджет Терміналу після приєднання до серверу Blynk...
   1) Blynk.begin(auth, wifi, ssid, pass);
   2) BLYNK_WRITE(V1);
   3) terminal.println(F("Blynk v" BLYNK_VERSION ": Device started"));
   4) Blynk.run().

Відповіді на тест

1	2	3	4	5	6
3)	4)	2)	2)	3)	3)

3.10 Робота з годинником реального часу

3.10.1 Теоретичні відомості про годинник реального часу

Іноді під час роботи з Arduino потрібно контролювати реальний час, причому навіть тоді, коли плата знеструмлена. Для цього застосовують спеціальні мікросхеми годинники (рис. 3.62).

Рисунок 3.62 – Модуль годинника реального часу

Мікросхема DS1302 містить у своєму складі крім власне годинника ще 31 байт статичного ОЗУ (оперативна пам'ять). Передача даних здійснюється за допомогою послідовного інтерфейсу. Послідовний інтерфейс для передачі

даних використовує одну сигнальну лінію, якою інформаційні біти передаються один за одним послідовно. Звідси назва інтерфейсу і порту.

Годинники можуть працювати в одному з двох форматів 12\24 і містять індикатор AM/PM. 12-годинний формат обчислення часу припускає розбивку 24 годин, що становлять добу, на два 12-годинних інтервали, які позначаються a.m. (Лат. Ante meridiem дослівно – «до полудня») і p.m. (Лат. Post meridiem дослівно – «після полудня»). Час подається у вигляді секунд, хвилин, годин, дні тижня, дата, місяць і рік. У разі якщо поточний місяць складається менше ніж з 31 дня, то DS1302 автоматично перерахує кількість днів у місяці, враховуючи високосність поточного року.

Для роботи з годинником застосовується бібліотека ds1302.h.

Функції бібліотеки:

– DS1302 (uint8_t ce_pin, uint8_t data_pin, uint8_t sclk_pin) – створює об'єкт для модуля годинника реального часу. В параметрах вказуються номери пінів, до яких підключений модуль;

– halt(bool enable) – встановлює годинник у робочий (false) або неробочий (true) режим;

– writeProtect(bool enable) – знімає (false) або встановлює (true) захист від запису;

– setDOW(uint8_t dow) – встановлює день тижня. Може задаватися цифрами від 1 (понеділок) до 7 (неділя) або терміновими константами: назва дня тижня англійською мовою у верхньому регістрі;

– setTime(uint8_t hour, uint8_t min, uint8_t sec) – встановлює час. У параметрах на першому місці вказується година (0–23), потім хвилина (0–59) і секунда (0–59);

– setDate(uint8_t date, uint8_t mon, uint16_t year) – встановлює дату: число (1–31), місяць (1–12), рік (2000–2099);

– getTime() – дорівнює (повертає) поточному значенню часу типу Time;

– getTimeStr(uint8_t format) – повертає поточний час у вигляді рядка. Має два формати подання: FORMAT_LONG (чч: мм: сс) і FORMAT_SHORT (чч: мм);

– getDateStr(uint8_t Yformat, uint8_t Dformat, char divider) – повертає поточну дату в одному з форматів: Yformat – визначає характер відображення року FORMAT_LONG (рррр) або FORMAT_SHORT (рр); Dformat – задає формат відображення дати: FORMAT_LITTLEENDIAN (дд.мм.рррр) – за замовчуванням; FORMAT_BIGENDIAN (рррр.мм.дд);

FORMAT_MIDDLEENDIAD (мм.дд.рррр); Divider – задає символ, який розділяє (за замовчуванням – крапка);

– getDOWStr(uint8_t format) – повертає день тижня у форматі: FORMAT_LONG – повні назви днів тижня; FORMAT_SHORT – скорочені назви;

– getMonthStr(uint8_t format) – повертає місяць у форматах: FORMAT_LONG – повні назви; FORMAT_SHORT – скорочені назви.

Структури

На відміну від масивів, які містять елементи одного типу, структури можуть бути різних типів.

Структура – це множина, яка складається з декількох елементів, кожен з яких може мати свій тип.

Кожен елемент структури називається полем. Для створення структури необхідно спочатку створити для неї свій тип даних, а потім оголосити змінну цього типу.

Приклад використання:

```
struct human // ім'я структури
{
// назви полів
char name[25];
int age;
char gender[7];
};
human D; // оголошення змінної типу структура
```

Звернення до поля структури відбувається за допомогою операції «.».

У загальному вигляді:

ім'я_змінної. ім'я_поля

Наприклад, запис «D.name» – звернення до поля name.

Методи ініціалізації структури:

– при оголошенні – значення задаються після імені змінної у фігурних дужках через кому: human D = {«Даша», 16, «жіночий»};

– у програмі:

```
D.name = «Даша»;
D.age = 16;
D.gender = «жіночий»;
```

Змінна D містить інформацію про одну людину. Для задання інформації про декілька людей можна використовувати масив структур.

3.10.2 Практична реалізація

Реалізуємо виведення поточного часу на LCD-дисплей. Для цього можна використовувати стандартний скетч як шаблон.

Зберемо схему, яка складається з модуля годинника реального часу і LCD -дисплея. Схема підключення модуля годинника показана на рис. 3.63.

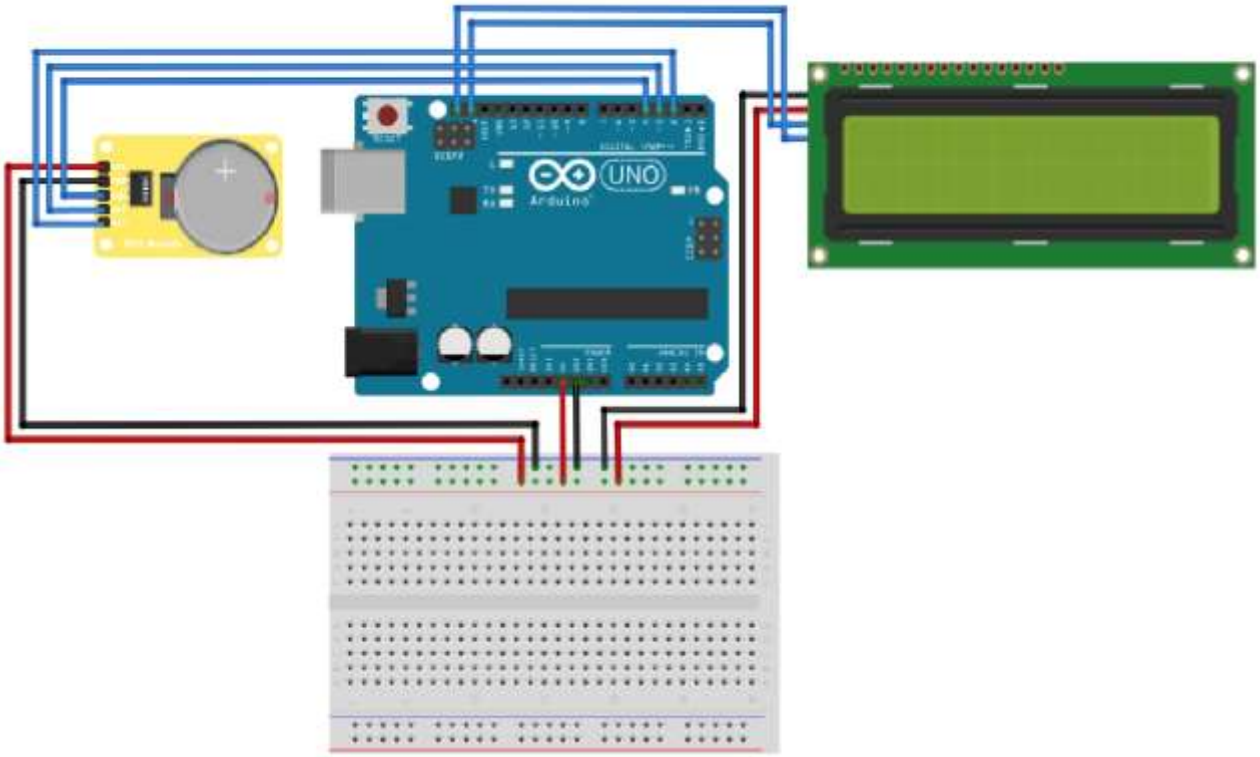


Рисунок 3.63 – Схема підключення модуля годинника

Напишемо такий код для Arduino:

```
#include<Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS1302.h>
DS1302 rtc(2, 3, 4);
LiquidCrystal_I2C lcd(0x27, 20, 4);
void setup()
{
rtc.halt(false);
rtc.writeProtect(false);
lcd.begin();
lcd.backlight();
rtc.setDOW(3); // Set Day-of-Week to WEDNESDAY
rtc.setTime(11, 56, 0); // Set the time to 11:56:00 (24hr format)
rtc.setDate(9, 11, 2022); // Set the date to August 8, 11, 2022
}
```

```
void loop()
{
lcd.setCursor(4, 0);
lcd.print(rtc.getTimeStr());
lcd.setCursor(4, 1);
lcd.print(rtc.getDOWStr(FORMAT_LONG));
lcd.setCursor(4, 2);
lcd.print(rtc.getDateStr());
delay (1000);
}
```

В результаті роботи на екрані має бути виведена вказана дата.

Наступним прикладом буде реалізація будильника, використовуючи годинник реального часу, LCD – дисплей і п’єзовипромінювач. Цей будильник можна налаштувати через послідовний порт, надсилаючи сигнали w, s, a, d (вгору, вниз, вліво, вправо відповідно).

Для реалізації проєкту зберіть схему, яка складається з модуля годинника реального часу, LCD – дисплея і п’єзовипромінювача (рис. 3.64).

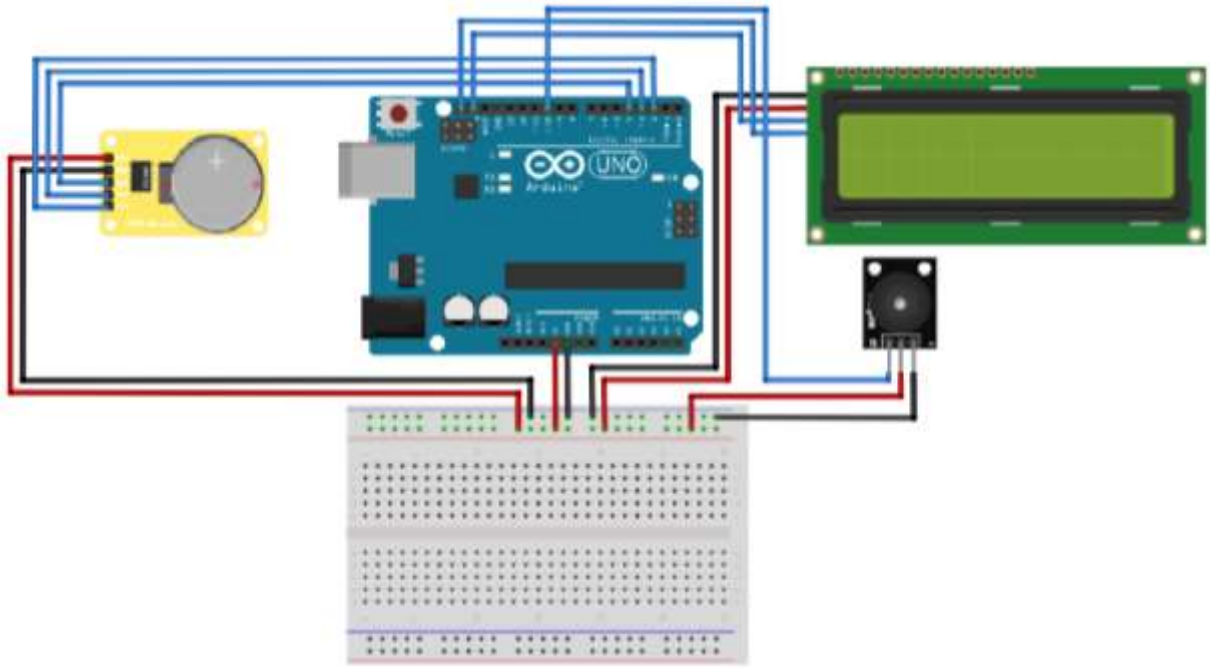


Рисунок 3.64 – Схема годинника з будильником

Код програми для Arduino такий:

```
#include<Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS1302.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

```
      // Init the DS1302
      // DS1302: CE (rst) pin -> Arduino Digital
      // I/O (dat) pin -> Arduino Digital
      // SCLK pin -> Arduino Digital
      DS1302 rtc(2, 3, 4);
      char val = ' ';
      Time tt;
      // h, h, m, m, s, s, alarmOn
      int alarm[7] = {1, 2, 0, 1, 0, 0, 1};
      int alarmMax[7] = {2, 9, 5, 9, 5, 9, 1};
      int alarmLen = 7;
      int alarmPosition = 5;
      boolean alarmSignal = false;
      boolean blink = true;
      int buzPin = 10;
      int hh = 0;
      int mm = 0;
      int ss = 0;

      void setup()
      {
        lcd.begin();
        lcd.backlight();
        randomSeed(analogRead(0));
        Serial.begin(9600);
      // Set the clock to run-mode, and disable the write protection
        rtc.halt(false);
        rtc.writeProtect(false);
      //  The  following  lines  can  be  commented  out  to  use  the  values
already stored in the DS1302
      // rtc.setDOW(FRIDAY); // Set Day-of-Week to FRIDAY
        rtc.setTime(12, 34, 0); // Set the time to 12:34:00 (24hr format)
        rtc.setDate(9, 11, 2016); // Set the date to August 9, 11, 2016
        pinMode(buzPin, OUTPUT);
      }
      void lcdPrintBlinkedDig(int pos)
      {
         if ((pos == alarmPosition) and (blink == 0))
          {
           lcd.print(« «);
          }
         else
          {
           lcd.print(alarm[pos]);
          }
```

```
}
void lcdPrintBlinkedStr()
{
   if ((6 == alarmPosition) and (blink == 0))
    {
     lcd.print(« «);
    }
   else
    {
     if (alarm[6] == 0)
      {
         lcd.print(«OFF»);
       }
   else
    {
      lcd.print(«ON «);
     }
   }
}
void loop()
{
   lcd.setCursor(4, 0);
   lcd.print(rtc.getTimeStr());
   lcd.setCursor(4, 1);
   lcd.print(«AL: «);
   lcdPrintBlinkedDig(0);
   lcdPrintBlinkedDig(1);
   lcd.print(«:»);
   lcdPrintBlinkedDig(2);
   lcdPrintBlinkedDig(3);
   lcd.print(«:»);
   lcdPrintBlinkedDig(4);
   lcdPrintBlinkedDig(5);
   lcd.print(« «);
   lcdPrintBlinkedStr();
   lcd.setCursor(4, 2);
   lcd.print(rtc.getDateStr());
   delay (250);
   blink = not blink;
   if (Serial.available())
    {
       val = Serial.read();
//lcd.print(val);
       switch (val)
    {
```

```
   case 'd':
               alarmPosition ++;
               if (alarmPosition>= alarmLen)
                {
                    alarmPosition = 0;
                 }
   break;
   case 'a':
                alarmPosition -i;
                if (alarmPosition <0)
                {
                     alarmPosition = alarmLen - 1;
                 }
   break;
   case 'w':
                  alarm[alarmPosition] ++;
                 if (alarm[alarmPosition]> alarmMax[alarmPosition])
                 {
                      alarm[alarmPosition] = 0;
                  }
   break;
   case 's':
                alarm[alarmPosition] -i;
                if (alarm[alarmPosition] <0)
                 {
                    alarm[alarmPosition] = alarmMax[alarmPosition];
                  }
    break;
}
// 23 hour max!
// 25, 26, 27,... -> 20
   if ((alarm[0] == 2) and (alarm[1]> 3))
    {
        alarm[1] = 0;
     }   }
// check alarm time
      if (alarm[6] != 0) // alarm = ON
{
// Get data from the DS1302
           tt = rtc.getTime();
           hh = alarm[0]*10 + alarm[1];
           mm = alarm[2]*10 + alarm[3];
            ss = alarm[4]*10 + alarm[5];
//lcd.setCursor(10, 0);
//lcd.print(hh);
```

```
            if ((tt.hour == hh) && (tt.min == mm))
          {
      alarmSignal = true;    }    }
    else
      {
      alarmSignal = false;        }
// show alarm signal
      lcd.setCursor(13, 0);
       if ((alarmSignal)&&(blink))
         {
             lcd.print(«ALARM!»);
              tone(buzPin, 400);
              delay(1000);            }
         else
            {
                lcd.print(« «);
noTone(buzPin);
           }}
```

Таким чином, вивчивши теоретичну частину даного уроку, ви ознайомилися з поняттям структури і алгоритмами вирішення завдань з використанням структур, розглянули будову і принципи роботи годинника реального часу. Розглянули бібліотеку DS1302.h, яка використовується для програмного керування годинником реального часу в програмі IDE Arduino.

Виконавши практичну частину, ви навчилися складати програми в IDE Arduino для керування годинником реального часу, а також правильно підключати даний пристрій.

3.10.3 Тест для самоперевірки

1. Передача даних годинника реального часу здійснюється за допомогою...
   1) графічного інтерфейсу;
   2) візуального інтерфейсу;
   3) тактильного інтерфейсу;
   4) послідовного інтерфейсу.
2. Скорочення a.m. означає...
   1) до полудня;
   2) полудень;
   3) після полудня;
   4) вечір.
3. Скорочення p.m. означає...
   1) до полудня;
   2) полудень;
   3) після полудня;
   4) вечір.

4. Функція getTime() повертає...
    1) поточне значення після полудня;
    2) поточне значення дати;
    3) поточне значення часу;
    4) поточне значення місяця.
5. Функція getMonthStr() повертає...
    1) поточне значення після полудня;
    2) поточне значення дати;
    3) поточне значення часу;
    4) поточне значення місяця.
6. Структура – це...
    1) множина, яка складається з декількох елементів, кожен з яких може мати свій тип;
    2) абстрактний тип даних з дисципліною доступу до елементів «перший прийшов – першим вийшов»;
    3) абстрактний тип даних, що являє собою упорядкований набір значень, в якому деяке значення може зустрічатися більше одного разу;
    4) структура даних, яка зберігає набір значень.

Відповіді на тест

1	2	3	4	5	6
4)	1)	3)	3)	4)	1)

3.11 Робота з матричною клавіатурою

3.11.1 Організація роботи з матричною клавіатурою

Коли необхідно подавати кілька команд на Arduino, найпростіше застосувати для цього необхідну кількість звичайних тактових кнопок. Втім якщо цих кнопок більше ніж дві, то виникає питання про оптимізацію підключення. Припустимо, потрібно підключити 16 кнопок. Для цього необхідно відвести 16 контактів, але, як правило, це буде марнотратством апаратних ресурсів мікроконтролера. Аби скоротити кількість використовуваних виводів, застосовують спеціальне з'єднання кнопок у вигляді матриці (рис. 3.65). У цьому випадку кнопки будуть об'єднані в структуру, яка складається з рядків і стовпців. Якщо контролеру необхідно дізнатися стан клавіш, проводиться послідовне опитування цієї структури, на це витрачається певний час.

Рисунок 3.65 – Матрична клавіатура

Приклад схеми з'єднань клавіатури зображений на рисунку 3.66.

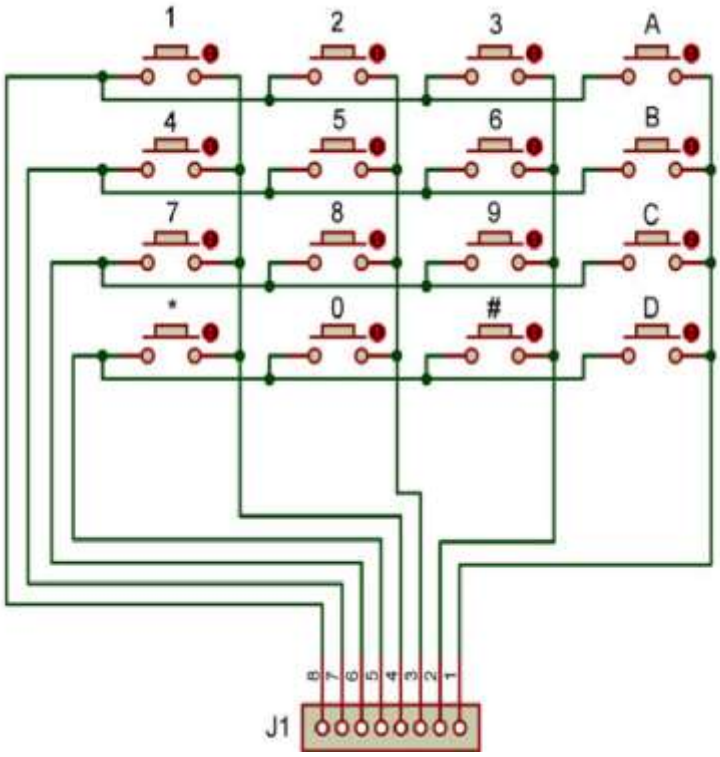


Рисунок 3.66 – Схема матричної клавіатури

Суть сканування полягає в тому, що спочатку подається сигнал високого рівня на перший рядок, на інші подається сигнал низького рівня, а зі стовпців відбувається зчитування. Якщо на першому рядку була натиснута кнопка, то під час зчитування стовпця буде видно, в якому стовпці це сталося. Далі активується наступний ряд, а попередній буде вимкнено і операція зчитування триває. Коли будуть перевірені всі рядки, операція почнеться знову з першого.

Більшість проєктів, які складніші, ніж просто миготіння світлодіодів (проєкти, які використовують тільки виводи), вимагають від користувача будь-якого введення. У багатьох випадках для отримання призначеного для користувача введення використовуються кнопки. Для Arduino існує

бібліотека Keypad.h, яка здатна обробляти введення з матричної клавіатури і проста у використанні.

Ця бібліотека усуває необхідність використання зовнішніх резисторів, які підтягують, оскільки вона використовує вбудовані в мікросхему резистори, які підтягують, а також обробляє/встановлює високий опір на всіх виводах невикористаного стовпчика. Вона, по суті, сканує стовпець за стовпцем. Виконується це шляхом установки низького рівня на виводі поточного стовпчика і читання значень виводів рядків для цього стовпця, а потім переходить до наступного стовпця і так далі, поки не просканує всі підключені/призначені виводи.

Крім використання вбудованих резисторів, які підтягують, бібліотека також обробляє брязкіт контактів. Бібліотека не використовує затримки; замість цього вона періодично використовує вбудовану функцію millis() Arduino і визначає, як довго була натиснута кнопка, і чи була зміна стану певної кнопки. Без затримок код виконується більш ефективно і не споживає обчислювальні ресурси, усуваючи необхідність обробки брязкоту контактів з використанням програмних затримок.

Даний тип клавіатур застосовується повсюди. У кожному комп'ютері клавіатура організована, як правило, так само, тільки кількість рядків і стовпців набагато більше.

3.11.2 Оператор вибору switch

Ви вже знайомі з умовним оператором if-else. Він дозволяє перевіряти кілька умов. Якщо умов багато, то зручно застосовувати оператор switch().

Синтаксис цього оператора такий:

```
switch(змінна)
{
case константа1:
<оператори>
break;
case константа2:
<оператори>
break;
default:
<оператори>
}
```

Даний оператор послідовно перевіряє, чи дорівнює змінна, яка стоїть після switch, константам, що знаходяться після слова case. Якщо змінна дорівнює константі, то виконуються дії, які знаходяться у відповідній секції. У кінці кожної секції, яка починається ключовим словом case, ставиться оператор break. Як і за якого значення має вести себе програма, визначається програмістом за допомогою так званої case-константи. Під час порівняння на символ потрібно символ укладати в одинарні лапки, інакше порівняння буде з числом.

Гілку default можна не описувати. Вона виконується, якщо жодна з умов не задоволена. Нижче показаний приклад використання оператора switch().

```
{
case 1: Serial.print(«one»); break;
case 2: Serial.print(«two»); ; break;
default: Serial.print(«not one or two»);
}
```

Важливо зазначити, що даний оператор може виконувати вибір тільки якщо аргумент дорівнює одному з перерахованих значень case, тобто перевірка виразів типу (x <7) неможлива.

3.11.3 Приклад введення з матричної клавіатури символів і відображення їх на LCD-дисплеї

Кнопкам відповідають такі символи (рис. 3.67).

1	2	3	+
4	5	6	-
7	8	9	/
*	0	#	=

Рисунок 3.67 – Розташування символів на клавіатурі

Крок 1. Зібрати схему, де будуть підключені матрична клавіатура і LCD-дисплей. Схема підключення матричної клавіатури зображена на рисунку 3.68.

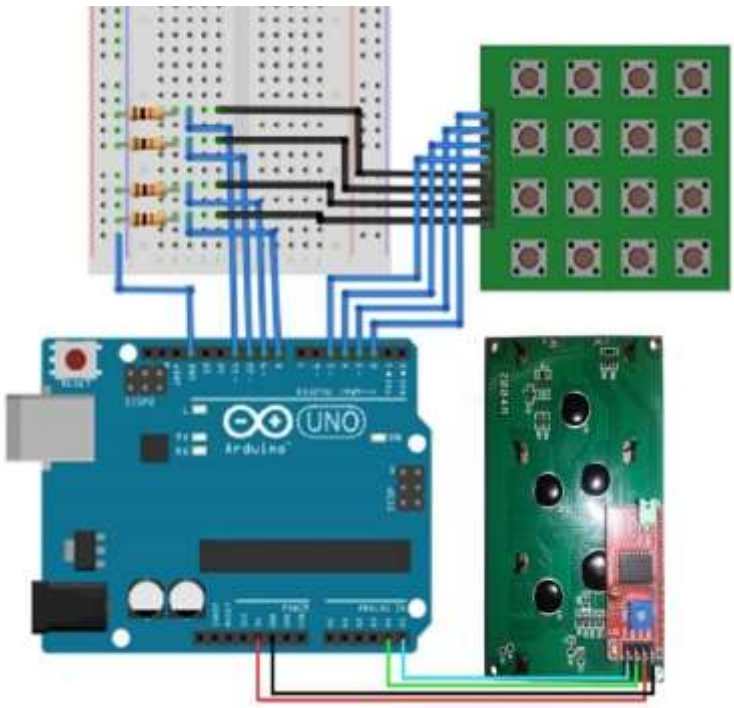


Рисунок 3.68 – Схема підключення матричної клавіатури

Для перевірки проєкту напишемо такий код:

```
#include<Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 40, 2);
introws[4] = {5, 4, 3, 2}; // вказуємо піни рядків
intcols[4] = {11, 10, 9, 8}; // і стовпців
char symbol[4][4] =
{
{'1', '2', '3', '+'},
{'4', '5', '6', '-'},
{'7', '8', '9', '/'},
{'*', '0', '#', ' = '}
};
void setup()
{
lcd.begin();
lcd.backlight();
lcd.print(«Enter symbol!»);
for (int i = 0; i <4; i ++)
{
pinMode(cols[i], INPUT);
pinMode(rows[i], OUTPUT);
};
}
```

```
void loop()
{
for(inti = 0; i<4; i ++)
{
//цикл для переходу по всіх рядках
digitalWrite(rows[i], HIGH); //подаємо на поточний рядок високий рівень
for(int j = 0; j <4; j ++)
{
//цикл для переходів по всіх стовпцях
if(digitalRead(cols[j]) == HIGH)
{
//якщо рівень високий, то кнопка натиснута
lcd.clear();
lcd.print(symbol[j][i]); //виводимо символ натиснутої кнопки
}
}
//вимикаємо високий рівень для пройденого рядка
digitalWrite(rows[i], LOW);
}
}
```

Вивчивши теоретичну частину даного розділу, ми детально розглянули пристрій матричної клавіатури, дізналися схему з'єднання кнопок на її модулі. Ознайомилися з оператором вибору switch, вивчили синтаксис і реалізацію даного оператора.

Виконавши практичну частину, навчилися правильно підключати матричну клавіатуру, реалізовувати введення символів з матричної клавіатури і відображення їх на LCD-дисплеї.

3.11.4 Тест для самоперевірки

1. switch – це…
   1) оператор циклу;
   2) оператор вибору;
   3) оператор функції;
   4) оператор умови.
2. Команда case визначає ...
   1) тип змінної, зазначеної в блоці switch;
   2) поведінку програми, якщо обране користувачем значення не увійшло до множини значень, які порівнюються;

3) множину альтернатив у виборі;
4) поведінку програми, якщо значення змінної, зчитаної у блоці switch, збіглося із значенням вказаної константи.

3. Команда case визначає ...
   1) тип змінної, зазначеної у блоці switch;
   2) поведінку програми, якщо значення змінної, зчитаної у блоці switch, збіглося із значенням зазначеної константи;
   3) поведінку програми, якщо обране користувачем значення не увійшло до множини значень, які порівнюються;
   4) множину альтернатив у виборі.
4. Бібліотека Keypad.h…
   1) обробляє виведення даних на дисплей;
   2) обробляє введення з матричної клавіатури;
   3) обробляє введення з клавіатури комп'ютера;
   4) обробляє виведення даних на монітор порту.
5. Бібліотека Keypad.h…
   1) усуває необхідність використання зовнішніх резисторів, які підтягують;
   2) керує підключеними резисторами, які підтягують;
   3) активує підключені резистори, які підтягують;
   4) усуває необхідність використання резисторів плати Arduno.
6. Бібліотека Keypad.h…
   1) встановлює високий опір на всіх виводах невикористаного стовпця матриці клавіатури;
   2) встановлює високий опір на всіх виводах невикористаного рядка матриці клавіатури;
   3) встановлює низький опір на всіх виводах невикористаного рядка матриці клавіатури;
   4) встановлює низький опір на всіх виводах невикористаного стовпця матриці клавіатури.

Відповіді на тест

1	2	3	4	5	6
2)	4)	3)	2)	1)	1)

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Patel, Keyur & Patel, Sunil & Scholar, P & Salazar, Carlos. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges.
2. Невлюдов І.Ш. Технологія програмування промислових контролерів в інтегрованому середовищі CODESYS: Навчальний посібник / І.Ш. Невлюдов, С.П. Новоселов, О.В. Сичова. – Харків: ХНУРЕ, 2019 . – 286 с.
3. What is the internet of things (IoT)? – Режим доступу https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT – 07.05.2022 р.
4. Internet of Things – Режим доступу https://en.wikipedia.org/wiki/ Internet_of_things – 07.05.2022 р.
5. Seneviratne Pradeeka. Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power. Apress, 2019. – 315 p.
6. Particle is a fully-integrated IoT platform that offers everything you need to deploy an IoT product – Режим доступу: https://www.particle.io – 07.05.2022 р.
7. Parihar, Yogendra Singh. (2019). Internet of Things and Nodemcu A review of use of Nodemcu ESP8266 in IoT products. 6. 1085.
8. Afifie, Nur & Wong Yoon Khang, Adam & Ja'afar, Abd & M Amin, A Fairuz & Alsayaydehahmad, Jamil & Indra, Win & Herawan, Safarudin & Ramli, Arnidza. (2021). Evaluation Method of Mesh Protocol over ESP32 and ESP8266. Baghdad Science Journal. 18. 1397. 10.21123/bsj.2021.18.4(Suppl.).1397.
9. Patel, Shruti & Kanawade, Shailaja. (2017). Internet of Things Based Smart Home with Intel Edison. 10.1007/978-981-10-2750-5_40.
10. Murugan, Kalpana & Narendranath, T. & Chand, S. (2022). An Adafruit Cloud-Based Health Monitoring Device Using IoT Technology. 10.1007/978-981-16-7330-6_2
11. Bräunl, Thomas. (2022). Arduino. 10.1007/978-981-16-0804-9_3
12. Bell, Charles. (2021). Introducing the Arduino. 10.1007/978-1-4842-7234-3_2
13. Mathe, Sudha & Pamarthy, Ashok & Kondaveeti, Hari & Vappangi, Suseela. (2022). A Review on Raspberry Pi and its Robotic Applications. 10.1109/AISP53593.2022.9760590
14. James, Alice & Seth, Avishkar & Mukhopadhyay, Subhas. (2022). Programming Arduino for IoT System. 10.1007/978-3-030-85863-6_5

15. Patil, Ashish. (2022). Supervisory Control and Data Acquisition for Motor Operation using Arduino. International Journal for Research in Applied Science and Engineering Technology. 10. 2602–2605. 10.22214/ijraset.2022.41845

16. L298. Dual Full Bridge Driver – Режим доступу: https://www.st.com/en/motor-drivers/l298.html – 07.05.2022 р.

17. Двигун з редуктором 1:48 (двовісний). – Режим доступу: https://arduino.ua/prod662-motor-s-redyktorom-148-dvyh-osevoi – 07.05.2022 р.

18. Getting Started With Stepper Motor 28BYJ-48 – Режим доступу: https://create.arduino.cc/projecthub/debanshudas23/getting-started-with-stepper-motor-28byj-48-3de8c9 – 07.05.2022 р.

19. Parvathi, A. & Basha, S. & Kumar, M. & Bhagavan, B. & Reddy, P. (2021). Industrial Protection System using Arduino and IoT. Journal of Communication Engineering and its Innovations. 7. 10.46610/JOCEI.2021.v07i02.004

20. Arduino Ir Remote Using Tsop Sensor – Режим доступу: https://techatronic.com/arduino-ir-remote-using-tsop-sensor/ – 07.05.2022 р.

21. TSOP4838 IR Receiver 38 kHz – Режим доступу: https://www.pcboard.ca/tsop4838-ir-receiver – 08.05.2022 р.

22. Solis Pino, Andrés & Alonso, Jose & Moguel, Enrique & Vicente-Chicote, Cristina & Alegria, Julio & Ruiz, Pablo. (2022). Software Product Lines for Industrial Robots: A Pilot Case with Arduino.

23. Odaba, Alphaeus & Ngyarmunta, Alan & Ohemu, Monday. (2022). Development and Realization of an Ultrasonic Ranging Detection and Tracking Device. American Journal of Modern Physics. 11. 22–31.

24. Sansury, Hendryg. (2019). Ultrasonic Sonar Object and Range Detection Measurement Display using HC-SR04 Sensor on Arduino ATMEGA 2560. ACMIT Proceedings. 3. 49–55.

25. Jian, Huang. (2017). Design of Angle Detection System Based on MPU6050. 10.2991/icemc-17.2017.2

26. Hasan, Md & Masum, Md Hafizur Rahman & Chowdhury, Kantish. (2022). Bluetooth Controlled Arduino Based Autonomous Car.

27. Client-Server Model – Режим доступу: https://www.geeksforgeeks.org/client-server-model/ – 08.05.2022 р.

28. ThingSpeak for IoT Projects – Режим доступу: https://thingspeak.com/ – 08.05.2022 р.

29. Знайомство з модулем ESP8266 – Режим доступу: https://hobbytech.com.ua/знакомимся-с-модулем-esp8266-подробнее/ – 08.05.2022 р.

30. Devraj, Gowru & Kumar, Mallam & Reddy, Challa & Kumar, Gulla & Varma, Kota. (2022). IoT Enabled Shipping Containers with Location Tracking and Environment Monitoring. International Journal for Research in Applied Science and Engineering Technology.

31. Pessanha Santos, Nuno. (2009). Arduino - Ethernet Library Tutorial.

32. v, Nivedan. (2019). Weather Monitoring System using IOT with Arduino Ethernet Shield. International Journal for Research in Applied Science and Engineering Technology. 7. 218–221.

33. v, Nivedan. (2019). Weather Monitoring System using IOT with Arduino Ethernet Shield. International Journal for Research in Applied Science and Engineering Technology. 7. 218-221. 10.22214/ijraset.2019.1038

34. How to Post on Twitter from Arduino and ESP8266 IoT Project – Режим доступу: https://nandgeek.com/blog/how-to-post-on-twitter-from-arduino-esp8266-thingspeak-project/ – 08.05.2022 p.

35. Ali, Zubair. (2019). Arduino & RFID Based Home Security Automation with Arduino code.

36. Охоронні системи та системи відображення інформації: навч. посібник / Упоряд.: С.П. Новоселов. Харків: ХНУРЕ, 2007. – 260 с.

37. Concepción, Tomás & Rovetto, Carlos & Acosta, Elia. (2015). Modeling with Petri colored nets of Arduino RFID reader Modelado con Redes de Petri coloreadas de un lector RFID sobre Arduino.

38. Blum, Jeremy. (2019). The I2C Bus.

39. Brad, Ana & Brad, Maria. (2021). Development of a smart clothing product using an Arduino platform. International Journal of Advanced Statistics and IT&C for Economics and Life Sciences. 11. 38–61.

40. Blynk IoT platform: for businesses and developers – Режим доступу: https://blynk.io – 07.05.2022 p.

ДОДАТОК А
КОД ПРОГРАМИ ДЛЯ РОБОТИ З ПЛАТФОРМОЮ BLINK

```
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <EEPROM.h>     // Читання і запис UID картки з/на EEPROM
#include <SPI.h>        // RC522 модуль використовує SPI protocol
#include <MFRC522.h>  // Бібліотека для RC522
#include <Servo.h>
#define BLYNK_PRINT Serial
#define COMMON_ANODE
#ifdef COMMON_ANODE
#define LED_ON LOW
#define LED_OFF HIGH
#else
#define LED_ON HIGH
#define LED_OFF LOW
#endif
#define redLed 4    // Піни для світлодіода
#define greenLed 5
#define blueLed 6
boolean wipeB; // Ініціалізація кнопки видалення Майстер-карти
boolean isBusy = false;
boolean match = false;
//Ініціалізація режимів програми
boolean programMode = false;  // Режим додавання/видалення карти
boolean wipeMode = false; // Режим видалення картки
boolean readCardMode = false; // Режим читання картки
boolean normalMode = false; // Режим очікування
boolean replaceMaster = false;
uint8_t successRead;  // Змінна для результату збереження читання карти
int redLedState = 0;
int greenLedState = 0;
int blueLedState = 0;
uint32_t operationTime = 0;
int counter = 0;
byte storedCard[4];   // Масив байтів ID карти з EEPROM
byte readCard[4];   // Масив байтів ID карти з RFID модуля
byte masterCard[4];   // Масив байтів ID Майстер-карти з EEPROM
// Скопіювати Auth Token з пошти або з додатку Blynk (в настройках
проекту)
```

```
      char auth[] = "YourAuthToken";
      // Ім'я та пароль Wi-Fi мережі
      char ssid[] = "Name";
      char pass[] = "password";
      // Швидкість спілкування з модулем ESP8266
      #define ESP8266_BAUD 115200
      ESP8266 wifi(&Serial);
      // Створення об'єкта модуля MFRC522
      #define SS_PIN 10
      #define RST_PIN 9
      MFRC522 mfrc522(SS_PIN, RST_PIN);
      // Підключення віртуального терміналу до Virtual Pin V1 застосунку
Blynk
      WidgetTerminal terminal(V1);
      Servo servo;
      // Відправка команд з Терміналу на Arduino
      BLYNK_WRITE(V1)
      {
        // Якщо ввести "master" в віджет терміналу, то він відповість ID
Майстер-карти
        if (String("master") == param.asStr()) {
          Serial.println(F("--------------------"));
          terminal.println(F("--------------------"));
          terminal.println("You said : 'master'");
          Serial.println(F("Master Card's UID"));
          terminal.println(F("Master Card's UID"));
          for ( uint8_t i = 0; i < 4; i++ ) {            // Читання ID
Майстер-карти з EEPROM
            masterCard[i] = EEPROM.read(2 + i);     // Запис ID в масив
байтів masterCard
            Serial.print(masterCard[i], HEX);
            terminal.print(masterCard[i], HEX);
          }
        } else {
          terminal.print("You said:");
          terminal.write(param.getBuffer(), param.getLength());
          terminal.println();
        }
        // Підтвердження відправлення
        terminal.flush();
      }
      BLYNK_WRITE(V2) {
        wipeB = param.asInt();
```

```
}
void setup()
{
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  servo.attach(3);
  servo.write(0);    // Переконайтеся, що двері закриті
  digitalWrite(redLed, LED_OFF);
  digitalWrite(greenLed, LED_OFF);
  digitalWrite(blueLed, LED_OFF);
  redLedState = 0;
  blueLedState = 0;
  greenLedState = 0;
  Serial.begin(9600);
  delay(10);
  Serial.begin(ESP8266_BAUD);
  delay(10);
  SPI.begin();           // Підключення MFRC522 по SPI протоколу
  mfrc522.PCD_Init();    // Ініціалізація MFRC522
  Serial.println(F("Access Control"));
  ShowReaderDetails();  // Відображення конфігурації MFRC522 модуля
  wifi.setDHCP(1, 1);
  Blynk.begin(auth, wifi, ssid, pass);
  // Виведення інформації про версії Blynk на віджет Терміналу
після приєднання до сервера Blynk
  terminal.println(F("Blynk v" BLYNK_VERSION ": Device started"));
  terminal.println(F("-------------"));
  terminal.println(F("Access Control"));
  terminal.flush();
  // Перевірка: якщо Майстер-карта записана, якщо ні - надати
користувачеві вибір своєї Майстер-карти
  if (EEPROM.read(1) != 143) {
    Serial.println(F("No Master Card Defined"));
    terminal.println(F("No Master Card Defined"));
    Serial.println(F("Scan A PICC to Define as Master Card"));
    terminal.println(F("Scan A PICC to Define as Master Card"));
    do {
      successRead = getID();      // Надаємо successRead значення 1
якщо карта зчитана, інакше 0
    }
    while (!successRead);     // Очікування успішного читання карти
    for ( uint8_t j = 0; j < 4; j++ ) {
```

```
            EEPROM.write( 2 + j, readCard[j] );  // Запис відсканованого
ID карти в EEPROM, почати з адреси 3
          }
          EEPROM.write(1, 143);
          Serial.println(F("Master Card Defined"));
          terminal.println(F("Master Card Defined"));
        }
        Serial.println(F("-------------------"));
        terminal.println(F("-------------------"));
        Serial.println(F("Master Card's UID"));
        terminal.println(F("Master Card's UID"));
        for ( uint8_t i = 0; i < 4; i++ ) {             // Читання ID
Майстер-карти з EEPROM
          masterCard[i] = EEPROM.read(2 + i);
          Serial.print(masterCard[i], HEX);
          terminal.print(masterCard[i], HEX);
        }
        Serial.println("");
        Serial.println(F("-------------------"));
        Serial.println(F("Everything is ready"));
        Serial.println(F("Waiting PICCs to be scanned"));
        terminal.println(F("-------------------"));
        terminal.println(F("Everything is ready"));
        terminal.println(F("Waiting PICCs to be scanned"));
        normalMode = true;
      }
      void loop()
      {
        Blynk.run();
        if (normalMode && !readCardMode && !wipeMode && !programMode) {
          successRead = getID();
          if (!successRead) normalModeOn();
          else {
            readCardMode = true;
            normalMode = false;
          }
        }
        if (readCardMode && !normalMode && !wipeMode && !programMode) {
          if ( isMaster(readCard)) {     // Якщо ID відсканованої карти
збігається з ID Майстер-карти запуск programMode
            programMode = true;
            readCardMode = false;
            successRead = 0;
```

```
            Serial.println(F("Hello Master - Entered Program Mode"));
            terminal.println(F("Hello Master - Entered Program Mode"));
            uint8_t count = EEPROM.read(0);     // Зчитування кількості
записаних ID-карт
            Serial.print(F("I have "));
            Serial.print(count);
            Serial.print(F(" record(s) on EEPROM"));
            Serial.println("");
            Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
            Serial.println(F("Scan Master Card again to Exit Program
Mode"));
            Serial.println(F("-----------------------------"));
            terminal.println(F("I have "));
            terminal.print(count);
            terminal.print(F(" record(s) on EEPROM"));
            terminal.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
            terminal.println(F("Scan Master Card again to Exit Program
Mode"));
            terminal.println(F("-----------------------------"));
          }
          else {
            if ( findID(readCard) ) { // Перевірка зчитаної карти
              if (!isBusy) {
                operationTime = millis();
                isBusy = true;
                granted(3000);         // Відкрити двері
              } else {
                granted(3000);
              }
            }
            else {
              if (!isBusy) {
                operationTime = millis();
                isBusy = true;
                denied();
              }
              else {
                denied();
              }
            }
          }
        }
        if (programMode && !normalMode && !readCardMode && !wipeMode) {
```

```
          successRead = getID();
          if (!successRead) {
            if (!isBusy) operationTime = millis();
          }
          else {
            if ( isMaster(readCard) ) { // Вихід з програми запису
              Serial.println(F("Master Card Scanned"));
              Serial.println(F("Exiting Program Mode"));
              Serial.println(F("-----------------------------"));
              programMode = false;
              normalMode = true;
              return;
            }
            else {
              if ( findID(readCard) ) { // Якщо сканована карта відома,
видалення її
                Serial.println(F("I know this PICC, removing..."));
                deleteID(readCard);
                Serial.println("-----------------------------");
               Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
              }
              else {       // Якщо сканована карта невідома, додавання її
                Serial.println(F("I do not know this PICC, adding..."));
                writeID(readCard);
                Serial.println(F("-----------------------------"));
               Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
              }
            }
          }
        }
        // Якщо кнопка стирання натиснута протягом 10 секунд, Майстер-
карта очищається
        if (wipeB != LOW && !wipeMode) { // Якщо кнопка натиснута
          //  Демонстрація звичайного режиму. Засвічується червоний
світлодіод, якщо картка не зчитується
          digitalWrite(redLed, LED_ON); // Ввімкнення червоного світлодіода
          digitalWrite(greenLed, LED_OFF); // Вимкнення зеленого світлодіода
          digitalWrite(blueLed, LED_OFF); // Вимкнення синього світлодіода
          redLedState = 1;
          greenLedState = 0;
          blueLedState = 0;
          Serial.println(F("Wipe Button Pressed"));
          Serial.println(F("Master Card will be Erased! in 10 seconds"));
```

```
        programMode = false;
        readCardMode = false;
        normalMode = false;
        wipeMode = true;
      }
      if (wipeMode && !normalMode && !readCardMode && !programMode) {
        if (!isBusy) {
          operationTime = millis();
          isBusy = true;
        } else {
          if (monitorWipeButton(10000)) {     // Give user enough time
to cancel operation and if button still be pressed, wipe EEPROM
            while (1);
          }
        }
      }
    }
    ////////////////////////////////////////        Надання    доступу
/////////////////////////////////
    void granted ( uint32_t setDelay) {
      Serial.println(F("Welcome, You shall pass"));
      terminal.println(F("Welcome, You shall pass"));
      uint32_t now = (uint32_t) operationTime;
      digitalWrite(blueLed, LED_OFF); // Вимкнення синього світлодіода
      digitalWrite(redLed, LED_OFF); // Вимкнення червоного світлодіода
      redLedState = 0;
      blueLedState = 0;
      if ((uint32_t)millis() - now < setDelay) servo.write(180);     //
Розблокування дверей. Залишити двері відкритими кілька секунд
      if ((uint32_t)millis() - now < setDelay + 1000) {
        digitalWrite(greenLed, LED_ON); // Ввімкнення зеленого світлодіода
        greenLedState = 1;
      }
      else {
        isBusy = false;
        servo.write(0);    // Блокування двері
        readCardMode = false;
        normalMode = true;
      }
    }

    ////////////////////////////////////////      Блокування      двері
/////////////////////////////////
```

```
  void denied() {
    Serial.println(F("You shall not pass"));
    terminal.println(F("You shall not pass"));
    uint32_t now = (uint32_t) operationTime;
    digitalWrite(greenLed, LED_OFF); // Вимкнення зеленого світлодіода
    digitalWrite(blueLed, LED_OFF); // Вимкнення синього світлодіода
    greenLedState = 0;
    blueLedState = 0;
    if ((uint32_t)millis() - now < 1200) {
      digitalWrite(redLed, LED_ON); // Ввімкнення червоного світлодіода
      redLedState = 1;
    }
    else {
      Serial.println(operationTime);
      isBusy = false;
      readCardMode = false;
      normalMode = true;
    }
  }
  ////////////////////////////////////////       Зчитування       ID
/////////////////////////////////
  uint8_t getID() {
    // Підготовка до читання карти
    if ( ! mfrc522.PICC_IsNewCardPresent()) { // Якщо нова картка
не притулена до RFID-зчитувача
      return 0;
    }
    if ( ! mfrc522.PICC_ReadCardSerial()) {
      return 0;
    }

    Serial.println(F("Scanned PICC's UID:"));
    terminal.println(F("Scanned PICC's UID:"));
    for ( uint8_t i = 0; i < 4; i++) {
      readCard[i] = mfrc522.uid.uidByte[i];
      Serial.print(readCard[i], HEX);
      terminal.print(readCard[i], HEX);
    }
    Serial.println("");
    terminal.println(F(""));
    mfrc522.PICC_HaltA(); // Зупинити читання
    return 1;
  }
```

```
void ShowReaderDetails() {
  // Зчитування версії програмного забезпечення MFRC522
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
  else if (v == 0x92)
    Serial.print(F(" = v2.0"));
  else
    Serial.print(F(" (unknown),probably a chinese clone?"));
  Serial.println("");
  // 0x00 или 0xFF - помилка в підключенні
  if ((v == 0x00) || (v == 0xFF)) {
    Serial.println(F("WARNING: Communication failure, is the
MFRC522 properly connected?"));
    terminal.println(F("WARNING: Communication failure, is the
MFRC522 properly connected?"));
    Serial.println(F("SYSTEM HALTED: Check connections."));
    terminal.println(F("SYSTEM HALTED: Check connections."));
    // Система RGB-візуалізації зупинена
    digitalWrite(greenLed, LED_OFF); // Вимкнути зелений світлодіод
    digitalWrite(blueLed, LED_OFF);  // Вимкнути синій світлодіод
    digitalWrite(redLed, LED_ON);  // Ввімкнути червоний світлодіод
    while (true); // do not go further
  }
}
//////////////////////////////////////      Звичайний      режим
/////////////////////////////////
void normalModeOn () {
  digitalWrite(blueLed,  LED_ON);     //  Якщо  синій  світлодіод
ввімкнений, то пристрій готовий до читання карти
  digitalWrite(redLed, LED_OFF);  // Вимкнути червоний світлодіод
  digitalWrite(greenLed, LED_OFF);  // Вимкнути зелений світлодіод
  redLedState = 0;
  greenLedState = 0;
  blueLedState = 1;
  servo.write(0);   // Блокування дверей
}

//////////////////////////////////////  Читання  ідентифікатора  з
EEPROM ///////////////////////////
```

```
    void readID( uint8_t number ) {
      uint8_t start = (number * 4 ) + 2;
      for ( uint8_t i = 0; i < 4; i++ ) {
        storedCard[i] = EEPROM.read(start + i);      // Зчитування
інформації з EEPROM в масив
      }
    }
    ////////////////////////////////////////   Додати   ідентифікатора
в EEPROM  ///////////////////////////////
    void writeID( byte a[] ) {
      if ( !findID( a ) ) {      // Перш ніж додати карту, потрібно
перевірити наявність її номера в базі
        uint8_t num = EEPROM.read(0);     // Кількість карт
        uint8_t start = ( num * 4 ) + 6;
        num++;               // Збільшення кількості карт на 1
        EEPROM.write( 0, num );     // Запис нової кількості карт
        for ( uint8_t j = 0; j < 4; j++ ) {
          EEPROM.write( start + j, a[j] );   // Запис значень масиву
в EEPROM в правильному положенні
        }
        successWrite();
        Serial.println(F("Successfully added ID record to EEPROM"));
        terminal.println(F("Successfully added ID record to EEPROM"));
      }
      else {
        failedWrite();
        Serial.println(F("Failed! There is something wrong with ID or
bad EEPROM"));
        terminal.println(F("Failed! There is something wrong with ID or
bad EEPROM"));
      }
    }
    ////////////////////////////////////////  Видалення  ID  з  EEPROM
///////////////////////////////
    void deleteID( byte a[] ) {
      if ( !findID( a ) ) {      // Перш ніж видалити карту з EEPROM,
потрібно перевірити, чи є вона в базі
        failedWrite();      // Якщо ні
        Serial.println(F("Failed! There is something wrong with ID or
bad EEPROM"));
        terminal.println(F("Failed! There is something wrong with ID or
bad EEPROM"));
      }
```

```
      else {
        uint8_t num = EEPROM.read(0);
        uint8_t slot;       // Ідентифікатор номера слота карти
        uint8_t start;      //  Позиція наступного слота
        uint8_t looping;    // Число повторень циклу
        uint8_t j;
        uint8_t  count  =  EEPROM.read(0);  //  Читання  першого  байта
EEPROM, який зберігає кількість карток
        slot = findIDSLOT( a );   // Номер слоту для видалення
        start = (slot * 4) + 2;
        looping = ((num - slot) * 4);
        num--;      // Зменшення кількості карт
        EEPROM.write( 0, num );   // Запис нової кількості карт
        for ( j = 0; j < looping; j++ ) {
          EEPROM.write( start + j, EEPROM.read(start + 4 + j));    //
Зсув значення масиву до 4-х місць раніше в EEPROM
        }
        for ( uint8_t k = 0; k < 4; k++ ) {         // Зсув
          EEPROM.write( start + j + k, 0);
        }
        successDelete();
        Serial.println(F("Successfully removed ID record from EEPROM"));
        terminal.println(F("Successfully removed ID record from EEPROM"));
      }
    }
    ////////////////////////////////////////      Перевірка      байта
//////////////////////////////////
    boolean checkTwo ( byte a[], byte b[] ) {
      if ( a[0] != 0 )      // Перевірка, що в масиві є інформація,
        match = true;       // яка на початку збігається
      for ( uint8_t k = 0; k < 4; k++ ) {
        if ( a[k] != b[k] ) // якщо a! = b, то встановити match = false
          match = false;
      }
      if ( match ) {      // Перевірка на збіг
        return true;
      }
      else  {
        return false;
      }
    }
    ////////////////////////////////////////        Пошук        слота
//////////////////////////////////
```

```
uint8_t findIDSLOT( byte find[] ) {
  uint8_t count = EEPROM.read(0);   // Читання першого байта з EEPROM
  for ( uint8_t i = 1; i <= count; i++ ) {    // який повторюється
один раз для кожного запису EEPROM
    readID(i);                  // Читання ідентифікатора в EEPROM,
він зберігається в storedCard[4]
    if ( checkTwo( find, storedCard ) ) {    // Перевірка читання
збереженої карти в EEPROM
      // Та ж інформація в масиві find [] ID карти
      return i;         // Номер слота карти
      break;          // Зупинка пошуку
    }
  }
}
//////////////////////////////////////   Пошук   ID   на   EEPROM
///////////////////////////////////
boolean findID( byte find[] ) {
  uint8_t count = EEPROM.read(0);   // Читання першого байта з EEPROM
  for ( uint8_t i = 1; i <= count; i++ ) {    // який повторюється
один раз для кожного запису EEPROM
    readID(i);             // Читання ідентифікатора в EEPROM, він
зберігається в storedCard[4]
    if ( checkTwo( find, storedCard ) ) {    // Перевірка читання
storedCard в EEPROM
      return true;
      break;  // Зупинка пошуку
    }
    else {    // Якщо немає, повернути  false
    }
  }
  return false;
}
//////////////////////////////////////     Запис     в     EEPROM
///////////////////////////////////
// Миготіння зеленого світлодіода 3 рази, щоб вказати успішний
запис в EEPROM
void successWrite() {
  digitalWrite(blueLed, LED_OFF); // Відключення синього світлодіода
  digitalWrite(redLed, LED_OFF); // Відключення червоного світлодіода
  redLedState = 0;
  blueLedState = 0;
  if (!isBusy) operationTime = millis();
  blinkLedWithoutDelay(greenLed, greenLedState, 6, 1200);
```

```
    }
    /////////////////////////////////////// Помилка запису в  EEPROM ///////////////////////////////////
    // Миготіння червоного світлодіода 3 рази, щоб вказати невдалий запис в EEPROM
    void failedWrite() {
      digitalWrite(blueLed, LED_OFF); // Відключення блакитного світлодіода
      digitalWrite(greenLed, LED_OFF); // Відключення зеленого світлодіода
      greenLedState = 0;
      blueLedState = 0;
      if (!isBusy) operationTime = millis();
      blinkLedWithoutDelay(redLed, redLedState, 6, 1200);
    }
    ///////////////////////////////////////  Видалення ID карти з EEPROM  ///////////////////////////////////
    //  Миготіння синього світлодіода 3 рази, щоб вказати успішне видалення  з EEPROM
    void successDelete() {
      digitalWrite(greenLed, LED_OFF); // Відключення блакитного світлодіода
      digitalWrite(redLed, LED_OFF); // Відключення червоного світлодіода
      redLedState = 0;
      greenLedState = 0;
      if (!isBusy) operationTime = millis();
      blinkLedWithoutDelay(blueLed, blueLedState, 6, 1200);
    }
    ////////////////////////  Перевірка чи є карта Майстер-картою ///////////////////////////////////
    // Перевірка, чи відповідає ідентифікатор основний картці
    boolean isMaster( byte test[] ) {
      if ( checkTwo( test, masterCard ) )
        return true;
      else
        return false;
    }
    void    blinkLedWithoutDelay(int    ledPin,    int    state,    int numberOfCycles, uint32_t interval) {
      isBusy = true;
      uint32_t now = (uint32_t) operationTime;
      int ledState = state;
      if ((uint32_t)millis() - now <= interval) {
        if ((uint32_t)millis() % (interval / numberOfCycles)) {
          if (ledState == LOW) ledState = HIGH;
          else ledState = LOW;
```

```
      digitalWrite(ledPin, ledState);
      counter++;
    }
  }
  else {
    counter = 0;
    isBusy = false;
  }
}
bool monitorWipeButton(uint32_t interval) {
  uint32_t now = (uint32_t) operationTime;
  if (wipeB != LOW && (uint32_t)millis() - now >= interval) {
    EEPROM.write(1, 0);
    Serial.println(F("Master Card Erased from device"));
    Serial.println(F("Please reset to re-program Master Card"));
    terminal.println(F("Master Card Erased from device"));
    terminal.println(F("Please reset to re-program Master Card"));
    isBusy = false;
    return true;
  }
  else {
    Serial.println(F("Master Card Erase Cancelled"));
    terminal.println(F("Master Card Erase Cancelled"));
    normalMode = true;
    wipeMode = false;
    return false;
  }
}
```

Електронне навчальне видання

НЕВЛЮДОВ Ігор Шакірович
АНДРУСЕВИЧ Володимир Анатолійович
НОВОСЕЛОВ Сергій Павлович
РЕЗНІЧЕНКО Олексій Георгійович

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ В УПРАВЛІННІ ПРИСТРОЯМИ НА МІКРОКОНТРОЛЕРАХ

Навчальний посібник

Рецензенти:

Филипенко О.І., д-р техн. наук, професор, декан факультету автоматики і комп'ютеризованих технологій ХНУРЕ;

Притчин С.Е., д-р техн. наук, професор кафедри автоматизації та інформаційних систем Кременчуцького національного університету імені Михайла Остроградського;

Альохіна С.В., д-р техн. наук, старший науковий співробітник, професор кафедри комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки ХНУРЕ.

Відповідальний випусковий І.Ш. Невлюдов
Редактор О.Г. Троценко
Комп'ютерна верстка Л.Ю. Свєтайло

План 2022 (перше півріччя), поз. 3

Підп. до використання 39.07.2022 Формат pdf. Обсяг даних 2,88 Mb

ХНУРЕ. Україна. 61166, Харків, просп. Науки, 14, E-mail: info@nure.ua

Підготовлено в редакційно-видавничому відділі ХНУРЕ
Свідоцтво суб'єкта видавничої справи ДК №1409 від 26.06.2003