

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА
ФІЗИКО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ

І. Є. Розанов, С. П. Сергієнко, Д. В. Чернов

**МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З КУРСУ
ІНТЕРНЕТ РЕЧЕЙ**

Вінниця
2019

УДК 004.738.5(076.5)

Р 64

*Рекомендовано до друку вченою радою
фізико-технічного факультету
(протокол № 5 від 24 грудня 2019 р.)*

Автори:

І. Є. Розанов, завідувач лабораторії;

С. П. Сергієнко, канд. техн. наук, доц. кафедри радіофізики та кібербезпеки;

Д. В. Чернов, канд. техн. наук, доц. кафедри радіофізики та кібербезпеки.

Рецензент: П. К. Ніколюк, д-р фіз.-мат. наук, проф., проф. кафедри комп'ютерних технологій.

Розанов І. Є., Сергієнко С. П., Чернов Д. В.

**Р 64 Методичні вказівки до виконання лабораторних робіт з
курсу Інтернет речей.** Вінниця : ДонНУ імені Василя Стуса, 2019.
60 с.

Представлені лабораторні роботи до курсу «Інтернет речей» рекомендовані для студентів вищих навчальних закладів за напрямками підготовки «Прикладна фізика. Технології Інтернету речей», «Кібербезпека» та «Комп'ютерні науки та інформаційні технології. Інтелектуальні інформаційні технології».

УДК 004.738.5(076.5)

© Розанов І. Є., 2019

© Сергієнко С. П., 2019

© Чернов Д. В., 2019

© ДонНУ імені Василя Стуса

ЗМІСТ

ВСТУП	4
1 Апаратно-програмна реалізація концепції інтернету речей на прикладі використання мікроконтролерів Arduino	5
1.1 Arduino як засіб для Інтернету речей	5
1.1.1. Сімейство Arduino	6
1.1.2. Датчик, актуатори, модулі	9
1.2. Мікрокомп'ютер Raspberry Pi	11
1.2.1. Raspberry Pi	11
1.2.2. Операційна система і програмне забезпечення	12
2 Інтернет речей. Архітектури і методи комунікацій	14
2.1. Інтернет речей	14
2.2. Архітектура інтернету речей	15
2.3. Технології комунікації між пристроями	17
Лабораторна робота № 1	19
Лабораторна робота № 2	29
Лабораторна робота № 3	51

ВСТУП

Інтернет речей (IoT) – це всесвітня павутина пристроїв, які з'єднані між собою та обмінюються інформацією між собою без втручання людини. Такі пристрої за допомогою мережі Інтернет транслюють основні дані у «хмару», звідки інші пристрої можуть збирати ці відомості для вирішення багатьох задач. Раніше ця технологія використовувалася тільки у «розумному домі» для керування освітленням, сигналізацією або температурою у приміщенні. Зараз до IoT вдаються в усіх галузях, де потрібна автоматизація процесів. Такими галузями є агросектор, логістика, маркетинг, медицина, транспорт, телекомунікації та багато інших. На сьогодні ринок Інтернету речей переживає період бурхливого зростання. Кількість пристроїв IoT вже перевищує чисельність населення Землі: у 2018 р. склало 22 млрд, а, за прогнозами, до 2022 р. перевищить 50 млрд. Проте, при такому бурхливому зростанні виникає багато нових проблем, пов'язаних насамперед з безпекою інформації, з якою оперує IoT-пристрій. Також важливими є й інші проблеми, такі як енергоспоживання, автономність, взаємовплив, узгодженість цих пристроїв між собою та відповідність до радіочастотних стандартів. Усі ці проблеми змушують подальше удосконалення IoT-пристроїв. У методичному посібнику наведено прості платформи, на яких реалізують IoT-пристрої, датчики та модулі, які використовуються в IoT-пристроях, програмне забезпечення таких пристроїв. Також описано архітектури Інтернету речей та методи комунікацій між пристроями. Посібник містить три лабораторні роботи, мета яких – ознайомити студентів з Інтернетом речей.

1 АПАРАТНО-ПРОГРАМНА РЕАЛІЗАЦІЯ КОНЦЕПЦІЇ ІНТЕРНЕТУ РЕЧЕЙ НА ПРИКЛАДІ ВИКОРИСТАННЯ МІКРОКОНТРОЛЕРІВ ARDUINO

1.1 Arduino як засіб для Інтернету речей

Arduino – загальна назва апаратно-програмних засобів для побудови систем автоматики і робототехніки, а також рішень в області Інтернету речей. Відмінною особливістю цієї платформи є простота програмування, детальна документація та реалізація апаратної платформи в різних варіаціях.

Для програмування мікроконтролерів використовується Arduino IDE (рис. 1) з підтримкою безлічі мов програмування, але фактично всі програми пишуться на C / C++, а компілюються і збираються за допомогою avr-gcc.

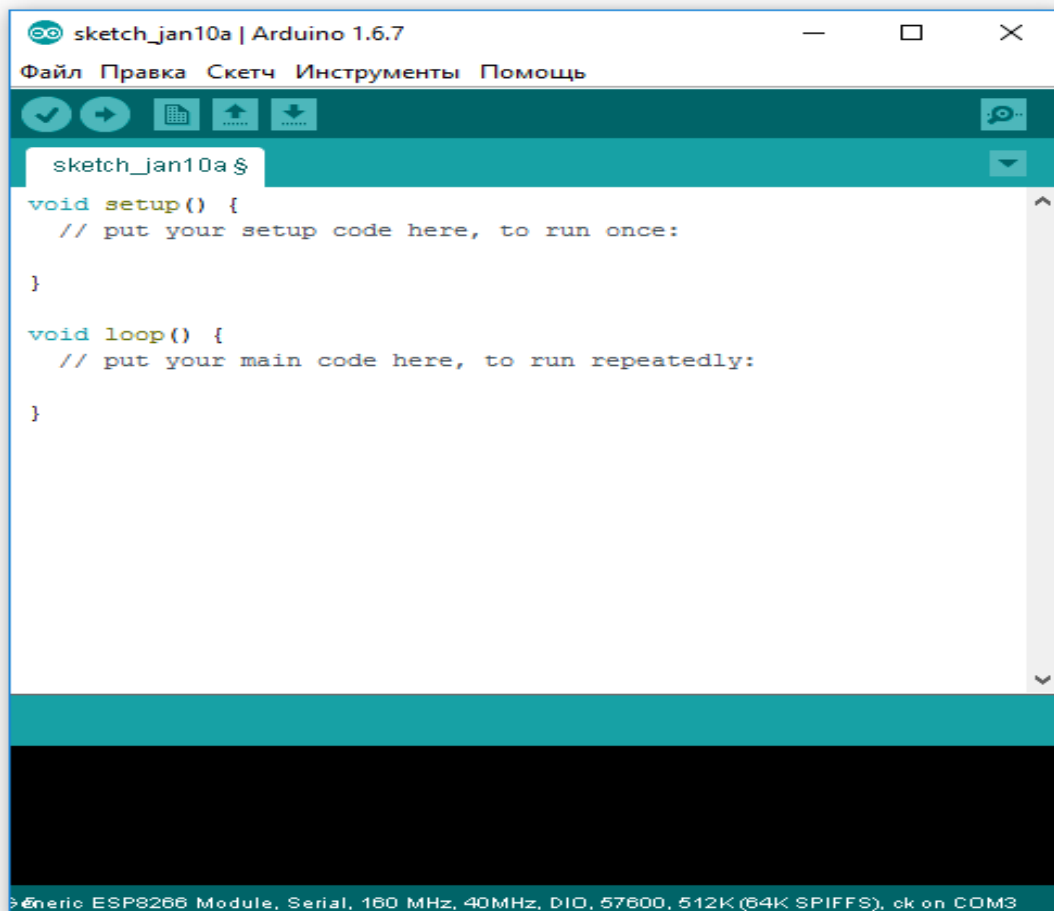


Рис. 1 – Середовище розробки Arduino IDE

В Arduino IDE за замовчуванням вже містяться приклади різних програм (Sketch) для перевірки працездатності мікроконтролера або швидкого створення своєї програми на базі існуючої, файли з програмою зберігаються з розширенням «.ino».

Найпростіша програма складається з двох функцій:

- `setup ()`: функція викликається одноразово при старті мікроконтролера;
- `loop ()`: функція викликається після `setup ()` в нескінченному циклі протягом всього часу роботи мікроконтролера.

Варто зауважити, що існують неофіційні, створені сторонніми виробниками, різні плагіни до різних середовищ розробки, наприклад: Eclipse plugin AVR-eclipse, Visualmicro для Microsoft Visual Studio тощо.

1.1.1 Сімейство Arduino

Існує багато різновидів мікроконтролерів, це пов'язано з відкритою ліцензією оригінальної версії Arduino, єдина умова якої – не використовувати офіційну назву Arduino в своїх мікроконтролерах без дозволу, щоб не заплутати кінцевого споживача в безлічі існуючих мікроконтролерів.

Офіційно існує кілька видів мікроконтролерів, що відрізняються між собою розміром і продуктивністю.

Arduino Mini являє собою невелику плату, спочатку засновану на ATmega168, але тепер підтримується ATmega328. Призначений для використання на макетах і вже готових продуктах. Він має 14 цифрових входних / вихідних пінів (із яких 6 можуть використовуватися як виходи PWM), 8 аналогових входів і 16 МГц кварцовий генератор. Його можна запрограмувати за допомогою USB Serial адаптера, або іншого послідовного адаптера USB, або RS232-TTL (рис. 2).

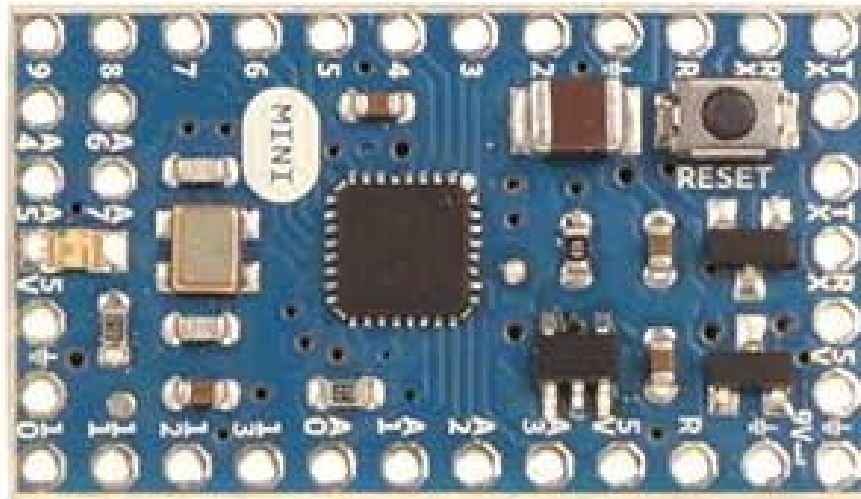


Рис. 2 – Плата Arduino Mini

Arduino Nano – компактна плата, наступна за розміром після Mini. Заснована на ATmega328P. Працює з кабелем USB-mini. Має 22 цифрові входи / виходи (із яких 6 можуть використовуватися як виходи PWM), а також 8 аналогових входів (рис. 3).

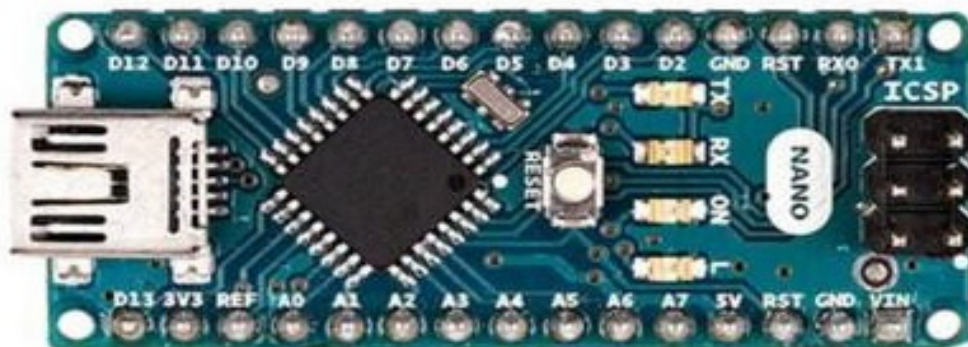


Рис. 3 – Плата Arduino Nano

Arduino Uno – середня за розміром плата, заснована на ATmega328P. Має окремий роз'єм живлення і повноцінний USB AB-порт, вибір живлення відбувається автоматично. На платі знаходяться 14 цифрових входів / виходів (із яких 6 можуть використовуватися як виходи PWM), 6 аналогових входів. Вважається найбільш придатною платою для знайомства з Arduino (рис. 4).



Рис. 4 – Плата Arduino Uno

Arduino Mega 2560 Rev3 – найбільша плата, яка дозволяє створити безліч складних проектів. Плата оснащена 54 цифровими входами / виходами, 16 аналоговими входами і досить великим об'ємом пам'яті. Має повноцінний USB AB-порт, окремий роз'єм живлення, вибір живлення відбувається автоматично. Така плата рекомендована для створення 3D-принтерів, роботів або інших складних проектів (рис. 5).



Рис. 5 – Плата Arduino MEGA 2560

1.1.2 Датчик, актуатори, модулі

Для Arduino і безлічі інших платформ проводиться велика кількість різних датчиків, сенсорів і модулів, що розширюють функціонал пристрою. Умовно їх можна розділити на три групи: датчики, актуатори і модулі.

Датчик – засіб вимірювань, призначений для вироблення сигналу вимірювальної інформації у формі, зручній для передачі, подальшого перетворення, обробки і (або) зберігання, але не піддається безпосередньому сприйняттю спостерігачем. Іншими словами, датчики (сенсори) дозволяють отримувати інформацію про навколишнє середовище і передавати її в зручному вигляді. Прикладом датчика може бути звичайний електронний термометр DS18B20 або LM35. Оскільки датчиків існує дуже багато, вибір конкретного датчика залежить від задачі, яку потрібно розв'язати.

Кожен датчик займає порти підключення на платі, це не критично, якщо ми працюємо з великими платами на кшталт Arduino Uno або MEGA, але якщо переважно використовувати плату меншого розміру, таку як Arduino Nano або Mini, що має істотно меншу кількість портів підключення, то необхідно розглянути більш просунуті (і дорогі) датчики, що поєднують в собі кілька датчиків. Прикладом суміщеного датчика може бути датчик GY-68 (рис. 6) (датчик атмосферного тиску і температури BMP180). Хоча цей датчик не показує високу точність вимірювання, однак демонструє компактність рішення задачі. Дорогі датчики мають високоякісні компоненти, що дозволяють отримувати більш точні вимірювання.



Рис. 6 – Датчик GY-68 (атмосферного тиску і температури BMP180)

Варто зазначити, що до датчиків належать різні пристрої для вимірювання світла, руху, теплової сигнатури або різного роду сигналів і тощо. У сукупності ці пристрої формують групу датчиків.

Актуатор – пристрій автоматичного управління або регулювання, що впливає на процес відповідно до командної інформації, що отримується. Складається з двох функціональних блоків: виконавчого пристрою (якщо виконавчий пристрій механічний, то його часто називають виконавчим механізмом) і регулюючого органу, наприклад, регулюючого клапана, і може оснащуватися додатковими блоками.

Цей пристрій дозволяє здійснювати будь-яку дію за певного сигналу. Прикладом такого пристрою можуть бути різні сервоприводи, наприклад, сервопривід MG995 (рис. 7).



Рис. 7 – Сервопривід MG995

Модуль – досить велика група, що дозволяє розширити функціонал плат Arduino. До них належать не тільки зовнішні модулі, які можна підключити до наявних роз'ємів, але і внутрішні, такі як мікроконтролер, ОЗУ, флеш-пам'ять тощо.

До зовнішніх пристроїв, які підключаються, можна віднести модулі зв'язку: WiFi (ESP8266 (рис. 8)), Bluetooth (HC-06), LoRa / LoRaWAN (RN2483), GPS тощо.

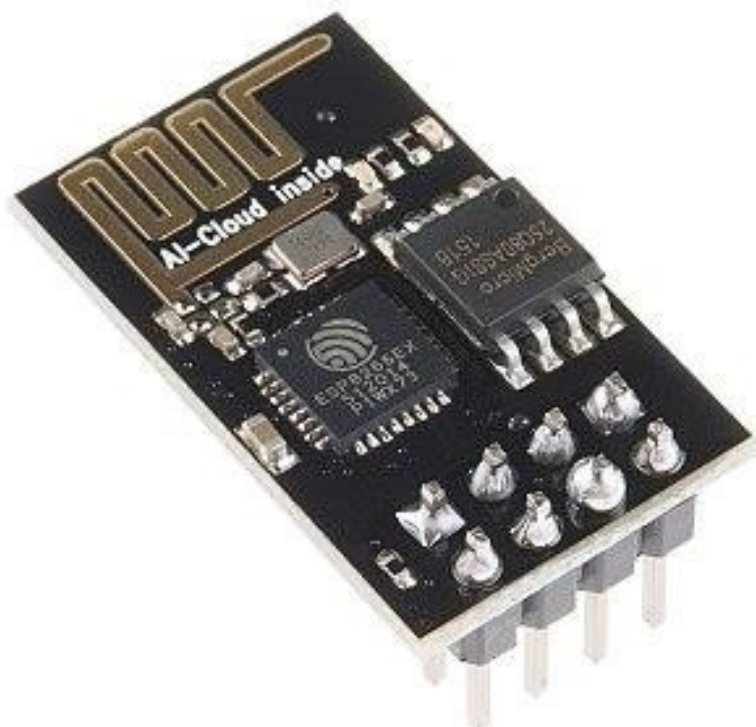


Рис. 8 – WiFi-модуль ESP8266

До внутрішніх належать всі модулі, розміщені на платі в стандартному виконанні Arduino, але завдяки відкритій ліцензії можна розробити свою плату під необхідні характеристики. Звичайно така плата не буде називатися Arduino (оскільки це – назва торгової марки).

1.2 Мікрокомп'ютер Raspberry Pi

1.2.1 Raspberry Pi

Мікрокомп'ютер – термін, що позначав комп'ютер, виконаний на основі мікропроцесора. На сьогодні існує безліч варіантів мікрокомп'ютерів, але ми розглядаємо Raspberry Pi.

Raspberry Pi – одноплатний комп'ютер, розміром з кредитну картку, спочатку розроблявся для опанування інформатики, але згодом набув широкого застосування в різних областях.

Архітектура плати Raspberry Pi 3b+ (рис. 9) схожа на Arduino, але має більш потужний 64-бітний процесор з частотою 1.4GHz, повноцінний модуль Wi-Fi і Lan, Bluetooth 4.2 / BLE, а також повноцінні два USB-

порти, вихід HDMI, а також аудіовихід. Тобто перед нами повноцінний мікрокомп'ютер з усіма необхідними для повноцінної роботи виходами.



Рис. 9 – Плата Raspberry Pi 3b +

Відмінною особливістю Raspberry Pi від інших мікрокомп'ютерів є наявність пінів для підключення зовнішніх модулів і датчиків, як на Arduino. Хоча функціонал мікрокомп'ютера ширший: до нього можна підключити дисплей або використовувати як сервер чи пристрій керування. На базі мікрокомп'ютера «будують» «розумні системи», такі як «SmartHome».

Креслення Raspberry Pi поширюються по відкритій ліцензії і будь-хто може зібрати свій мікрокомп'ютер з уже наявними кресленнями або купити модифіковану модель під власні потреби. На ринку достатньо моделей мікрокомп'ютерів, наприклад, з великою кількістю ОЗУ або розміщеною на платі флеш-пам'яттю, більш потужним процесором, порівняно з Raspberry Pi.

1.2.2 Операційна система і програмне забезпечення

Для мікрокомп'ютерів існує безліч варіантів операційних систем на базі Linux і одна операційна система, розроблена компанією Microsoft спеціально для мікрокомп'ютерів, – Windows 10 Internet of Things.

Отже, є два варіанти роботи з мікрокомп'ютером: з використанням безкоштовних рішень на базі Linux або платних від Microsoft. В рамках цієї роботи розглянемо систему на базі Linux.

На сайті Raspberry Pi офіційно підтримується два дистрибутиви Linux на базі Debian:

1. Noobs. Розрахована на користувачів-початківців. Має всі необхідні базові програми для початку програмування (в тому числі Arduino) і безліч встановлених програм для комфортної роботи з комп'ютером, в тому числі і браузер.

2. RASPBIAN – операційна система для досвідчених користувачів, створена і оптимізована спеціально для Raspberry Pi, має базовий набір програм і утиліт.

Сторонні розробники випускають свої дистрибутиви, що розрізняються включеними програмами і пакетами, а також збірками, що використовуються, наприклад:

- Ubuntu MATE for the Raspberry Pi 2 and Raspberry Pi 3. Має повністю налаштовану оболонку, встановлені драйвери і програми. Дистрибутив заснований на Ubuntu;
- OSMC (open source media center) – дистрибутив для побудови на базі Raspberry Pi домашнього мультимедійного центру;
- PiNet – централізований центр для роботи в класі (лабораторії), побудований так, щоб зберігати всю інформацію про користувачів і їх системі під час навчання програмування на мікроконтролерах.

Розглянуті операційні системи демонструють широку сферу застосування мікрокомп'ютерів і гнучкість їхнього налаштування під різні завдання.

2 ІНТЕРНЕТ РЕЧЕЙ. АРХІТЕКТУРИ І МЕТОДИ КОМУНІКАЦІЙ

2.1 Інтернет речей

Інтернет речей – це, згідно з визначенням Роба Ван Краненбурга, «концепція простору, в якому все з аналогового і цифрового світів може бути поєднане. Вона перевизначить наші стосунки з об'єктами, а також властивості і суть власне об'єктів». Іншими словами, це не просто безліч сенсорів, датчиків і приладів, об'єднаних в одну мережу. Головна особливість цієї мережі – це однозначна ідентифікація кожного об'єкта, а також більш тісна інтеграція реального і «віртуального» світів, де взаємодія відбувається між людьми і пристроями.

Розвиток «розумних речей», на думку Роба Ван Краненбурга, складається з чотирьох рівнів:

- 1) 1 рівень: ідентифікація кожного об'єкта в мережі;
- 2) 2 рівень: визначення та обслуговування потреб споживача;
- 3) 3 рівень: глобальна урбанізація та автоматизація систем міста;
- 4) 4 рівень: впровадження інтернету речей досягає планетарних масштабів, поява концепції «Сенсорна планета».

На сьогодні концепція розвитку інтернету речей зазнає множинних змін: рівнів стає більше, дроблення проблем, що розв'язуються, стає вузькоспеціалізованим. Це пов'язано з розвитком нових стандартів, більш чітким розумінням суті проблем, які потрібно розв'язати, а також з розвитком технологій в цілому, наприклад: появою нових технологій передачі енергії, розвиток речей, які носяться (wear electronics) і т. д. Все це розширює сфери застосування Інтернету речей в різних областях (рис. 10).

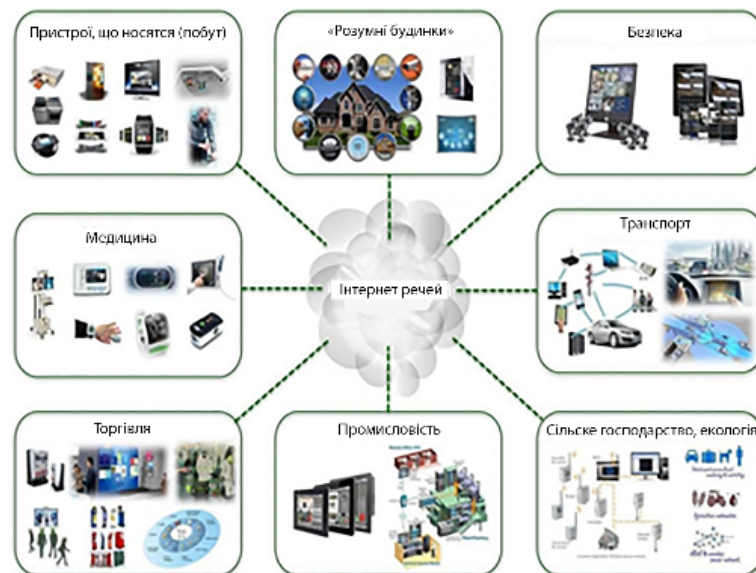


Рис. 10 – Приклад застосування Інтернету речей в різних сферах

2.2 Архітектура Інтернету речей

Стандартизація – це важливий момент будь-якої області. На сьогодні існує дуже багато видів архітектурних рішень, але типового (основного) не існує, оскільки перші спроби стандартизувати цю область були зроблені всього 4 роки тому, отже, зазначений процес ще не завершився.

Відсутність загальноприйнятих стандартів призвела до створення безлічі розрізнених систем, таких як «розумний будинок», від різних компаній, що призводить до ряду проблем:

- створення надлишкової кількості стандартів;
- надмірне регламентування найбільш простих об'єктів і процесів;
- велика кількість організацій і стандартів призводить до лобіювання інтересів окремих компаній в збиток загальних завдань стандартизації;
- довгі терміни розробки стандартів призводять до їхнього морального старіння, оскільки вони не встигають за розвитком технологій, особливо на ранньому етапі.

Отже, варто розглянути дві основні архітектури: програмну і фізичну. Умовно програмна архітектура складається з 7 рівнів:

1. Фізичний рівень – це датчики та електронні пристрої, які здатні підключатися до «речей» і отримувати дані від них.

2. Датчики збирають дані, але нам необхідно перетворити їх в зрозумілий формат і підключити цей пристрій до системи, використовуючи протокол обміну даними, який потрібно налаштувати.
 3. Підключення до мережі. Підключення пристрою до бездротової або провідної мережі. Ця можливість підключення змінюється на основі контексту і домену.
 4. Акумуляція даних. Можна сказати, що цей шар відповідає за рівень безпеки та доступ до даних. Цей шар повинен бути достатньо «мобільним» для внесення необхідних змін.
 5. Абстрагування даних. Зібрані дані використовуються для прийняття рішення або для цілей звітності. Це важливий шар, в який входить фактично створене рішення і бізнес-логіка.
 6. Рівень додатків. На цьому рівні відбувається контроль, аналіз і подання звітів системи. Грунтуючись на цих даних, ми можемо відображати звіти або застосовувати машинне навчання, якусь спеціальну логіку або використовувати інтелектуальне рішення і посилати сигнал назад на датчики.
 7. Останній рівень – це шар призначеного для користувача інтерфейсу.
- Це короткий опис організації програмної частини Інтернету речей. Програмна архітектура представлена на рис. 11.

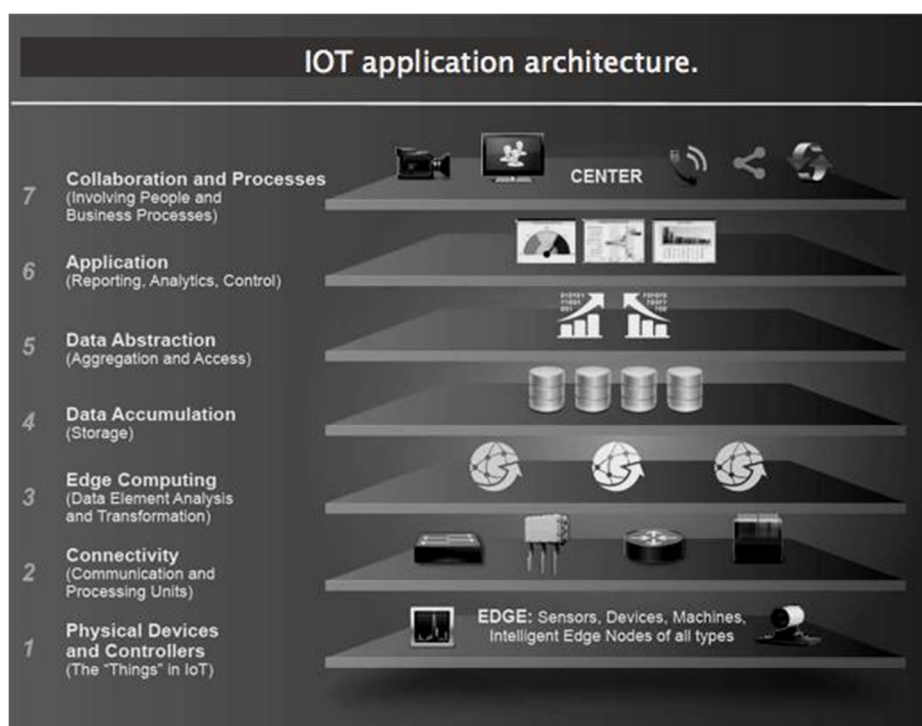


Рис. 11 – Приклад програмної архітектури Інтернету речей

Проста архітектура системи у фізичній реалізації являє собою взаємодію всіх учасників мережі (датчиків, пристроїв тощо), а також обробку отриманої інформації та прийняття рішень. Одна з таких реалізацій представлена на рис. 12. Залежно від завдань, що розв'язуються, і пристроїв, що використовуються, архітектура системи буде змінюватися. Під час її створення необхідно враховувати поточний розвиток стандартів і можливість подальшої підтримки системи і її розвитку.

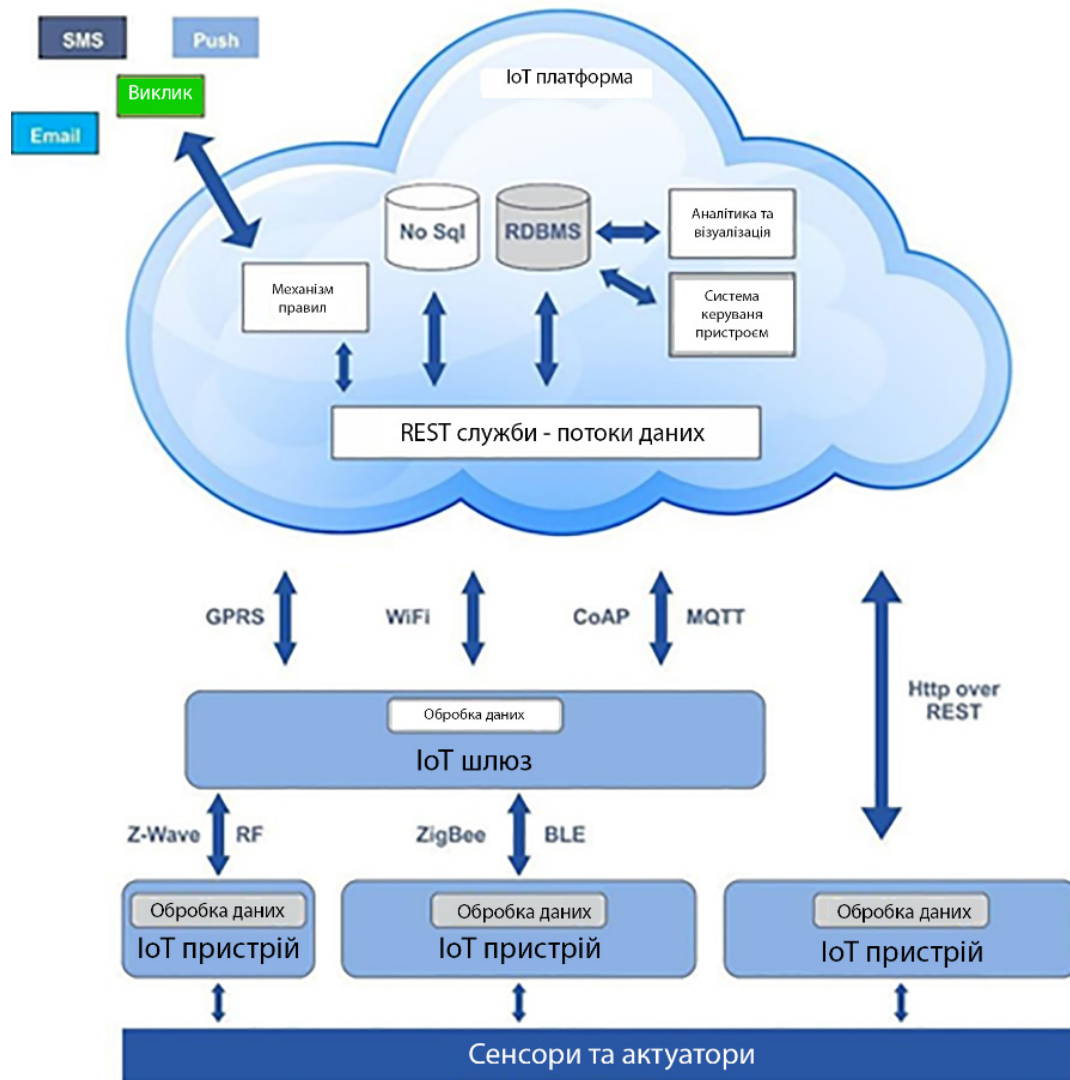


Рис. 12 – Приклад реалізації архітектури Інтернету речей

2.3 Технології комунікації між пристроями

Усі датчики і пристрої повинні бути пов'язані в єдину мережу для обміну інформацією між собою. Рішення, який тип зв'язку вибрати, ґрунтується на вирішенні конкретного завдання і оточення. Розглянемо основні технології комунікації між пристроями.

Wi-Fi – одна з найпоширеніших бездротових технологій передачі даних. Відповідає стандарту IEEE 802.11 (n / b / g) і функціонує на частотах 2,4 ГГц і 5 ГГц, залежно від використаного обладнання. Забезпечує високу швидкість передачі даних, але має ряд недоліків:

- високе енергоспоживання;
- в діапазоні 2,4 ГГц безліч пристроїв і пересічних технологій (Bluetooth);
- різні експлуатаційні обмеження для різних країн;
- слабка взломостійкість стандарту шифрування WEP: нові стандарти WPA і WPA2 мають більшу надійність.

Bluetooth дозволяє обмінюватися інформацією пристроїв, що знаходяться в радіусі 10 метрів (нові версії пристроїв працюють на відстані до 16 метрів). Має істотний недолік в безпеці, а також на якість зв'язку сильно впливають зовнішні фактори. До плюсів можна віднести низьке енергоспоживання і постійне поліпшення стандарту.

Технології стільникового зв'язку (2G, 3G, 4G, 5G). Ця технологія має ряд переваг, порівняно з розглянутими раніше. Велике покриття і досить швидка швидкість передачі даних (для мереж з технологією 3G / 4G) дозволяє охопити більшу територію при реалізації проектів без необхідності пошуку окремої точки доступу до мережі. Основною проблемою вважається обов'язкова прив'язка до стільникового оператора, можливе навантаження на вишки стільникового оператора (або їхня недоступність), високе енергоспоживання при передачі даних (особливо при слабкому покритті мережі).

LoRaWan – відкритий протокол для високоємних (до 1 млн пристроїв в одній мережі) мереж з великим радіусом дії і низьким енергоспоживанням. Протокол забезпечує двосторонній зв'язок з шифруванням для всіх класів пристроїв. Архітектура протоколу розроблялася в тому числі і для того, щоб легко знайти мобільні об'єкти для відстеження пересувань. Це найбільш швидко зростаючий напрям додатків Інтернету речей. Основним недоліком є швидкість передачі даних (до 5 КБ/с), але для передачі даних з датчиків цього цілком достатньо.

Local Area Network, LAN – комп’ютерна мережа, зазвичай покриває відносно невелику територію. Сполучення здійснюється за допомогою мережевого кабелю. Має високу швидкість передачі даних, високу стійкість перед перешкодами, певну адресацію всередині мережі. Із мінусів варто зауважити відсутність маршрутизаторів, фізичної доступності для підключення пристрою до мережі кабелем.

Наведені технології не є вичерпним описом всіх існуючих рішень, але є основними при роботі з інтернет-речами.

Лабораторна робота № 1

Тема: Підготовка робочого середовища для проведення лабораторних робіт. Інструментарій IoT-розробника.

Мета: ознайомлення та розгортання програмного забезпечення, необхідного для роботи з пристроями Інтернету речей.

Задачі:

- 1) встановити Arduino IDE та налаштувати на роботу з використанням ESP32;
- 2) встановити MQTT-broker Mosquitto;
- 3) встановити набір ПЗ для роботи з Node-red.

Теоретична частина

Arduino IDE. Середовище для програмування

Для початку будь-якої розробки необхідно обрати середовище для програмування. У нашому випадку необхідне середовище, яке дозволить нам програмування пристроїв IoT (Internet of Things) [1], є простим в освоєнні та має весь необхідний функціонал для початку роботи. Тому у цьому курсі буде використано Arduino IDE [2]. Для того щоб встановити Arduino IDE, необхідно завантажити інсталятор з офіційного сайту та запустити його. Має відкритись вікно ліцензійної згоди (рис. 13). Якщо Ви згодні з усіма пунктами, натисніть «I agree» та оберіть усі пункти з наступного меню (рис. 14). Після цього необхідно обрати директорію для установки (рис. 15). Після натискання на кнопку «Install» має розпочатися процес установки ПЗ. Установка має завершитися вікном на рис. 16.

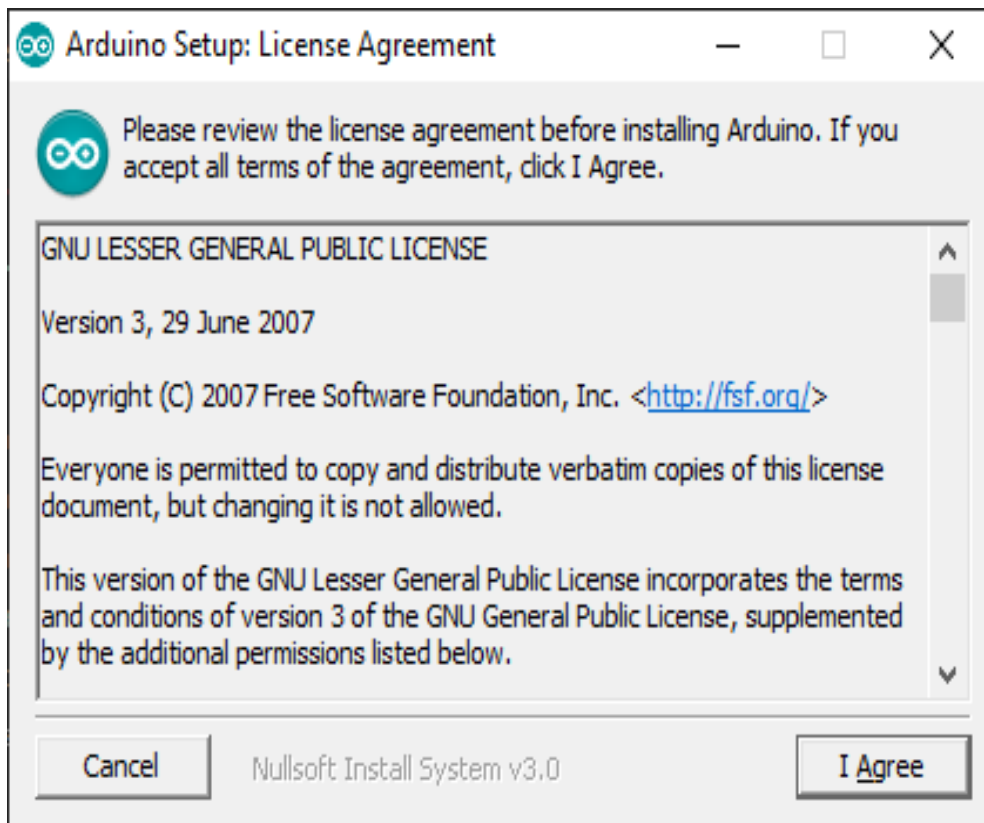


Рис. 13 – Ліцензійна згода

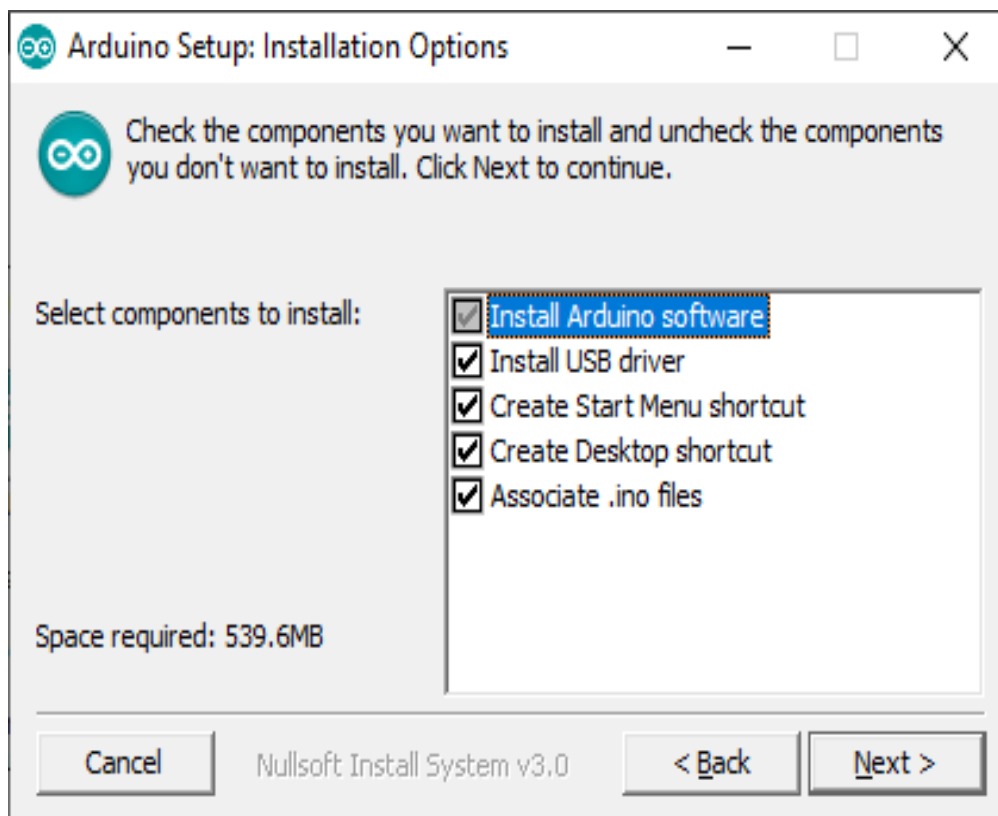


Рис. 14 – Деталі установки

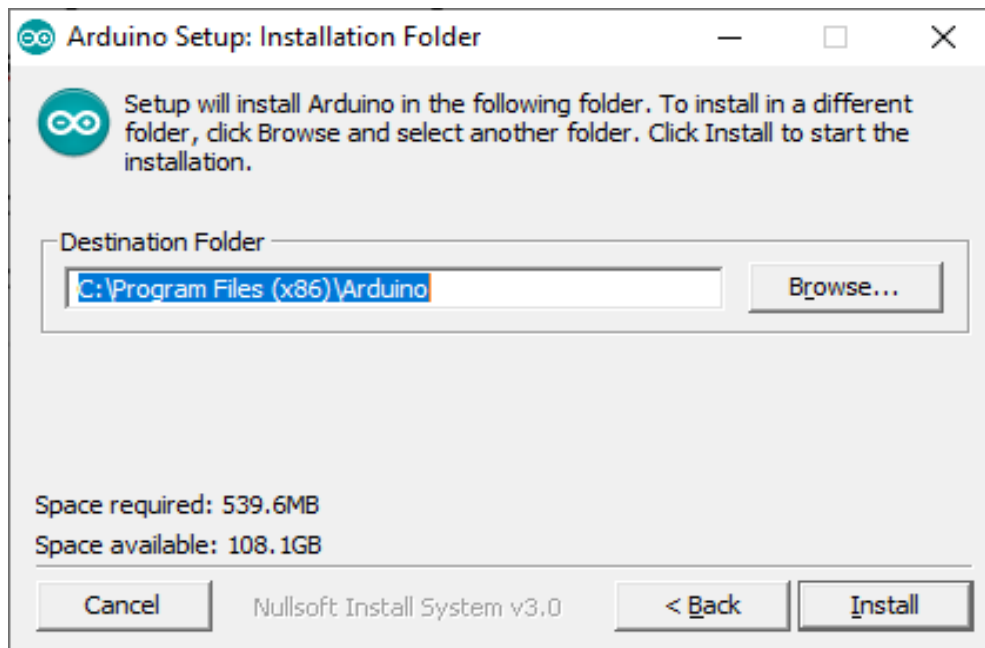


Рис. 15 – Діалог вибору директорії установки

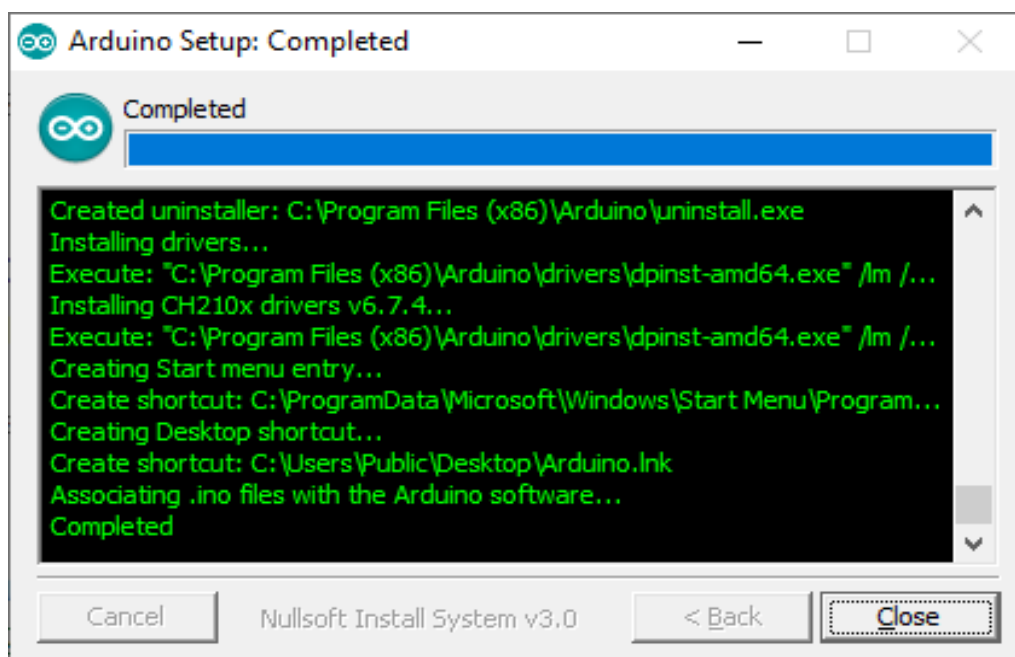


Рис. 16 – Успішна установка Arduino IDE

Після успішної установки Arduino IDE необхідно перейти до наступного кроку – підключення бібліотеки для роботи з ключовим для цього курсу модулем ESP32 [3]. Цей модуль IoT дуже популярний на сьогодні та підтримується такими сервісами, як Amazon AWS, Google IoT та Microsoft Azure. Для того щоб почати роботу з ним через Arduino IDE,

необхідно запуснути IDE та у меню «Файл» відкрити вікно «Налаштування» (рис. 17). Далі необхідно у полі «URL менеджерів додаткових плат» вказати посилання на такі пакети для менеджера плат: https://dl.espressif.com/dl/package_esp32_index.json, https://arduino.esp8266.com/stable/package_esp8266com_index.json, або ввести їх у вікні рис. 18.

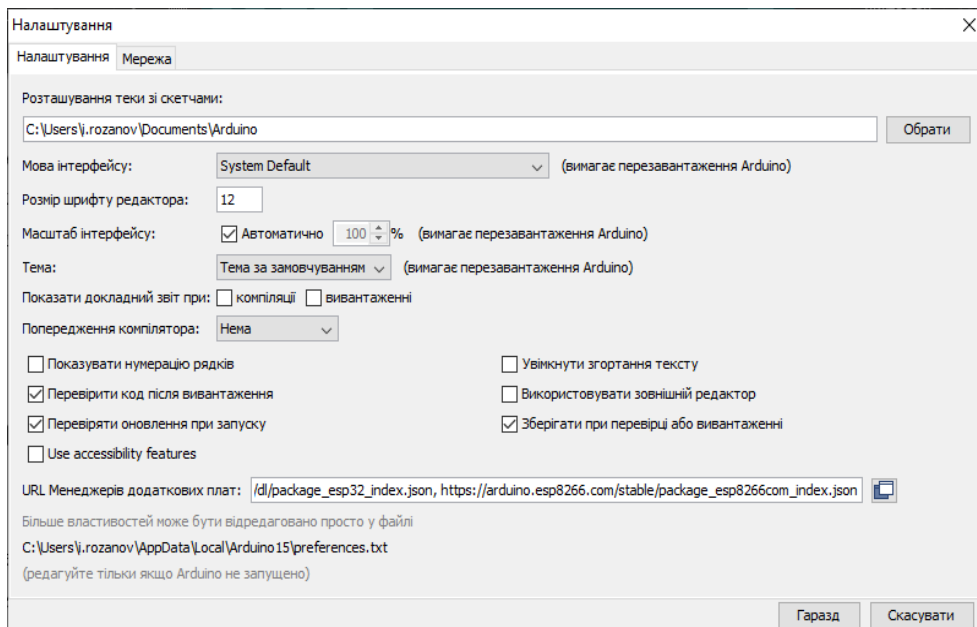


Рис. 17 – Вікно налаштувань Arduino IDE

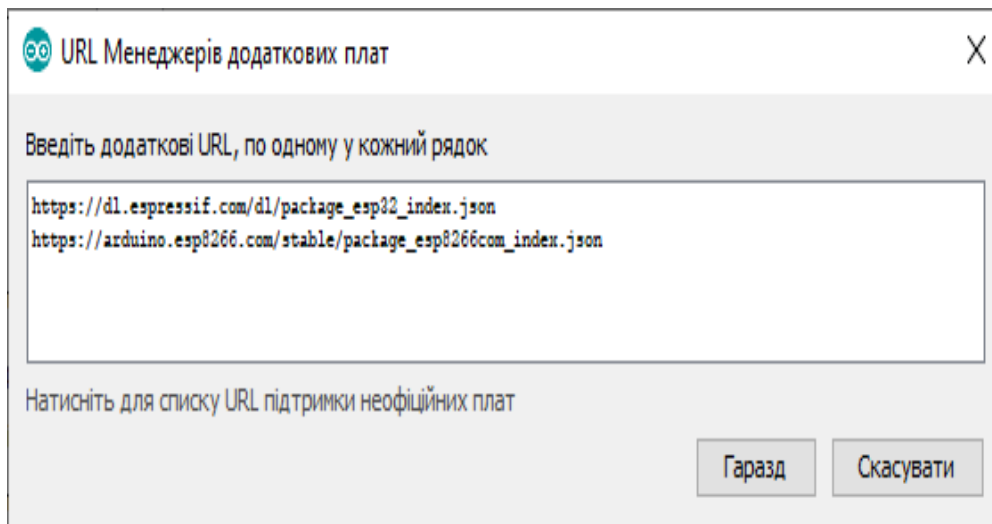


Рис. 18 – Список пакетів для менеджера плат

Після цього необхідно у меню «Інструменти» перейти у менеджер плат (рис. 19) та ввести у рядок пошуку «плат esp32» і встановити набір плат у середовище (рис. 20).

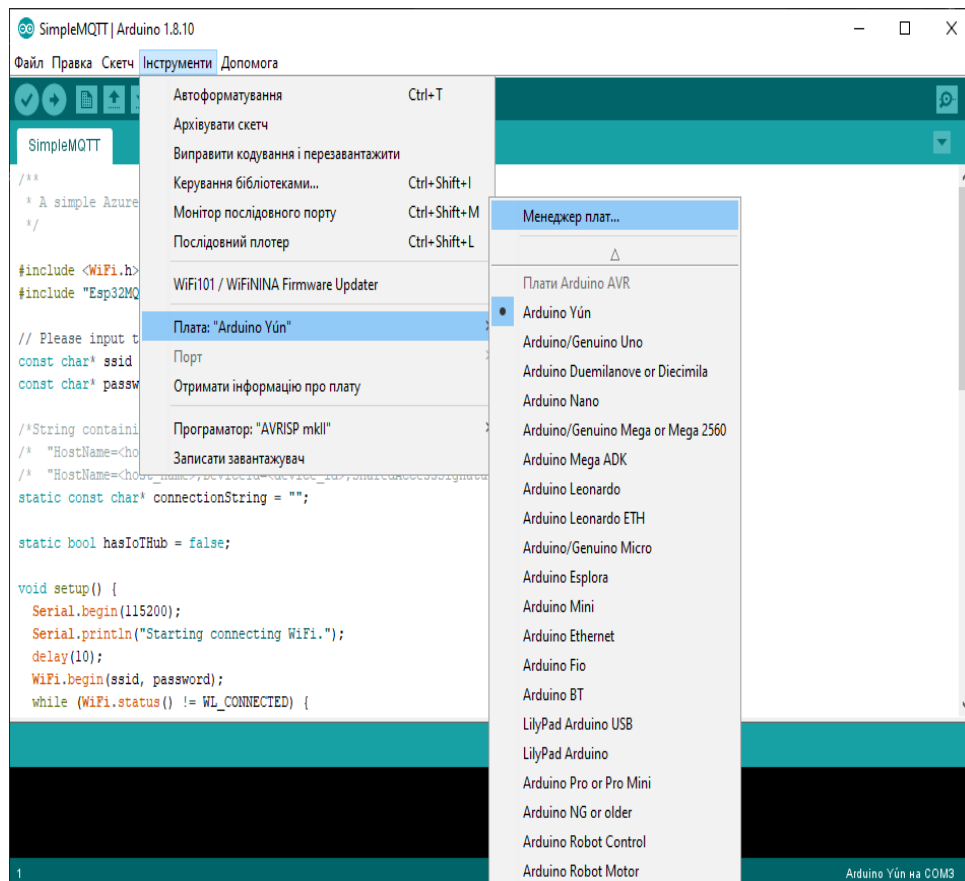


Рис. 19 – Меню менеджера плат

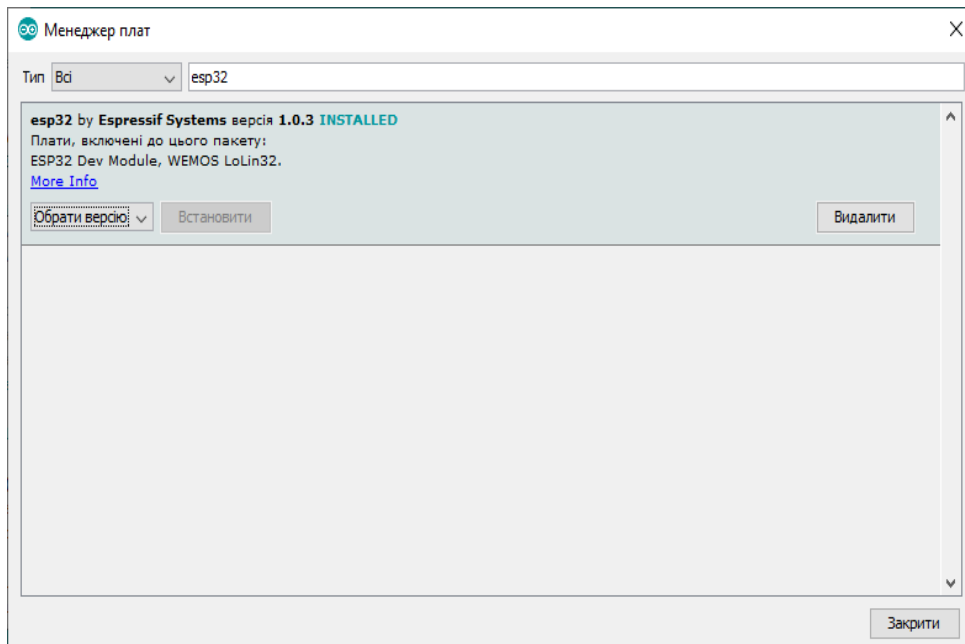


Рис. 20 – Менеджер плат

Після цього у меню плат стануть доступні плати розробника з лінійки модулів ESP32. Нас цікавить DOIT ESP32 DEVKIT V1, тому що саме він наявний у лабораторії. Але на цьому етапі процес підготовки програмного середовища ще не завершено, тому що варто приділити увагу також

процесу установки бібліотек. Для цього необхідно перейти в меню «Скетч» та обрати меню «Додати бібліотеку» (рис. 21). У подальшому нам знадобиться бібліотека для роботи з датчиком DHT22 [4], тому розглянемо процес установки бібліотеки на цьому прикладі. Для цього необхідно відкрити вікно «керування бібліотеками» та у пошуку ввести запит «DHT22 ESP32», потім потрібно обрати необхідну бібліотеку (рис. 22).

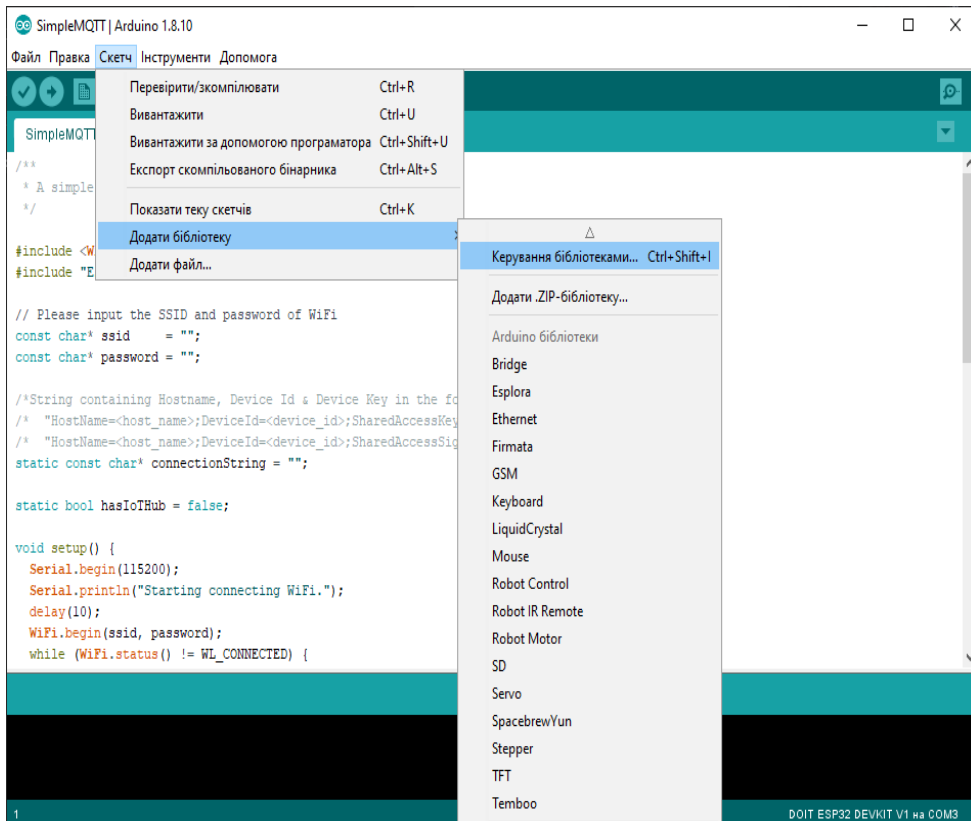


Рис. 21 – Менеджер бібліотек



Рис. 22 – Вибір бібліотеки у менеджері бібліотек

Встановлення MQTT-брокера Mosquitto

Окрім саме «речей» з інтернету речей, важливим елементом концепції IoT є шляхи зв'язку між «речами». Найпоширенішим протоколом зв'язку в IoT є MQTT [5]. Взагалі в IoT використовується безліч протоколів на базі TCP/IP, але у цьому курсі буде використано саме MQTT, тому що він є де-факто стандартом галузі, бо був розроблений саме для використання в Інтернеті речей. Ключовими поняттями для MQTT є брокер та клієнт. Брокер – це програмне забезпечення, що забезпечує транзит повідомлень між різними клієнтами, які підключені до сервера, на якому розгорнуто MQTT-брокер. Детальніше про роботу протоколу йтиметься у наступній лабораторній роботі.

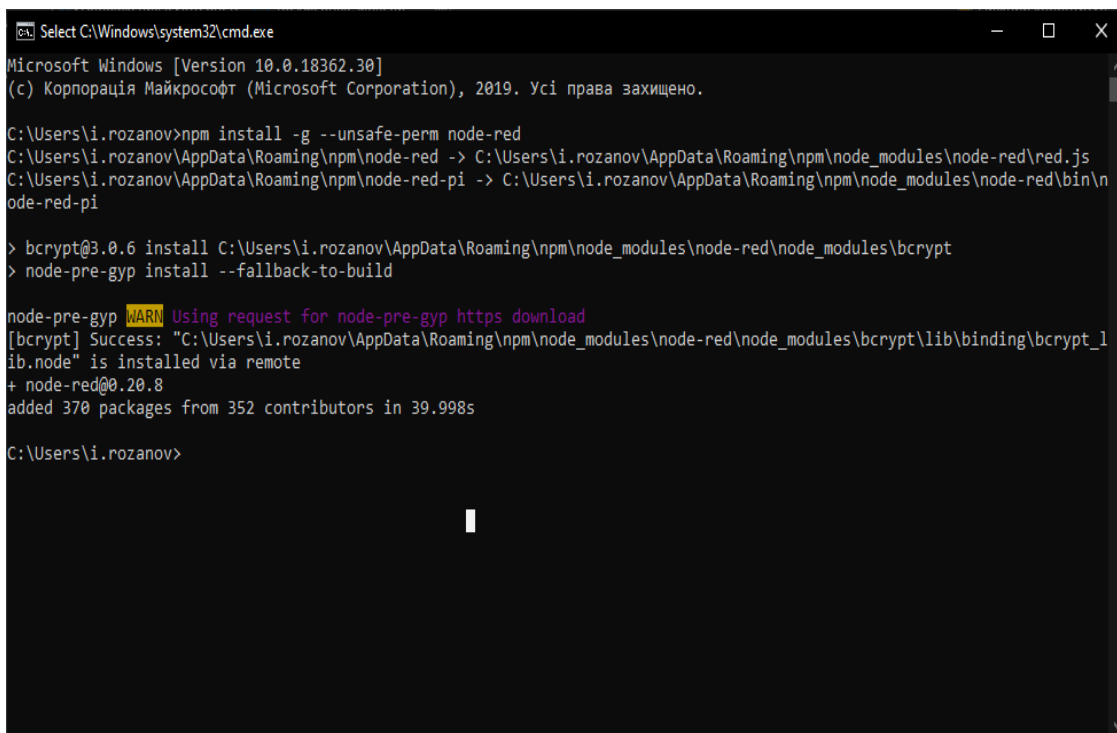
Загальною рекомендацією для встановлення та використання консольних додатків, які надані у цій лабораторній роботі, є встановлення модуля Ubuntu 18.04 на Windows 10.

Як програмне забезпечення, що буде виконувати функції брокера для проведення лабораторних робіт, буде використовуватися Mosquitto [6]. Для того щоб встановити цей MQTT-брокер, необхідно перейти на офіційний сайт та завантажити інсталятор для Вашої операційної системи. Після завершення установки, за умови, що mosquito було встановлено як службу Windows, у випадку установки на цю операційну систему запускати брокер непотрібно, він буде автоматично запущений під час запуску операційної системи. В інших випадках необхідно виконати команду `mosquitto` через термінал командного рядка. Також, окрім брокера, є можливість використовувати два типи клієнтів від `mosquitto` – `mosquitto_pub` для публікування повідомлень та `mosquitto_sub` для підписки на отримання повідомлень. Це може бути доречно для перевірки правильності роботи пристроїв Інтернету речей та взаємодії з ними за допомогою ПК.

Встановлення Node-red та необхідного для його роботи ПЗ

У попередніх розділах було розглянуто програмне забезпечення, необхідне для роботи безпосередньо з пристроями IoT та для забезпечення їхньої взаємодії між собою та з іншими сервісами. Сервісів, які можуть бути споживачами інформації від пристроїв IoT, також існує

безліч, як і апаратних рішень серед пристроїв та програмних рішень серед засобів комунікації. Сервісом, який є достатньо гнучким, щоб дозволити розробку інтерфейсу користувача певної системи Інтернету речей, а також забезпечення можливості програмної обробки даних та їхньої передачі у базу даних або іншим споживачам інформації за допомогою інших протоколів, є Node-Red [7]. Це засіб для програмування рішень для екосистеми інтернету речей на базі програмного оточення Node.js [8], особливістю якого є використання потоків даних та функціональних вузлів на графічному інтерфейсі для спрощення розробки багатопоточних додатків. Установка складається з двох пунктів – встановлення Node.js та Node-red поверх нього. Для того щоб встановити Node.js, необхідно завантажити інсталятор з офіційного сайту та виконати інструкції. Після успішної установки необхідно відкрити командний рядок та виконати команду «`npm install-g--unsafe-perm node-red`» (або «`sudo npm install-g--unsafe-perm node-red`» у випадку linux-систем (рис. 23). Для того щоб почати роботу у node-red, необхідно відкрити командний рядок та виконати команду Node-red. У випадку з Windows можливо знадобиться перейти до папки зі встановленим пакетом. Середовище розробника буде доступне у браузері за адресою localhost:1880 (рис. 24).



```
Microsoft Windows [Version 10.0.18362.30]
(c) Корпорація Майкрософт (Microsoft Corporation), 2019. Усі права захищено.

C:\Users\i.rozanov>npm install -g --unsafe-perm node-red
C:\Users\i.rozanov\AppData\Roaming\npm\node-red -> C:\Users\i.rozanov\AppData\Roaming\npm\node_modules\node-red\red.js
C:\Users\i.rozanov\AppData\Roaming\npm\node-red-pi -> C:\Users\i.rozanov\AppData\Roaming\npm\node_modules\node-red\bin\node-red-pi

> bcrypt@3.0.6 install C:\Users\i.rozanov\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

node-pre-gyp WARN Using request for node-pre-gyp https download
[bcrypt] Success: "C:\Users\i.rozanov\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt\lib\binding\bcrypt_1
ib.node" is installed via remote
+ node-red@0.20.8
added 370 packages from 352 contributors in 39.998s

C:\Users\i.rozanov>
```

Рис. 23 – Установка Node-red

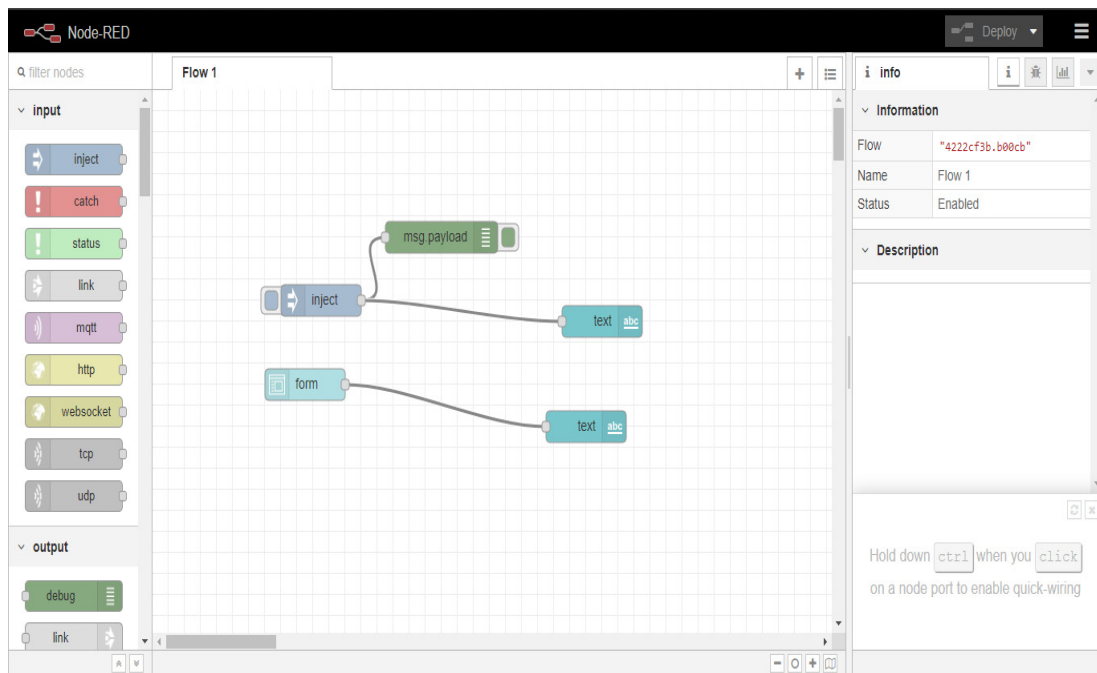


Рис. 24 – Інтерфейс Node-red у браузері

Практична частина

Виконання роботи

1. Скачати наведені у теоретичному матеріалі ПЗ з поданих посилань (офіційних сайтів та репозитаріїв).
2. Встановити Arduino IDE.
3. Запустити Arduino IDE та записати свою групу, ПІБ та час і дату виконання як коментар до коду.
4. Зробити скріншот вікна програми.
5. Встановити Mosquitto (та `mosquito_client` у випадку linux та mac os).
6. Запустити брокер через термінал та зробити скріншот його роботи.
7. Встановити Node.js.
8. Встановити Node-red через термінал (командою `npm install node-red`).
9. Запустити Node-red та зробити скріншот з терміналу, та з браузера. У браузері необхідно розмістити код з коментарем, у якому необхідно вказати групу, ПІБ та дату.

Звіт має містити: титульну сторінку, скріншоти з ходу роботи та відповіді на контрольні питання.

Контрольні питання

1. Опишіть процес завантаження коду в ESP32 після написання у Arduino IDE.
2. Наведіть приклади інших (окрім Arduino IDE) середовищ розробки та мов, якими можна користуватися для програмування ESP32 та операційних систем, що можна встановити на модуль.
3. Розкажіть про те, як працює Mosquitto.
4. Наведіть приклади протоколів, крізь які ESP32 та пристрої Інтернету речей можуть передавати дані через Інтернет.
5. Чи можна розгорнути MQTT-брокер на мікроконтролері? На яких мікроконтролерах це можна зробити?
6. Назвіть ключові плюси та мінуси сервісу Node-red.
7. Які ще сервіси, окрім Node-red, можна використовувати для взаємодії між пристроями Інтернету речей та іншими сервісами в Інтернеті?

Список використаних посилань

1. <https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot>
2. <https://www.arduino.cc/en/main/software>
3. <https://www.espressif.com/en/products/hardware/esp32/overview>
4. <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>
5. <http://mqtt.org/>
6. <https://mosquitto.org/>
7. <https://nodered.org/>
8. <https://nodejs.org/uk/>

Лабораторна робота № 2

Тема: Входи / виходи загального призначення. Вхідні та вихідні цифрові сигнали. Взаємодія між двома модулями ESP32 за допомогою MQTT.

Мета: ознайомитися із взаємодією двох пристроїв Інтернету речей на прикладі керування світлодіодами на одному модулі за допомогою кнопок на іншому.

Задачі:

- 1) ознайомитися з функціонуванням входів / виходів загального призначення за допомогою світлодіодів та кнопок;
- 2) реалізувати підключення ESP32 до Wi-fi та MQTT-брокера;
- 3) дослідити особливості роботи протоколу MQTT та взаємодії між різними пристроями.

Теоретична частина

1. Ознайомлення з функціонуванням входів / виходів загального призначення за допомогою світлодіодів та кнопок.

ESP32 [1] має 36 вбудованих входів / виходів загального призначення (GPIO – general purpose input / output). З їхнім розташуванням на наявному у лабораторії модулі можна ознайомитися на рис. 25. Всі вони мають можливість бути налаштованими як для читання цифрового сигналу, так і для його генерації. Деякі з них мають додаткові функції, наприклад підключені до вбудованого АЦП [2], або до апаратних інтерфейсів передачі даних (I²C [3], UART [4], SPI [5]), чи мають можливість використовуватися як сенсорні елементи. Також наявні службові піни для виводу живлення та заземлення. Але певні GPIO мають деякі обмеження у використанні. Отже, GPIO з 34 по 39 можуть працювати лише на вхід, а використання GPIO з 6 по 11 і 0, 1 та 3 рекомендується лише за їхнім призначенням, яке прописане у документації, тому що вони підключені безпосередньо до пам'яті модуля і можуть викликати нестабільність його роботи. Робота з цифровими входами / виходами найпростіша, тому з неї і починається ознайомлення з функціоналом модуля. Враховуючи, що програмування буде здійснюватися за допомогою Arduino IDE, для реалізації програми, яка буде вмикати / вимикати світлодіод [6], необхідно підключити його за поданою на рис. 26 схемою та написати наведений на рис. 27 код програми.

Світлодіод необхідно підключати через струмообмежувальний резистор у 100 Ом. Програма наявна у прикладах у самому середовищі, лише потрібно змінити необхідний GPIO на той, що використовується на схемі. Також варто зазначити, що програми, які пишуться в Arduino IDE, називаються скетчі, і цей термін буде використовуватися далі.

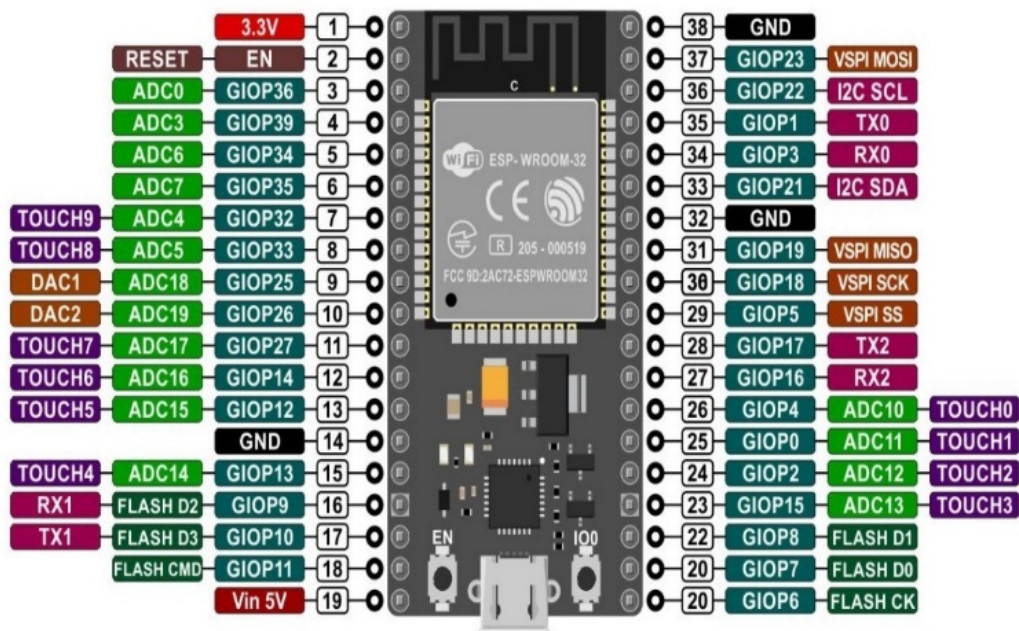


Рис. 25 – Розташування та призначення входів / виходів ESP32

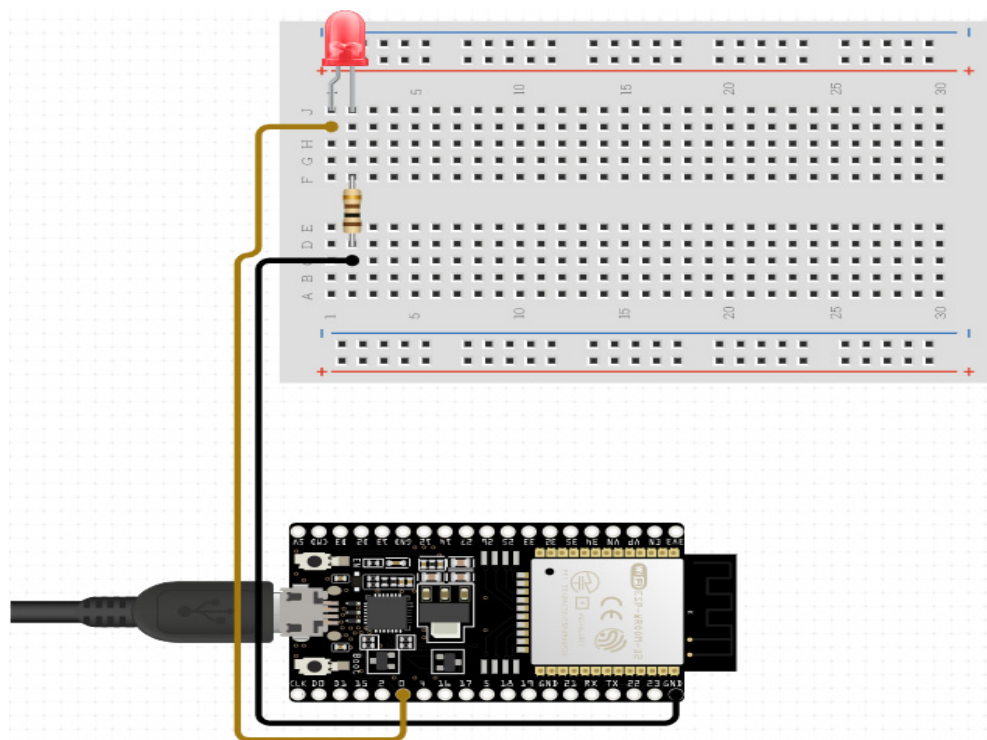


Рис. 26 – Підключення світлодіоду до ESP32

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 0 as an output.
  pinMode(0, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(0, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(0, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

Рис. 27 – Код програми Blink

На цьому скетчі відбувається таке:

- 1) у функції `setup()` ініціалізується GPIO, до якого під'єднано світлодіод;
- 2) у функції `loop()` (яка працює циклічно впродовж усього часу, доки пристрій ввімкнений), спочатку посилається сигнал високого значення, тобто подається напруга 3,3 В, а після затримки в 1 секунду посилається сигнал низького рівня, тобто 0 В, також на 1 секунду.

Під час виконання роботи цього скетча світлодіод буде періодично вмикатися та вимикатися із заданою затримкою, тому що на GPIO, до якого він під'єднаний, подаються керуючі сигнали, струму у яких достатньо, щоб запалити світлодіод.

Наступним кроком у вивченні роботи GPIO буде читання цифрового сигналу. Для цього буде використано тактову кнопку. Для її підключення необхідно скористатися схемою на рис. 28. Підключення кнопки необхідно здійснювати через струмообмежувальний резистор 10 кОм, аби уникнути пошкодження GPIO високим струмом. Для того щоб відстежити стан натискання кнопки, буде використано повідомлення через послідовний порт.

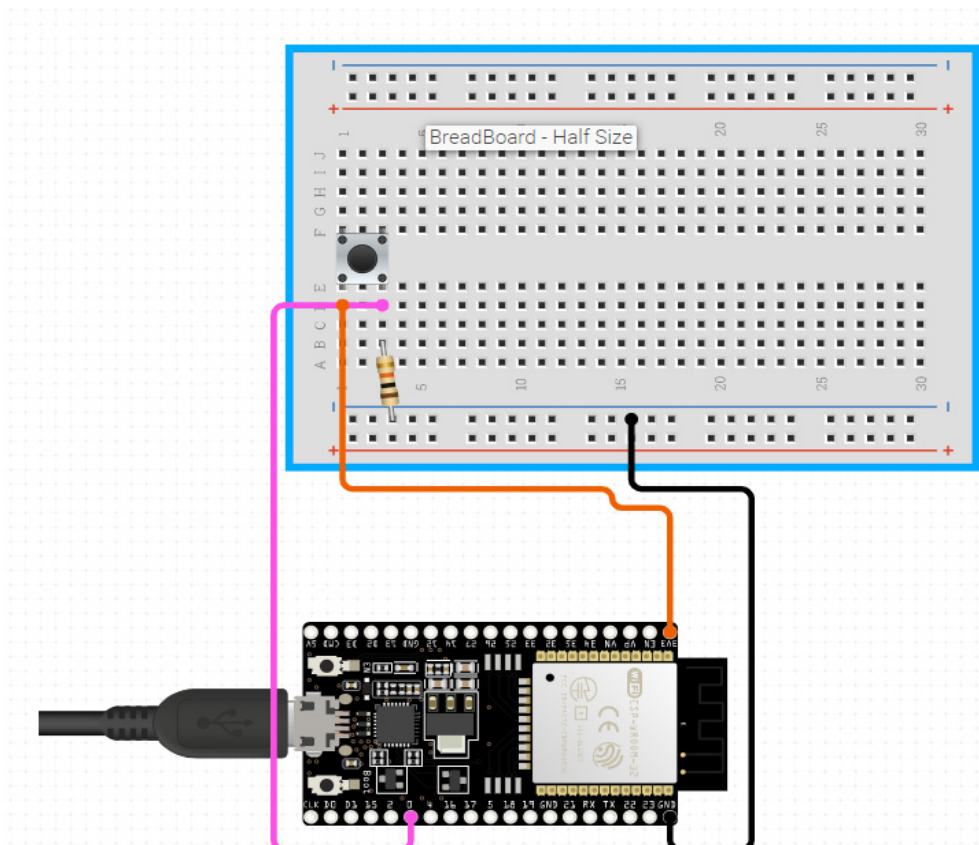


Рис. 28 – Підключення тактової кнопки до ESP32

Послідовний порт є основним засобом для налаштування та випробування будь якого скетча, також він доволі поширений серед готових рішень на базі мікроконтролерів ATmega та пристроїв IoT. Для того щоб мати можливість взаємодіяти з пристроєм через ПК, необхідно використовувати методи класу Serial [7]. Розглянемо приклад коду, який наведено на рис. 30. Для зручності ініціалізується змінна `pushButton`, якій має бути присвоєно номер GPIO, до якого під'єднано кнопку. Далі відбувається ініціалізація послідовного порту та виводу, до якого під'єднано кнопку. Вивід конфігурується, як і INPUT, тобто вхід. Після цього програма здійснює циклічне читання стану кнопки за допомогою функції `digitalRead` та виводить стан у послідовний порт із заданим інтервалом. Повідомлення будуть виводитися тільки за умови, що на вхід було подано логічну одиницю, тобто високий рівень сигналу (3,3 В). Це можливе тільки при натисненні на кнопку.


```

// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);          // delay in between reads for stability
}

```

Рис. 30 – Скетч для зчитування стану кнопки та виведення у послідовний порт

Реалізація підключення ESP32 до Wi-fi та MQTT-брокера

Після ознайомлення з базовими функціями ESP32 необхідно перейти до функціоналу, який є основним за призначенням – підключення та робота з Wi-fi-мережами. ESP32 може працювати у трьох режимах: як клієнт, який під’єднується до існуючої мережі, як точка доступу до Wi-Fi та у гібридному режимі. Нас буде цікавити перший варіант, тому що простіше за все реалізувати взаємодію між декількома пристроями саме через єдину точку доступу та сервер. Для підключення до мережі Wi-Fi необхідно скористатися бібліотекою Wi-fi [8]. У загальному випадку, для реалізації підключення до звичайної «домашньої» точки доступу, можна скористатися прикладом «WifiClientBasic», який наявний у прикладах до бібліотеки. Код можна подивитися за посиланням [9]. У цьому скетчі відбувається підключення до мережі Wi-fi, яку необхідно вказати у методі WiFiMulti.addAP(«SSID», «passpasspass»), замість ssid та passpasspass. Далі код виконує підключення до локального сервера та обмінюється з ним пакетами. У випадку з бездротовою мережею у нашій лабораторії буде необхідно скористатися іншим прикладом підключення, через те що вона є корпоративною і має певні умови для підключення, а саме: необхідність авторизації через логін / пароль від корпоративного облікового запису, тому що використовується протокол захисту EAP.

Приклад коду наявний за посиланням [10]. У цьому прикладі виконується підключення за заданими параметрами EAP_IDENTITY «login»; EAP_PASSWORD «password»; const char* ssid = «eduroam»; const char* host = «arduino.php5.sk»; останній параметр використовується лише для того, щоб перевірити підключення до віддаленого сервера. Необхідний мінімум коду для підключення до корпоративної мережі Wi-fi наведено у додатку 1. Цей тип підключення потрібен для доступу в Інтернет, але для роботи у локальній мережі, з MQTT-брокером, доцільніше скористатися скетчем, який наведено у додатку 2.

Якщо підключення до Wi-fi відбулося вдало, то необхідно перейти до наступного кроку – розглянути взаємодію модуля та MQTT-брокера. Розглянемо ключові поняття цього протоколу [11].

Для того щоб програмно реалізувати роботу з MQTT, нам необхідна бібліотека, яка спростить цей процес. У цьому курсі буде використовуватися бібліотека PubSubClient [12]. Приклад використання цієї бібліотеки наведено у додатку 2.

Взаємодія між пристроями у MQTT-мережі

Процес «спілкування» між клієнтами (а клієнтом може бути і пристрій IoT, і програмне забезпечення на телефоні, ПК або сервері) полягає у тому, що є MQTT-брокер, який є вузлом, до якого надходять повідомлення. Повідомлення надходять у певні топіки, назва яких встановлюється на розсуд розробника, або згідно з певними вимогами, які можуть бути у проекті. Клієнти можуть публікувати повідомлення у певний топик на брокері, а можуть бути підписані на певний топик та отримувати повідомлення, які надходять у цей топик. Отже, можлива доставка одного і того самого повідомлення майже необмеженій кількості клієнтів. У випадку цієї лабораторної роботи буде необхідно налагодити обмін повідомленнями між двома модулями ESP32. Обидва модулі будуть клієнтами відносно брокера, але один з них має виконувати функції пристрою керування, а інший – виконувати певні команди. Для цього, перш за все, необхідно домовитися, що у системі буде умовний топик для повідомлень, які будуть передавати команду, та топик, у який будуть надходити статуси пристроїв. Отже, можна за допомогою взаємодії людини або середовища з пристроєм активувати дію іншого пристрою або програми на сервері, який прослуховує певний

топiк. Наприклад, запис у базу даних стану середовища за показниками метеостанції або надсилання е-мейлу натисненням певної кнопки на бездротовому пристрої. Можливостей для використання протоколу безлiч, у практичній частині роботи буде наведено найпростіший.

Практична частина

Завдання 1. Керування світлодіодом за допомогою тактових кнопок за допомогою модуля ESP32.

Виконання роботи

1. Підключити схему на лабораторному стенді (рис. 30), як наведено схемі (рис. 31). Світлодіоди, кнопки та резистори вже змонтовано на макетній платі, яка не потребує пайки, їх лише потрібно під'єднати до GPIO за варіантами завдання.

2. Під'єднати модуль до ПК через USB-кабель та обрати у Arduino IDE параметри для роботи с платою, як на рис. 32.

3. Реалізувати програму, яка буде циклічно зчитувати стан усіх трьох кнопок, та виводити повідомлення з розташуванням кнопки у послідовний порт при натисканні на неї. Світлодіоди та кнопки позначені як «ліва», «середня» та «права» (1, 2, та 3 відповідно).

Також під час натисканні на певну кнопку має загорятися певний світлодіод. Він має горіти за алгоритмом, згідно з варіантом завдання.

Варіанти до завдання 1

Варіант	Виводи для підключення периферії	Алгоритм роботи світлодіода
1	36, 39, 34, 25, 32, 33	При натисканні на кнопку, яка збігається зі світлодіодом, світлодіод має ввімкнутися і горіти, поки кнопка натиснута.
2	35, 32, 33, 25, 26, 27	Світлодіод має загорятися при натисканні на кнопку та горіти, поки кнопка не буде натиснута ще раз.
3	25, 26, 27, 14, 12, 13	Світлодіод має горіти, поки кнопка натиснута, та горіти ще 10 секунд після того, як її відпустили.
4	14, 12, 13, 19, 18, 5	Поки кнопка натиснута, світлодіод має вмикатися та вимикатися з інтервалом у секунду.

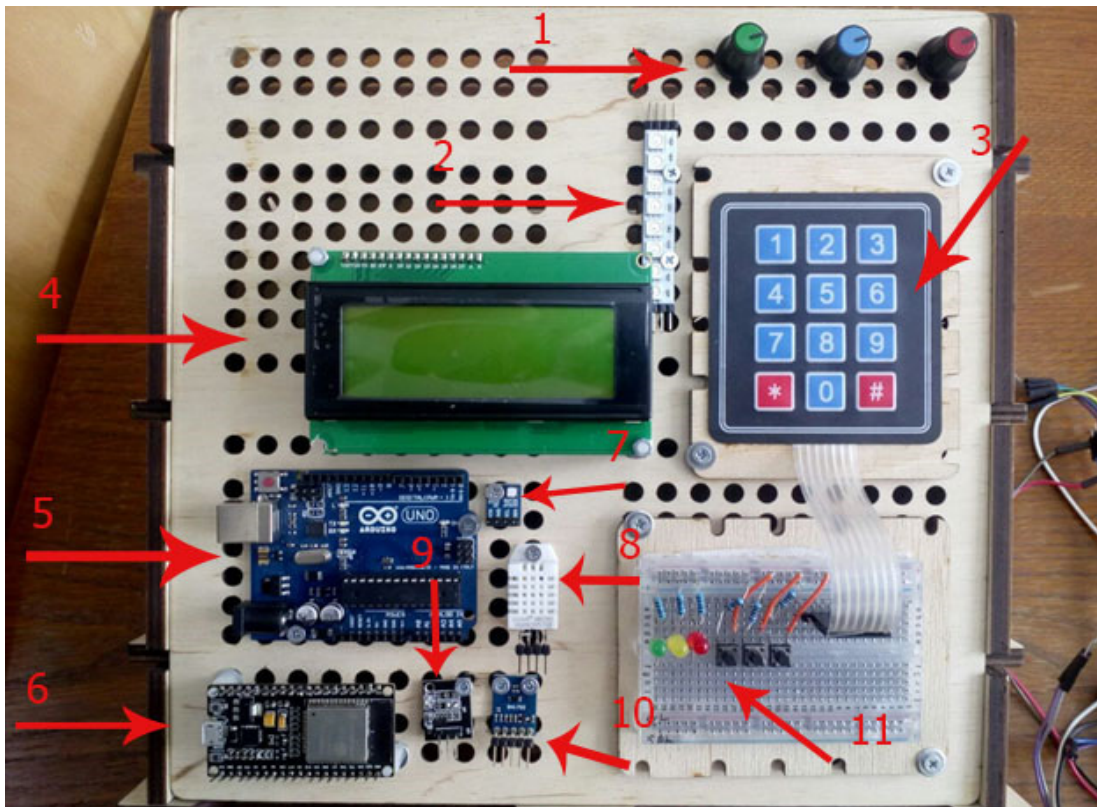


Рис. 30 – Лабораторний стенд Інтернету речей: 1) потенціометри; 2) адресна RGB стрічка; 3) матрична клавіатура; 4) LCD дисплей з I2C конвертором; 5) Arduino Uno; 6) ESP32 devboard; 7) датчик температури і вологості HTU21; 8) датчик температури і вологості DHT22; 9) датчик Холла; 10) датчик освітленості BH1750; 11) макетна плата та змонтована схема зі світлодіодами та кнопками із необхідною обв'язкою

Завдання 2, частина 1. Виведення статусу кнопок у певний топик MQTT-брокера.

За наявності виконаного завдання 1 та зразків коду з додатків стає можливим виконання першої частини завдання 2. Воно полягає у тому, що тепер дані про стан кнопок необхідно виводити не тільки у послідовний порт, але і передавати через Wi-fi іншим пристроям. Для виконання цього завдання необхідно отримати параметри мережі, до якої необхідно підключити пристрій, та запустити MQTT-брокер mosquitto на локальному ПК або отримати адресу сервера під час виконання роботи. Важливо, що модуль ESP32 та брокер для виконання лабораторної роботи мають знаходитися в одній мережі Wi-fi.

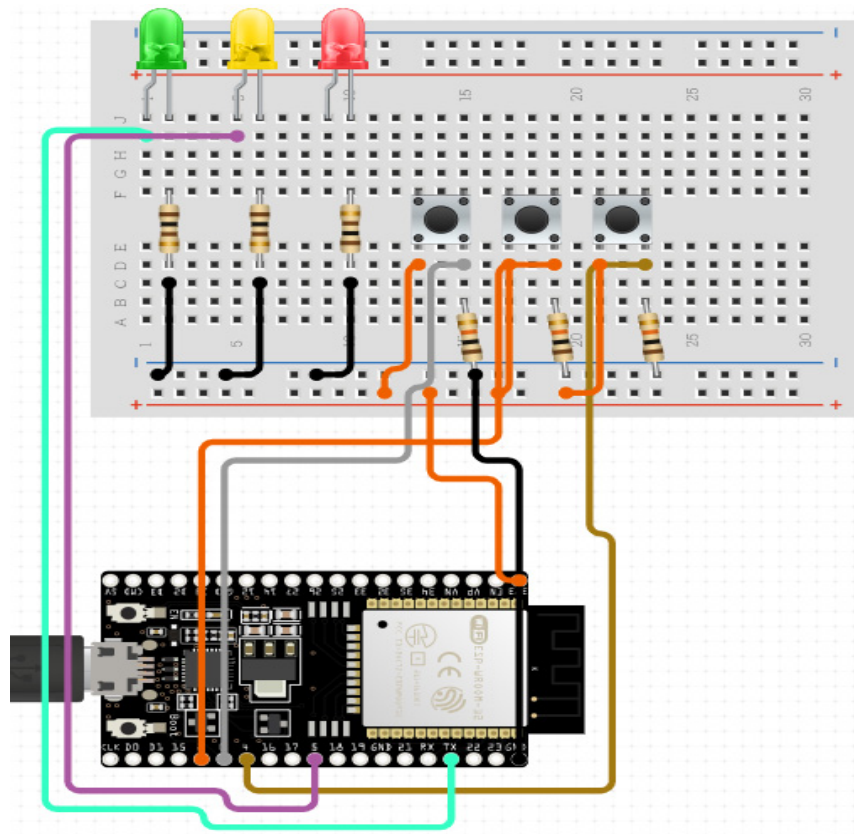


Рис. 31 – Схема підключення трьох кнопок та трьох світлодіодів до ESP32

Для виконання цього завдання також потрібно виконати такі кроки:

- 1) мати зібрану з попереднього завдання схему підключення (згідно з варіантом);
- 2) отримати дані для підключення до мережі Wi-fi у лабораторії;
- 3) розгорнути або отримати адресу вже розгорнутого MQTT-брокера. Якщо брокер розгорнуто на локальному ПК, то його адресу можна дізнатися командою `ipconfig` у командному рядку;
- 4) далі необхідно скористатися додатком 2 для реалізації підключення модуля до Wi-fi та брокера, а також відредагувати код у такий спосіб, щоб контролер стежив за станом кнопок та відправляв їхній стан у послідовний порт та у топик «`button_status_%номер варіанта%_%номер кнопки%`» на вказану адресу брокера. Перевірка стану кнопок має бути у функції `loop()`;
- 5) для перевірки роботи цього пристрою необхідно скористатися додатком `mosquitto_sub` [13]. Для цього необхідно перейти у папку, де встановлено брокер (у випадку windows), та запустити звідти командний рядок, з якого

необхідно викликати додаток `mosquito_sub` з параметрами `-h%адреса брокера у Вашій мережі% -p 1883 -t button_status_%номер варіанта%_%номер кнопки%` (рис. 33). У випадку, якщо брокер розгорнуто на Вашому локальному ПК, вказувати параметри `-h` та `-p` необов'язково, тому що доцільно скористатися параметрами за замовчуванням.

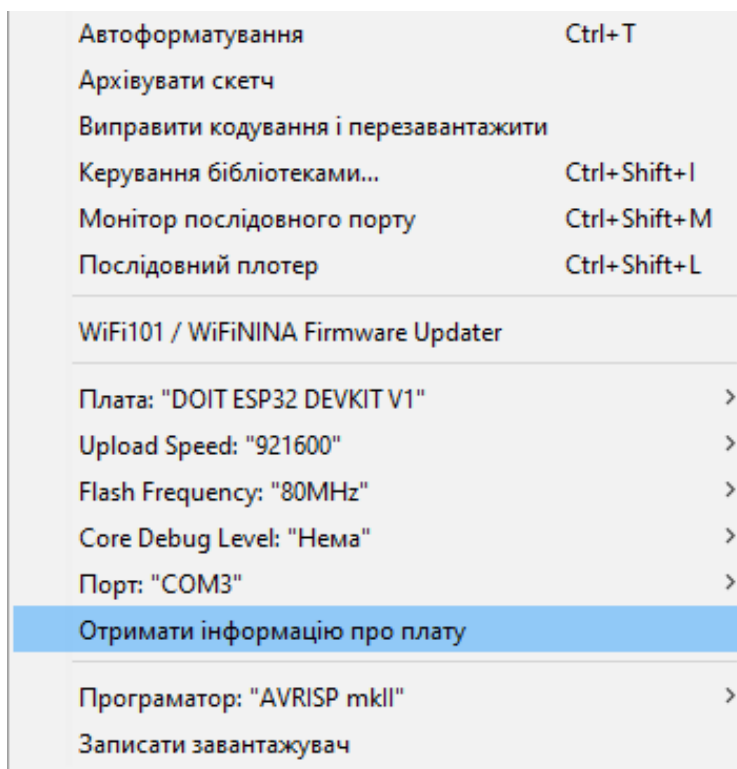


Рис. 32 – Параметри завантаження коду у мікроконтролер

```
j13walker@j13Walker-laptop: ~
--insecure : do not check that the server certificate hostname matches the remote
hostname. Using this option means that you cannot be sure that the
remote host is the server you wish to connect to and so is insecure.
Do not use this option in a production environment.
--tls-engine : If set, enables the use of a SSL engine device.
--tls-engine-kpass-sha1 : SHA1 of the key password to be used with the selected SSL engine.
--psk : pre-shared-key in hexadecimal (no leading 0x) to enable TLS-PSK mode.
--psk-identity : client identity string for TLS-PSK mode.
--proxy : SOCKS5 proxy URL of the form:
socks5h://[username[:password]@]hostname[:port]
Only "none" and "username" authentication is supported.

See https://mosquitto.org/ for more information.

j13walker@j13Walker-laptop:~$ mosquito_sub -t button_status_3_1
on
off
-
```

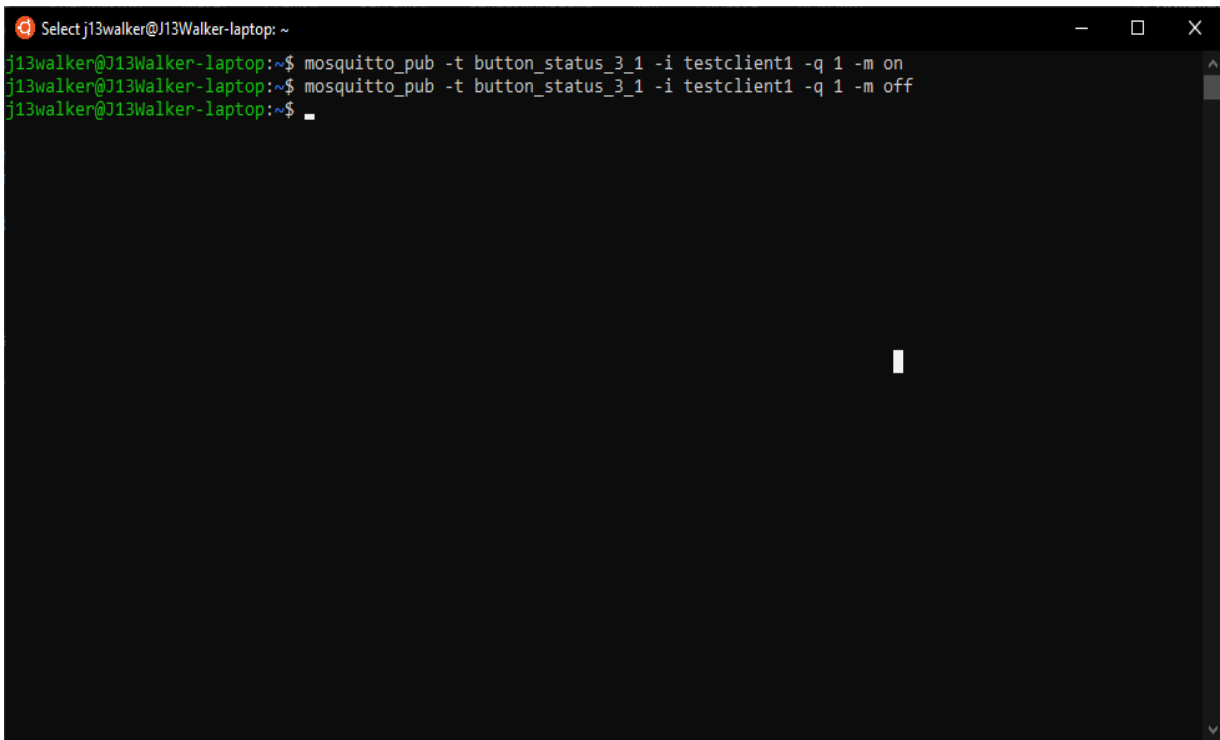
Рис. 33 – Вікно терміналу, у якому виводяться повідомлення з топіка, на який оформлено підписку додатка `mosquito_sub`

Завдання 2 частина 2. Підписка модуля ESP32 на повідомлення MQTT-брокера та виконання команд. Керування світлодіодами на віддаленому модулі.

Наступним кроком у вивченні взаємодії між застосуваннями Інтернету речей є отримання повідомлень з обраного топіка на MQTT-брокері та їхня обробка. Для цього нам також знадобиться розгорнутий брокер. За основу для виконання можна взяти код від частини 1, але цього разу нам потрібно працювати з функцією `callback()`, а саме: визначення варіантів виконання програми залежно від того, на який топік надійшло повідомлення, і що воно містить. У головному циклі програма має лише очікувати на повідомлення, а виконання дій має бути визначено у функції `callback()`. З попередньої частини відомо, що під час натискання на кнопку формується повідомлення у топік «`button_status_%номер варіанта%_%номер кнопки%`». Отже, можна визначити, чи була натиснута певна кнопка, і активувати світлодіод.

Виконання роботи

1. Відредагувати код першої частини завдання у такий спосіб, щоб при натисканні кнопки однократно здійснювалася відправка повідомлення «on» у топік «`button_status_%номер варіанта%_%номер кнопки%`». При відпусканні має відправлятися повідомлення «off» у той самий топік.
2. Завантажити код на один лабораторний стенд.
3. На інший лабораторний стенд завантажити код, у якому реалізовано керування світлодіодами за повідомленнями з топіка «`button_status_%номер варіанта%_%номер кнопки%`».
4. Перевірити роботу обох модулів окремо за допомогою клієнтських додатків `mosquitto_pub` [14] та `mosquitto_sub` (рис. 34).
5. Провести інтеграцію модулів шляхом одночасного підключення до Wi-fi.



```
Select j13walker@J13Walker-laptop: ~  
j13walker@J13Walker-laptop:~$ mosquitto_pub -t button_status_3_1 -i testclient1 -q 1 -m on  
j13walker@J13Walker-laptop:~$ mosquitto_pub -t button_status_3_1 -i testclient1 -q 1 -m off  
j13walker@J13Walker-laptop:~$
```

Рис. 34 – Приклад команди для тестування відправки повідомлень

Результатом роботи всієї системи має бути можливість керування світлодіодами одного стенда за допомогою кнопок на іншому.

Звіт з виконання лабораторної роботи має містити титульний аркуш, схему підключення та фото лабораторного стенда з під'єднаними контактами, згідно зі схемою, лістинг коду усіх трьох завдань з коментарями щодо його роботи та відповіді на контрольні питання.

Контрольні питання

1. Опишіть, які ще функції можуть виконувати GPIO, окрім цифрового вводу / виводу сигналу.
2. Назвіть GPIO, які не рекомендується використовувати не за призначенням, яке вказано у документації до модуля, та опишіть їхній функціонал.
3. Перелічіть апаратні і програмні протоколи зв'язку, які можна використовувати на ESP32.
4. Які протоколи зв'язку використовуються у цій лабораторній роботі?

5. Дайте визначення основним термінам MQTT-протоколу.
6. Дайте визначення терміна «Quality of Service». Опишіть рівні QoS.
7. Наведіть приклади використання MQTT-протоколу для побудування систем Інтернету речей.
8. Скільки клієнтів можна підключити до одного брокера?

Необхідний мінімум для підключення до корпоративної мережі

```
#include  
<WiFi.h>  
//Wifi  
library
```

```
#include "esp_wpa2.h" //wpa2 бібліотека для підключення до  
корпоративних мереж
```

```
#define EAP_IDENTITY "login" // тут необхідно вказати  
логін Вашого облікового запису в корпоративній мережі
```

```
#define EAP_PASSWORD "password" // пароль від  
корпоративного облікового запису
```

```
const char* ssid = "eduroam"; // назва корпоративної мережі.  
У нашому випадку «open.donnu.edu.ua»
```

```
int counter = 0;
```

```
void setup() {
```

```
  Serial.begin(115200);
```

```
  delay(10);
```

```
  Serial.println();
```

```
  Serial.print("Connecting to network: ");
```

```
  Serial.println(ssid);
```

```
  WiFi.disconnect(true); // відключення від мережі Wi-fi для  
встановлення нового підключення
```

```
  Wi-Fi.mode(WIFI_STA); // ініціалізація режиму роботи  
Wi-fi
```

```

    esp_wifi_sta_wpa2_ent_set_identity((uint8_t
*)EAP_IDENTITY, strlen(EAP_IDENTITY)); //
ідентифікація

    esp_wifi_sta_wpa2_ent_set_username((uint8_t
*)EAP_IDENTITY, strlen(EAP_IDENTITY)); // передача
імені користувача, яке збігається з логіном

    esp_wifi_sta_wpa2_ent_set_password((uint8_t
*)EAP_PASSWORD, strlen(EAP_PASSWORD)); // введення
пароля

    esp_wpa2_config_t config =
WPA2_CONFIG_INIT_DEFAULT(); // встановлення
поданих вище налаштувань за замовчуванням

    esp_wifi_sta_wpa2_ent_enable(&config); // дозвіл на роботу
за налаштуваннями

    WiFi.begin(ssid); // підключення до Wi-fi за заданими
налаштуваннями

    while (WiFi.status() != WL_CONNECTED) {

        delay(500);

        Serial.print(".");

        counter++;

        if(counter>=60){ // перезавантаження модуля через 30
секунд простою

            ESP.restart();

        }

    }

    Serial.println("");

    Serial.println("WiFi connected");

    Serial.println("IP address set: ");

```

```

    Serial.println(WiFi.localIP()); // виведення локальної
адреси модуля
}

void loop() {
    if (WiFi.status() == WL_CONNECTED) { // Якщо ми
підключилися до корпоративної мережі
        counter = 0; // скинути лічильник
        Serial.println("Wifi is still connected with IP: ");
        Serial.println(WiFi.localIP()); // проінформувати
користувача про його адресу
    } else if (WiFi.status() != WL_CONNECTED) { //
перепідключитись, якщо з'єднання розірвано
        WiFi.begin(ssid);
    }

    while (WiFi.status() != WL_CONNECTED) { // виводити
цяточки під час перепідключення
        delay(500);
        Serial.print(".");
        counter++;
        if(counter>=60){ // перезавантажити модуль за 30 секунд
простою
            ESP.restart();
        }
    }
}
}

```

Приклад коду для підключення до мережі Wi-fi із захистом wpa2 personal (через ssid та пароль)

```
#include
<PubSubClient.h>

const char* ssid = "....."; // Ім'я мережі, яке буде повідомлене під
час лабораторної роботи

const char* password = "....."; // Пароль до мережі, який буде
повідомлено під час лабораторної роботи

const char* mqtt_server = "....."; // Адреса MQTT-брокера, який
запущено локально на Вашому комп'ютері або його адресу буде
повідомлено під час лабораторної роботи

WiFiClient espClient; // ініціалізація Wi-fi клієнта

PubSubClient client(espClient); // ініціалізація MQTT-клієнта

long lastMsg = 0;

char msg[50];

int value = 0;

void setup_wifi() { // функція, яка налаштовує Wi-fi

  delay(10);

  // We start by connecting to a Wi-fi network

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

```

`void callback(char* topic, byte* payload, unsigned int length) { //`
Функція, яка позначає реакцію модуля на появу повідомлення у топіку, який було вказано при виклику функції

```

Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
}
Serial.println();

```

// Як приклад було наведено вмикання та вимикання світлодіоду під час появи певного повідомлення в обраному топіку

```

// Switch on the LED if an 1 was received as first character
if ((char)payload[0] == '1') {

```

```

    digitalWrite(BUILTIN_LED, LOW); // Turn the LED on (Note
that LOW is the voltage level

    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
} else {

    digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by
making the voltage HIGH

}
}

void reconnect() { // функція для підключення до брокера та
відправлення повідомлення у топик під час підключення

// Loop until we're reconnected

while (!client.connected()) {

    Serial.print("Attempting MQTT connection...");

    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    // Attempt to connect
    if (client.connect(clientId.c_str())) {

        Serial.println("connected");

        // Once connected, publish an announcement...
        client.publish("outTopic", "hello world");

        // ... and resubscribe
        client.subscribe("inTopic");
    } else {

        Serial.print("failed, rc=");

        Serial.print(client.state());

```

```

Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);           // Initialize the
  BUILTIN_LED pin as an output
  Serial.begin(115200);
  setup_wifi(); //налаштування Wi-fi
  client.setServer(mqtt_server, 1883); // визначення адреси MQTT-
  брокера
  client.setCallback(callback); // визначення відповіді
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
  }
}

```



```
snprintf(msg, 50, "hello world #%ld", value);
Serial.print("Publish message: ");
Serial.println(msg);
client.publish("outTopic", msg);
// тут відбувається відправлення повідомлення з певним
числовим значенням
}
}
```

Список рекомендованих джерел

1. <https://www.espressif.com/en/products/hardware/esp32/overview>
2. https://uk.wikipedia.org/wiki/Аналого-цифровий_перетворювач
3. <http://www.i2c-bus.org/>
4. https://ru.wikipedia.org/wiki/Универсальный_асинхронный_приёмопередатчик
5. https://ru.wikipedia.org/wiki/Serial_Peripheral_Interface
6. <https://uk.wikipedia.org/wiki/Світлодіод>
7. <https://www.arduino.cc/reference/en/language/functions/communication/serial/>
8. <https://www.arduino.cc/en/Reference/WiFi>
9. <https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/examples/WiFiClientBasic/WiFiClientBasic.ino>
10. <https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi/examples/WiFiClientEnterprise>
11. <https://mosquitto.org/man/mqtt-7.html>
12. <https://pubsubclient.knolleary.net/>
13. https://mosquitto.org/man/mosquitto_sub-1.html
14. https://mosquitto.org/man/mosquitto_pub-1.html

Лабораторна робота № 3

Тема: Застосування апаратних засобів введення / виведення інформації у IoT.

Мета: розглянути взаємодію між двома пристроями Інтернету речей шляхом передачі текстової інформації та її виведення на LCD-дисплей.

Задачі:

- 1) ознайомитися з введенням інформації за допомоги матричної клавіатури;
- 2) ознайомитися із виведенням інформації на LCD дисплей;
- 3) дослідити особливості передачі текстових даних між пристроями.

Теоретична частина

Ознайомлення з введенням інформації за допомогою матричної клавіатури 2 x 9. Таке рішення дуже поширене серед охоронних систем, наприклад для введення пароля або здійснення певних налаштувань. Як приклад такого периферійного пристрою розглянемо наявну на лабораторному стенді матричну клавіатуру. Загалом мембранна матрична клавіатура (далі – клавіатура) побудована за схемою, наведеною на рис. 35. Для її підключення до ESP32 необхідно задіяти 7 GPIO. Як можна бачити, для зменшення кількості задіяних виводів кнопки об'єднують в рядки і стовпці, і потім, подаючи одиницю послідовно на кожен рядок, ми зчитуємо значення зі стовпців на предмет присутності на них логічного сигналу «1». Знаючи, на який рядок ми подали напругу і на якому стовпці вона з'явилася ми можемо обчислити, яка саме кнопка натиснута. Можна робити навпаки – напругу подавати на стовпці, а опитувати рядки – різниці немає. Для спрощення роботи з клавіатурою потрібно використовувати бібліотеку `Keypad.h`, яка наявна у каталозі бібліотек Arduino IDE.

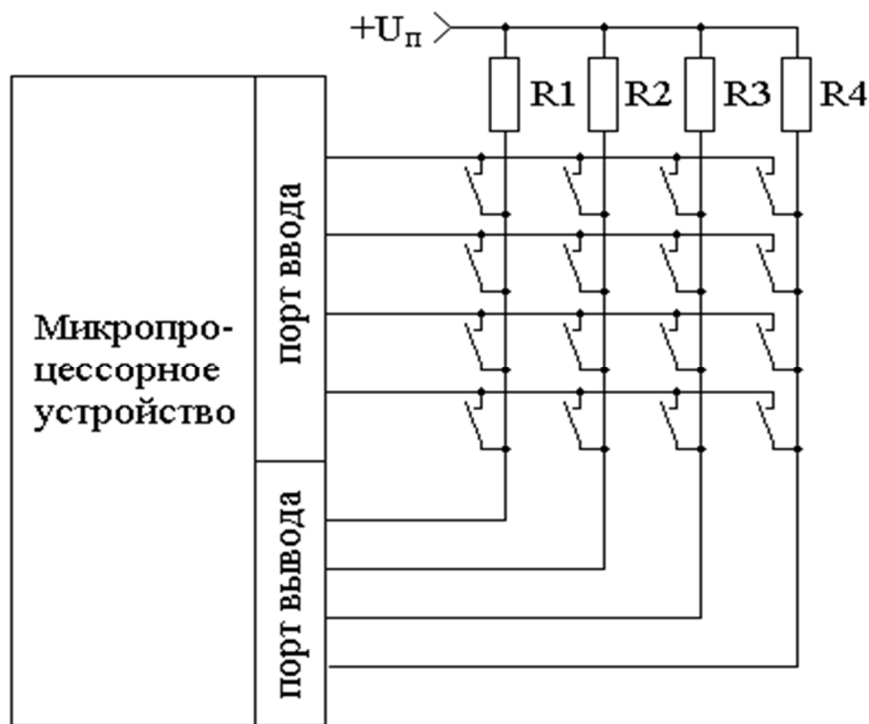


Рис. 35 – Принципова схема підключення матричної клавіатури

У випадку з ESP32 схема підключення буде виглядати, як на рис. 36. Щодо програмної реалізації, існує бібліотека keypad.h [1], яка дозволяє спростити обробку сигналів при натисканні на кнопки. Детальний опис функцій та приклади коду можна знайти за посиланням [2].

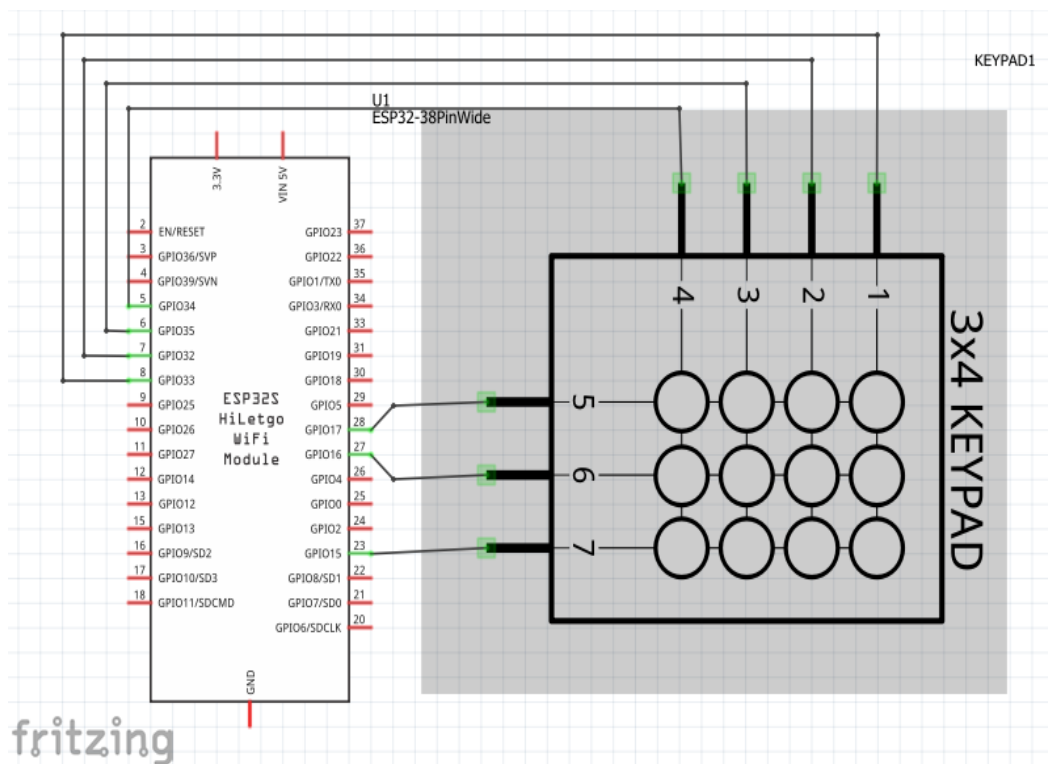


Рис. 36 – Схема підключення матричної клавіатури до ESP32

Ознайомлення із виведенням інформації на LCD-дисплей

Наступним периферійним пристроєм, який необхідно розглянути, є LCD-дисплей. Прикладом такого дисплею є модель 2004, наявна на лабораторному стенді. Наявний на стенді дисплей є текстовим дисплеєм на 20 символів у чотири рядки, який керується контролером HD44780 [3]. Для спрощення керування та схеми підключення до дисплея під'єднано I2C модуль розширення на базі чипа PCF8574T [4]. Для керування цим дисплеєм використовується бібліотека LiquidCrystal_PCF8574 [5], яка створена на основі бібліотек LiquidCrystal та застосовує функціонал бібліотеки Wire (для комунікації по I2C та TWI). Подана бібліотека рекомендована для виконання лабораторних робіт, хоча в наявності велика кількість інших, завдяки тому що всі вони мають відкритий вихідний код.

Типова схема підключення цього модуля дисплея до ESP32 наведена на рис. 37. На поданий дисплей можна виводити лише текстові дані, тому що він розбитий на знакомісця та не передбачає роботи з графікою. Бібліотека підтримує можливість керування курсором, позиціонування на необхідну координату (знакомісце) та очистку дисплею. З повним функціоналом можна ознайомитися за посиланням вище. Також існує бібліотека, у якій реалізовано функціонал меню, що спрощує розробку інтерфейсу користувача на базі пристрою Інтернету речей [6].

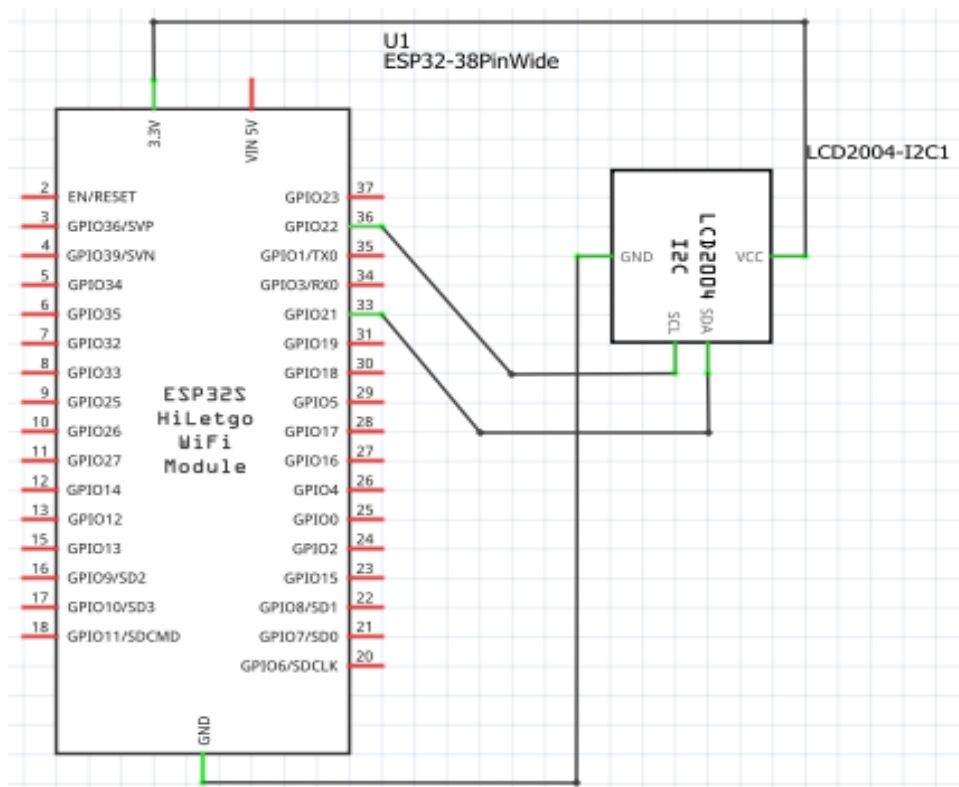


Рис. 37 – Схема підключення LCD-дисплея з модулем i2c до ESP32

Запуск і тестування

Для управління текстом на РК-дисплеї є три основні функції:

1) `begin` (підсумкові стовпці, загальні рядки). Ця функція використовується всередині `setup ()` для ініціалізації розміру дисплея, що використовується. Якщо це 20x4, то: `lcd.begin (20,4)`, інакше, якщо це 16x2, тоді: `lcd.begin (16,2)`;

2) `setCursor` (номер стовпчика, номер рядка). Ця функція поміщає курсор на пристрої в потрібне положення. Будь-який текст, який відображається після цієї функції, почнеться із вказаної Вами позиції. Наприклад, використовуйте: `lcd.setCursor (4,0)`, тобто п'ятий стовпець і перший рядок (починаючи з 0,0);

3) `print` («текст»). Ця функція використовується для друку тексту на РК-дисплеї. Яким би не був рядок всередині «», вона відображається на дисплеї.

Практична частина

Виконання роботи

1. Написати програму для зібраної схеми, яка буде виконувати такі дії: при натисканні на клавішу в монітор порту має надходити значення цієї клавіші.
2. Перевірити правильність роботи програми, натискаючи на всі клавіші матричної клавіатури, контролюючи відповідність натиснутих клавіш та інформації, яка з'являється на дисплеї.

Контрольні питання

1. Напишіть мініпрограми, які будуть реалізовувати функцію затримки, використовуючи, замість `delay ()`, функцію `millis ()` /.
2. За допомогою якої команди можна скинути дисплей?
3. Опишіть принцип роботи матричної клавіатури:

4. Чи буде працювати клавіатура, якщо підключити не всі висновки? Якщо так, то у який спосіб?

5. Як пов'язані кількість шин з'єднання матричного сенсора з кількістю клавіш?

Список рекомендованих джерел

1. <https://github.com/Chris--A/Keypad>
2. <https://playground.arduino.cc/Code/Keypad/>
3. <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>
4. https://ecee.colorado.edu/~mcclurel/Philips_I2C_IO_Expander_PCF8574_4.pdf
5. https://github.com/mathertel/LiquidCrystal_PCF8574

Додаток 1

```
1  int PinOut[4] {5, 4, 3, 2}; // Піни виходу
2
3  int PinIn[4] {9, 8, 7, 6}; // Піни входу
4  int val = 0;
5  const char value[4][4]
6
7  { {'1', '4', '7', '*'},
8    {'2', '5', '8', '0' },
9    {'3', '6', '9', '#'},
10   {'A', 'B', 'C', 'D'}
11 };
12 // подвійний масив, що позначає кнопку
13 int b = 0; // змінна, якій присвоюється число з масиву (номер кнопки)
14
15 void setup()
16 {
17   pinMode (2, OUTPUT); // ініціалізуються порти на вихід
18                       // (подають нулі на стовпці)
19   pinMode (3, OUTPUT);
20   pinMode (4, OUTPUT);
21   pinMode (5, OUTPUT);
22
23   pinMode (6, INPUT); // ініціалізуються порти на вхід з підтяжкою до
24                       // плюси (приймають нулі на рядках)
```



```

25  digitalWrite(6, HIGH);
26  pinMode (7, INPUT);
27  digitalWrite(7, HIGH);
28  pinMode (8, INPUT);
29  digitalWrite(8, HIGH);
30  pinMode (9, INPUT);
31  digitalWrite(9, HIGH);
32
33  Serial.begin(9600); // відкриваємо Serial-порт
34  }
35
36  void matrix () // створюємо функцію для читання кнопок
37  {
38  for (int i = 1; i <= 4; i++) // цикл, що передає 0 за всіма стовпцями
39  {
40  digitalWrite(PinOut[i - 1], LOW); // якщо і менше 4, то відправляємо 0
41  //на ніжку
42  for (int j = 1; j <= 4; j++) // цикл, що приймають 0 по рядках
43  {
44  if (digitalRead(PinIn[j - 1]) == LOW) // якщо один із зазначених портів
45  // входу дорівнює 0, то..
46  {
47  Serial.println( value[i - 1][j - 1]); // то b дорівнює значенню з
// подвійного масиву
delay (175);  }
}
}

```

```
digitalWrite(PinOut[i - 1], HIGH); // подаємо назад високий рівень }  
}  
  
void loop()  
{  
matrix(); // використовуємо функцію опитування матричної клавіатури  
}
```

ДЛЯ ПОДАТОК

Навчальне видання

*Розанов Іван Євгенович
Сергієнко Сергій Петрович
Чернов Дмитро Вікторович*

**МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ
ЛАБОРАТОРНИХ РОБІТ З КУРСУ
ІНТЕРНЕТ РЕЧЕЙ**

Редактор А. О. Цяпало

Технічний редактор Т. О. Алімова

Підписано до друку 22.01.2020 р.
Формат 60x84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 3,49
Тираж 15 прим. Зам. № 138

Донецький національний університет імені Василя Стуса,
21021, м. Вінниця, вул. 600-річчя, 21
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК № 5945 від 15.01.2018 р.