Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет радіотехніки та телекомунікацій
(повне найменування інституту, факультету)

Інформаційної технології електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

Магістр
(ступінь вищої освіти)

на тему Дослідження можливостей однозначного керування та відеозв'язку з модулем ESP32 з використанням Raspberry Pi

Виконав: студент(ка) VI курсу, групи РТ-519М

Спеціальності 172 Телекомунікації та радіотехніка (код і найменування спеціальності)

Освітня програма (спеціалізація) Інтелектуальні технології мікросистемної радіоелектронної техніки

Шкрум А.Ю.
(прізвище та ініціали)

Керівник Фарафонов О.Ю.
(прізвище та ініціали)

Рецензент Воскобойник В.О.
(прізвище та ініціали)

2020

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформаційних та радіотехнологій факультет радіоелектроніки та телекомунікаційних систем

Кафедра інформаційних технологій електронних засобів

Ступінь вищої освіти Магістр

Спеціальність 172 "Телекомунікації та радіотехніка"

(код і найменування)

Освітня програма (спеціалізація) Інженерія інформаційних технологій мікросистемної техніки радіоелектронних пристроїв

(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри Шило Г.М.

к.т.н. доц. кафедри ІТЕЗ

«19» грудня 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Шкуди Андрій Юрійович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження можливостей оптичного перехвату та підробки трафіку протоколу ESP32 з використанням Raspberry Pi

керівник проєкту (роботи) Рогаченко Олексій Юрійович к.т.н. доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «15» січня 2020 року № 450

2. Строк подання студентом проєкту (роботи) 19 грудня 2020 року

3. Вихідні дані до проєкту (роботи) Використання модуля ESP32 по протоколу для передачі відео в бібліотеку додаткових пакетів відео та передачі відео веб-сервер Raspberry Pi

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Опис основні положення завдань; Розробка перехвату передачі відео по протоколу, створення веб-інтерфейсу Raspberry Pi, підключення управління через Raspberry Pi. Розробка економічної частини дослідницької проєкту, Охорона праці на робочих місцях при експлуатації

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-2	Фадєєва О.Ю. к.т.н. доцент кафедри ПЕЗ	01.10.20	
3	Левченко Н.М. професор	15.10.20	
4	Якімчук Ю.В.	03.11.20	
5	Поспєєва І.Е	11.12.20	

7. Дата видачі завдання «15» жовтня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз сучасного стану	3 тижня	
2	Огляд області дослідження та постановка завдань	2 тижня	
3	Розробка методики управління	4 тижня	
4	Моделювання роботи системи	3 тижня	
5	Розрахунок економічної ефективності	1	
6	Оформлення презентації та доповіді	1	
7	Оформлення пояснювальної записки	2	

Студент(ка) _____ (підпис) Якімчук А.Ю. (прізвище та ініціали)

Керівник проєкту (роботи) _____ (підпис) Фадєєва О.Ю. (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: сторінок 110, рисунків 19, таблиць 6, джерел 14, додатків 3.

Об’єкт досліджень: керування по відеозв’язку ESP32 з використанням Bluetooth та Raspberry Pi.

У першому розділі проводиться огляд управління та передача відео з ESP32 по Bluetooth.

У другому розділі проводиться розробка програми управління ESP32 за допомогою Bluetooth та Raspberry PI.

У третьому розділі проводиться розрахунок економічної ефективності та теоретичної окупності науково-дослідницького проекту.

У четвертому розділі розглядаються питання з охорони праці та безпеки в надзвичайних ситуаціях.

МІКРОКОНТРОЛЕР, ПРОГРАМА, ARDUINO, RASPBERRY PI,СКРИПТ.

ABSTRACT

Explanatory note to the diploma project: pages 110, figures 19, tables 6, sources 14, appendices 3.

Object of study: control of ESP32 video communication using Bluetooth and Raspberry Pi.

The first section provides an overview of control and video transfer from ESP32 to Bluetooth.

The second section develops the ESP32 control program using Bluetooth and Raspberry PI.

The third section calculates the economic efficiency and theoretical payback of the research project.

The fourth section addresses issues of occupational safety and health in emergencies.

MICROCONTROLLER, PROGRAM, ARDUINO, RASPBERRY PI, SCRIPT.

ЗМІСТ

ВСТУП

Стрімкий розвиток електроніки, наноелектроніки та мікропроцесорної техніки, інформаційно-комунікаційних технологій надає великі можливості у сферах комунікації та зв'язку, тому у сучасному житті ми можемо користуватись пристроями, які раніше були недоступні.

Сучасні пристрої побудовані за допомогою мікроконтролерів.

Мікроконтролери широко використовуються у побутовій техніці, автомобільній електроніці, аерокосмічній та військовій галузях, для охоронних та моніторингових цілей.

Мікроконтролер – це спеціалізований мікроелектронний програмований прилад, призначений для використання у керуючих пристроях, системах передачі відео та даних, у системах керування технологічними процесами. На сьогоднішній день мікроконтролери є найбільш поширеними з потужних і гнучких засобів розробки електронних схем. Це найважливіший компонент будь-якої інформаційно-вимірювальної системи.

Мікроконтролери також широко застосовуються в автомобільній електроніці. Наприклад автомобіль «Peugeot 206» має на борту 27 мікроконтролерів, а в автомобілях високого класу, таких як «BMW» сьомої серії, використовується понад 60 мікроконтролерів. Мікроконтролери керують вприском палива, світлотехнікою, двигунами двірників, склопідіймачів та дзеркал заднього виду.

Використання мікроконтролерів у бездротових системах безпеки відноситься к новим рішенням, в яких використовуються найсучасніші технології. Відеоспостереження можна охарактеризувати як інтелектуальну техніку обробки відео, створену для надання допомоги персоналу безпеки шляхом надання оповіщень у режимі реального часу та підтримки ефективного відеоаналізу. Це допоможе швидко реагувати у різноманітних ситуаціях.

Щоб написати програму, треба вивчити особливості мікроконтролера, його технічні характеристики, призначення його виводів, особливості застосування у схемі.

При розробці пристрою виникає необхідність у виборі мікроконтролера, що задовольняє вимогам по продуктивності, надійності, умовам застосування.

В середині мікроконтролера є пам'ять даних, пам'ять програм, генератор тактових імпульсів, таймери, лічильники, паралельні та послідовні порти. Система мінімальної конфігурації на основі мікроконтролера складається з блока живлення, безпосередньо мікросхеми контролера та кількох пасивних елементів.

Використання мікроконтролерів дозволяє конструювати пристрої, що володіють такими якостями, як невеликі габарити, відносна дешевизна, простота і надійність, сумісність з персональним комп'ютером через стандартні інтерфейси.

У світі багато розробників та виробників мікроконтролерів. З використанням програмного забезпечення можна написати програму для мікроконтролера та налаштувати її, а з допомогою програматору занести програму у вигляді hex-коду в пам’ять мікроконтролера. Увімкнувши мікроконтролер, починає працювати програма.

Програмування мікроконтролерів є невід’ємною частиною їх застосування. Технологія, що дозволяє програмувати електронні компоненти систем - внутрішньосхемне програмування. Це програмування об'єднує процес програмування та тестування на виробництві. Це дає можливість внесення оперативних змін без зупинки процесу виробництва.

У широкому розумінні відеоспостереження можна охарактеризувати як інтелектуальну техніку обробки відео, створену для надання допомоги персоналу безпеки шляхом надання надійних оповіщень у режимі реального часу та підтримки ефективного відеоаналізу.

1 УПРАВЛІННЯ ТА ПЕРЕДАЧА ДАНИХ ESP32 ПО BLUETOOTH

1.1 ESP32-підключення

Для пристрою передачі даних відео був обран мікроконтролер ESP32. Він володіє відмінною масштабованістю, не залежить від виробника комплектуючих пристроїв, та не належить до вузького спектру використання.

ESP32 став лідером у своєму сегменті.

ESP32 — це серія мікроконтролерів типу «система на кристалі», що мають інтегровані контролери Wi-Fi і Bluetooth, мають низьке енергоспоживання і невисоку ціну. ESP32 створений та розроблений компанією Espressif Systems. Це китайська компанія, яка розташована у Шанхаї, а виробляється ESP32 компанією TSMC. ESP32 є наступником мікроконтролера ESP8266. [9]

Мікроконтролер ESP32 випущен у 2015 році і має значну перевагу над аналогами не тільки через низьку ціну та низьке енерговживання. В одній мікросхемі є поєднання Wi-Fi і Bluetooth. Він оснащений двохядерним 32-бітовим процесором, який працює на частотах 80, 160 або 240 МГц. У систему інтегровані антенні комутатори, радіочастотні компоненти, фільтри, підсилювачі, модулі управління живленням.

Китайская фирма Espressif Systems Pte. Ltd. — это полупроводниковая фаблесс-компания с головным офисом в парку високих технологій Шанхая (Shanghai Zhangjiang High-Tech Park, KHP), хорошо известная во всем мире как одна из ведущих производителей простых, бюджетных чипов Bluetooth и Wi-Fi. Фирма не имеет собственных производственных мощностей и производит свою продукцию під торговой маркой Espressif на OEM-предприятиях, таких як Taiwan Semiconductor Manufacturing Company (TSMC). [5]

На базі цього мікроконтролера існують модульні SMT плати. Це дозволяє створювати прототипи, макети майбутнього пристрою на базі цього

мікроконтролера без необхідності створення самої плати. Все це значно покращує якість виробництва та підготовчий етап.

Це потужний та недорогий мікроконтролер.

Підключається ESP32 до комп'ютера за допомогою USB роз’єму.

Особливості ESP32 включають в себе наступне: [10]

- процесор Xtensa, двоядерний (або одноядерної) 32-розрядний LX6 мікропроцесор, що працює на 160 або 240 МГц і виконує до 600 DMIPS;
- ультра низька потужність (ОТП) співпроцесора;
- пам'ять: 520 Кб пам'яті SRAM;
- бездротовий зв'язок wi-fi: 802.11 b/g/N;
- bluetooth B4.2 БР/EDR і BLE;
- 12-розрядний АЦП до 18 каналів;
- 2 × 8-біт цапи;
- 10 × сенсорних датчиків (ємнісних датчиків і контролерів);
- датчик температури;
- 4 × СВО;
- 2 × i2s для інтерфейсів;
- 3 × UART;
- хост-контролер SD/SDIO/CE-ATA/MMC/eMMC;
- підпорядкований контролер SDIO/SPI;
- інтерфейс Ethernet Mac з виділеними DMA і стандарти IEEE 1588 точного часу за протоколом підтримки;
- інфрачервоний пульт дистанційного управління (передавач/приймач, до 8 каналів);
- можливість підключення двигунів та світлодіодів через ШІМ-вихід;
- ультра низька потужність;
- аналоговий передпідсилювач;
- безпечне завантаження;
- шифрування флеш;

- 1024-бітний ключ, до 768 біт для клієнтів;
- криптографічне апаратне прискорення AES, SHA-2, RSA;
- криптографії на основі еліптичних кривих (ECC);
- генератор випадкових чисел (ГВЧ);
- управління живленням:
- внутрішній низький регулятор відключення;
- індивідуальний енергетичний домен для RTC;
- прокидання з переривання від GPIO, таймера, вимірювання АЦП, переривання ємнісного сенсорного датчика [10]

Для програмування ESP32 можна використовувати такі програмні додатки, як Mongoose OS, MicroPyhon, NodeMcu, Arduino, Platformio.org. За їх допомогою користувачи можуть створювати власні додатки з застосуванням GCC 5.2.0 та C ++. [1].

Модуль ESP32 відрізняється від попередніх вдосконаленним обладнанням, збільшенням функціоналу та об’ємом пам'яті.

На мікроконтролері ESP32 встановлен швидкий Wi-Fi та Bluetooth. Канали Wi-Fi і Bluetooth виконані у вигляді окремих апаратних блоків. Працюють вони під управлінням процесорів Xtensa виробництва фірми Tensilica.

У ESP32 потужний процесор, за допомогою якого можливо реалізовувати різні проекти, от легкіх до складних. Об’єм пам’яті у модулі ESP32 512 Кб. ESP32 також має контакти,до яких прикріплені ємнісні сенсорні датчики. ESP32 має вісімнадцять 12-бітних АЦП каналів. ESP32 має камеру OV2640, декілька GPIO для підключення периферійних пристроїв, має роз’єм для карт microSD. MicroSD користовується для зберігання зображень, зроблених за допомогою камери. [2]

ESP32 має і недоліки, такі як відсутність бібліотек для підтримуючих сенсорів і мала кількість драйверів. У майбутньому недолікивони будут подолані.

Для подальшої роботи підключаємо ESP32 к комп’ютеру через USB.

У рисунку 1.1 зображений мікроконтролер ESP32. [1]

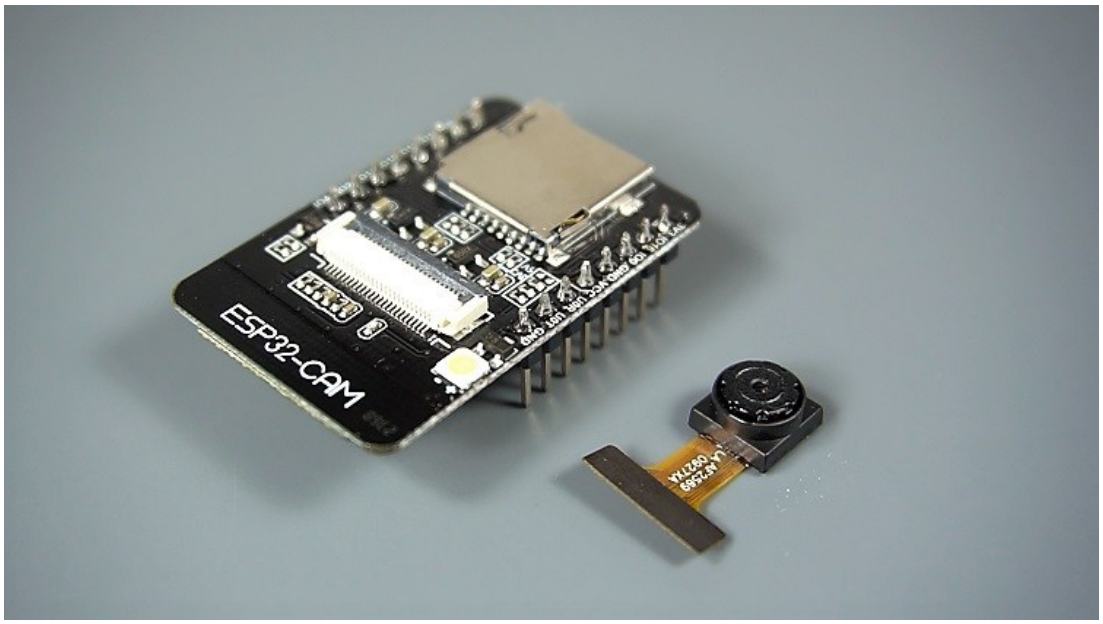

Рисунок 1.1- Мікроконтролер ESP32

ESP32 працює в якості центрального процесора за підтримкою Open CPU, і як підлеглий пристрої в режимі slave device, яким керує зовнішній мікроконтролер.

Для ESP32 Espressif застосувала операційну систему реального часу FreeRTOS.

Розглянемо встановлення ESP32 у середовище Arduino IDE. [6]

Для встановлення виконуємо наступні дії:

- відкриваємо вікно налаштувань у середовищі Arduino IDE. Вибираємо пункт меню «Файл> Налаштування» («File> Preferences»);
- у рядку «Додаткові посилання для Менеджера плат» копіюємо адресу: https://dl.espressif.com/dl/package_esp32_index.json;
- відкриваємо менеджер плат. Натискаємо на рядок «Інструменти> Плата> Менеджер плат» («Tools> Boards> Boards Manager»);

- вводимо в пошуку «ESP32» і натискаємо кнопку «Install» для «ESP32 by Espressif Systems».

Після встановлення ESP32 у середовищі Arduino IDE проводимо перевірку встановлення.

Виконуємо послідовно наступні дії:

- запускаємо середовище Arduino IDE;
- знаходимо в меню рядок «Інструменти> Плата» («Tools> Board») і вибираємо потрібну плату;
- вибираємо порт у меню «Порт»;
- відкриваємо «Файл> Приклади> WiFi (ESP32)> WiFi Scan» («File> Examples> WiFi (ESP32)> WiFi Scan»);
- натискаємо кнопку «Завантаження» («Upload») в середовищі Arduino IDE;
- на екрані бачимо повідомлення "Done uploading» («Завантаження завершено»);
- відкриваємо вікно послідовного COM-порту (Serial Monitor) середовища Arduino IDE і налаштуємо швидкість передачі на 115 200 бод;
- натискаємо кнопку «Enable» на платі ESP32 і бачимо мережі, які доступні для ESP32.

Після перевірки завантажуємо на ESP32 свій скетч. Вибіраємо ім'я плати і порт COM.

Виконуємо наступні дії:

- утримуємо кнопку «BOOT» на платі ESP32;
- нажімаємо кнопку «Завантажити» в середовищі Arduino IDE, щоб завантажити свій скетч;
- побачимо «Підключення» у Arduino IDE, та відпускаємо кнопку «BOOT»;
- бачимо повідомлення "Готово".

На ESP32 запущений новий скетч. Натискаємо кнопку «ВКЛЮЧИТИ», перезапускаємо ESP32 і запускаємо новий завантажений скетч.

За допомогою датчика PIR можливо виявляти рух. Якщо виявленно рух, ESP32 запускає таймер і включає світлодіод на кількість секунд, заданих раніше. Світлодіод буде відключений, коли зворотний відлік таймера буде закінчен.

Щоб ініціювати подію за допомогою датчика PIR, використовується переривання. Для встановлення переривання в IDE Arduino, будемо використовувать функцію attachlnterrupt ():

// attachInterrupt(digitalPinToInterrupt(GPIO), function, mode).

Для фактичного встановлення GPIO у якості контакту переривання, використовуємо функцію dagatalPinTolnterrapt.

Використання можливо у 5 різних режимах:

- «low»: переривання буде запускатися щоразу, якщо на виводі є LOW;
- «високо»: переривання буде запускатися щоразу, якщо на виводі є «високий» рівень;
- «міна»: переривання буде запускатися щоразу, коли вивід змінює значення;
- «падіння»: коли зміна з «високого» на «низькій»;
- «висходящий»: вивод змінюється з «низькій» до «високий»

Будемо використовувати таймер замість функції затримки ().

Широко використовується фукція затримки «delay ()». Програма повинна чекати кілька мілісекунд до переходу у наступний рядок коду:

// delay(time in milliseconds).

Для повернення кількості мілісекунд, які пройшли з моменту запуска програми використовуємо функцію «millis»:

// millis().

1.2 Управління ESP32 по Bluetooth

ESP32 поставляється не тільки з Wi-Fi, а також з Bluetooth і Bluetooth Low Energy (BLE).

ESP32 містить Bluetooth link controller і радіочастотний блок Bluetooth baseband, може працювати як в режимі класичного Bluetooth BR / EDR, так і в економному режимі BLE. [3]

Це дозволяє ефективно застосовувати мікроконтролер в різноманітних проектах в поєднанні з модулями ARDUINO і Raspberry Pi.

Bluetooth Low Energy, скорочено BLE, це енергозберігаючий варіант Bluetooth, дозволяє споживати дуже мало енергії. BLE споживає приблизно в 100 разів менше енергії, ніж Bluetooth.

Розглянемо як за допомогою програми Bluetooth виконуються обмін даними між смартфоном або комп’ютером і мікроконтролером ESP32.

Завдання полягає в тому, щоб встановити бездротове з'єднання з ESP32.

Для розробці використовуємо середовище Arduino IDE. [4]

Виконуємо налаштування програми:

- додаємо програму з файлу бібліотеки BluetoothSerial, яка переводить модуль в режим Bluetooth SPP;
- завантажуємо останню версію бібліотеки з її офіційної сторінки на GitHub. Посилання на скачування Bluetooth Serial Library: #include "BluetoothSerial.h";
- робимо вибір об'єкта для проведення передачі даних: BluetoothSerial ESP_BT;
- встановлюємо швидкість послідовної передачі даних і ініціалізацію сигналу Bluetooth з ім'ям пристроя;
- друкуємо значення змінної в послідовний монітор, для перевірки інформації, одержаної з ARDUINO.

Далі наведен повний код програми:

```
BluetoothSerial ESP_BT; // Об'єкт для Bluetooth

int incoming;
int LED_BUILTIN = 2;
void setup () {; // Вводимо установку
Serial.begin(115200); // Запускаємо послідовний монітор зі швидкістю 115200
ESP_BT. begin ("ESP32_LED_Control"); // Задаємо ім'я нашого пристрою Bluetooth
Serial.println("Bluetooth Device is Ready to Pair"); // По готовності повідомляємо, що пристрій готовий до об'єднання.

pinMode (LED_BUILTIN, OUTPUT); // Задаємо контакт підключення світлодіода як вихідний
}
void loop () {

if (ESP_BT. Available ()) // Перевіряємо, отримали що-небудь від Bluetooth модуля
{
incoming = ESP_BT. read (); // Читаємо, що отримали
Serial.print("Received:"); Serial.println(incoming);

if (incoming == 49); // Якщо значення дорівнює одиниці, включаємо світлодіод
{
digitalWrite (LED_BUILTIN, HIGH);
ESP_BT. println ("LED turned ON");
```

```
}

if (incoming == 48); // Якщо значення дорівнює нулю, вимикаємо світлодіод
{
digitalWrite (LED_BUILTIN, LOW); // Включення світлодіода
ESP_BT. println ("LED turned OFF"); // Виключення світлодіода
}
}
delay (20); // Задаємо затримку
```

У диспетчері пристроїв знаходимо номер COM-порту. У Arduino IDE вибираємо даний порт і відкриваємо "Монітор порту".

Починаємо вибирати у блоці Terminal налаштування Force on.

На моніторі порту відображаються команди, які можна посилати на модуль. Відображення команд порту наведено у рисунку 1.2.

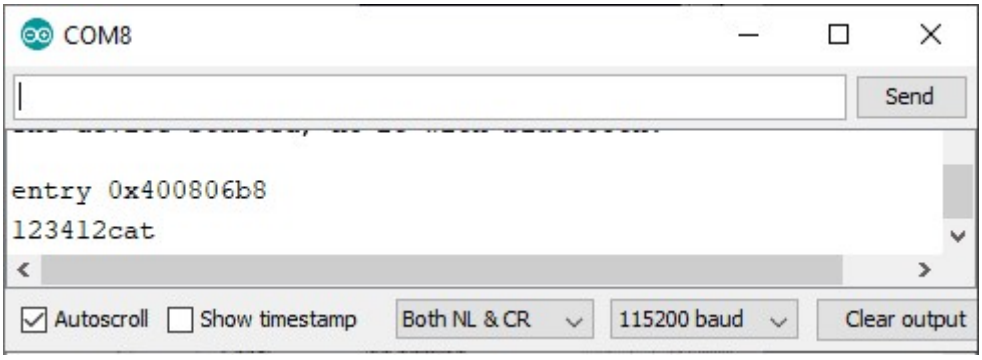

Рисунок 1.2 – Відображення команд порту

Функцію зворотного виклику додаємо в скетч через register_callback (). Так можемо відстежувати момент з'єднання комп'ютера чи смартфона.

Код програми зворотного виклику:

```
#include "BluetoothSerial.h"; // Підключаємо бібліотеку Bluetooth
```

```
BluetoothSerial SerialBT;
void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param){
  if(event == ESP_SPP_SRV_OPEN_EVT){; // Вводимо зворотні сигнали
    Serial.println("Client Connected"); // Підключення клієнта
  }
}
void setup () {; // Вводимо установку
  Serial.begin(115200); // Встановлюємо швідкість затримки
  SerialBT.register_callback(callback); // Передавання сігналу
   if (!SerialBT.begin("ESP32")) {
    Serial.println("An error occurred initializing Bluetooth"); // Під час ініціалізації Bluetooth сталася помилка
  }
  else {
    Serial.println("Bluetooth initialized"); // Bluetooth ініціалізований
  }
}
void loop () {
  while (SerialBT.available()) {: // SerialBT доступний
    Serial.write(SerialBT.read()); // Запис та читання Bluetooth
  }
  delay (50); // Задаємо затримку
}
```

Знову запускаємо скетч і встановлюємо з'єднання через Putty. В моніторі порту з'явиться повідомлення Client Connected Putty.

```
 #include "BluetoothSerial.h": //Підключення Bluetooth
BluetoothSerial SerialBT;
void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {
```

```
  if (event == ESP_SPP_SRV_OPEN_EVT) {: // Вводимо зворотні сигнали
   Serial.println("Client Connected"); // Підключення клієнта

     Serial.println("Client Connected has address:"); //Підключення клієнта за адресою

   for (int i = 0; i <6; i++) {
    Serial.printf("%02X", param> srv_open.rem_bda[i]);
    if (i <5) {
     Serial.print(":");
    }
   }
   Serial.println("--------------");
  }
 }

 void setup() {: // Вводимо установку
  Serial.begin(115200); // Встановлюємо швідкість затримки

  SerialBT.register_callback(callback); // Передавання сігналу
  if (!SerialBT.begin("ESP32")) {
   Serial.println("An error occurred initializing Bluetooth"); // Під час ініціалізації Bluetooth сталася помилка

  }
  else {
   Serial.println("Bluetooth initialized"); // Bluetooth ініціалізований
  }
 }
```

```
void loop() {
  while (SerialBT.available()) {
    Serial.write(SerialBT.read()); // Запис та читання Bluetooth
  }

  delay (50); // Задаємо затримку

}
```

Лістинг програми Bluetooth Terminal.

```
#include "esp_bt_main.h" // Підключаємо бібліотеку
#include "esp_bt_device.h" //Підключаємо бібліотеку пристроїв

bool initBluetooth()в
{
  if (!btStart()) { // Початок
    Serial.println("Failed to initialize controller"); //Помилка ініціалізації контроллера
    return false;
  }

  if (esp_bluedroid_init() != ESP_OK) {
    Serial.println("Failed to initialize bluedroid"); // Помилка ініціалізації
    return false;
  }
  if (esp_bluedroid_enable() != ESP_OK) { //Якщо  вімкнено
    Serial.println("Failed to enable bluedroid"); // Помилка вмикання
    return false;
```

```
  }
 }

 void printDeviceAddress() { //Вводимо адресу  пристрою

  const uint8_t* point = esp_bt_dev_get_address(); //Пристрій отримав адрес

  for (int i = 0; i <6; i++) { //Для
   char str[3]; //Змінна
   sprintf(str, "%02X", (int)point[i]); //
   Serial.print(str);

   if (i <5) { //Якщо
    Serial.print(":");
   }
  }
 }

 void setup () { //Вводимо налаштування
  Serial.begin(115200); //Швидкість з якою буде передаватись у монітор

  initBluetooth(); //
  printDeviceAddress();
 }

 void loop () {}
```

Після запуска скетча откриваємо монитор порта. Побачимо адресу вида 24:0A:C4:27:07:66. Скопіюємо адресу і визначим виробника через сайт. Виробник Espressif Inc.

Дамо нове ім'я пристрою.

Функція esp_bt_dev_set_device_name () дозволяє привласнити пристрою нове ім'я.

```
#include "esp_bt_main.h" //Вмикання бібліотек
#include "esp_gap_bt_api.h"
#include "esp_bt_device.h"

bool initBluetooth(const char *deviceName) //Ім’я пристрою
{
 if (!btStart()) {
  Serial.println("Failed to initialize controller"); //Помилка ініціалізації
  return false;
 }
 if (esp_bluedroid_init() != ESP_OK) {
  Serial.println("Failed to initialize bluedroid"); //Помилка ініціалізації
  return false;
 }
 if (esp_bluedroid_enable() != ESP_OK) {
  Serial.println("Failed to enable bluedroid"); //Помилка ініціалізації
  return false;
 }
 esp_bt_dev_set_device_name(deviceName);
 esp_bt_gap_set_scan_mode
(ESP_BT_SCAN_MODE_CONNECTABLE_DISCOVERABLE);
}
```

```
void setup() { //Налаштування
  Serial.begin(115200);

  if (!initBluetooth("ESP32 BT")) { //Якщо ESP32  підключен до Bluetooth
    Serial.println("Bluetooth init failed"); //Помилка ініціалізації
  };
}

void loop () {} //Цикл
```

Запустимо приклад і відкриємо монітор порту. Додаткових повідомлень не бачимо, значить ініціалізація модуля пройшла успішно. Побачимо список доступних Bluetooth-пристроїв при відкритті вікна налаштувань в Windows або на Android-телефоні. У цьому списку бачимо пристрій під ім'ям "ESP32 BT".

1.3 Передача відео с ESP32

Завдання проекту: передача відео з ESP32 по wifi з одночасним керуванням ESP32 по Bluetooth.

У модулі ESP32 є два типи Bluetooth: один - класичний Bluetooth, а інший - BLE (Bluetooth Low Energy).

Класичний Bluetooth - це простий Bluetooth, який використовується для передачі файлів і інших даних.

Використовуємо функцію Serial Bluetooth в ESP32 для відправки команд в ESP32 і перемикання стану світлодіода на платі.

Ідея програми - ініціювати послідовне з'єднання Bluetooth за допомогою ESP32. Якщо вхідні дані рівні «1», тоді включається світлодіод, а якщо це «0», світлодіод відключається. Додаємо файл заголовка BluetoothSerial, який змушує ESP32 Bluetooth працювати як Bluetoth SSP.

Сильною стороною Bluetooth є не велика пропускна здатність, а мале електроспоживання та зручність у використанні.

Bluetooth Low Energy, скорочено BLE, являє собою енергозберігаючий варіант Bluetooth. Основне застосування BLE - передача невеликих обсягів інформації. Bluetooth завжди включений, а BLE знаходиться в сплячому режимі.

BLE підходить для додатків, яким необхідно обмінюватися невеликими обсягами інформації.

З Bluetooth Low Energy, є два типи пристроїв: сервер і клієнт. ESP32 може діяти як клієнт або як сервер.

Сервер містить дані, які клієнт може прочитати. Клієнт сканує прилеглі пристрої і, коли знаходить сервер встановлює з'єднання. Це називається двухточечной зв'язком.

ESP32-камера - маленький модуль камери з чіпом ESP32-S. Крім камери OV2640 і декількох GPIO для підключення периферійних пристроїв, він має слот для карт microSD, який може бути корисний для зберігання зображень, зроблених за допомогою камери.

Основні характеристики ESP32-CAM:

- бездротовий модуль - ESP32-S WiFi 802.11 b / g / n + модуль Bluetooth;
- зовнішнє сховище - слот для карт micro- SD ємністю до 4 ГБ;
- підтримка камер OV2640 (продається з платою) або OV7670;
- формат зображення - JPEG (тільки OV2640), BMP, відтінки сірого;
- світлодіодний спалах.
- контакти - 16 з інтерфейсами UART, SPI, I2C, PWM
- напруга живлення - 5 В;

Споживана потужність:

- без використання спалаху - 180 мА;
- при включеній спалаху - 310 мА;
- глибокий сон - 6 мА;

- модем-сон - 20 мА;

- легкий сон - 6,7 мА.

Розміри - 40,5 х 27 х 4,5 мм

Температурний діапазон:

- робочий: 20 - 85 °С;

- зберігання: -40 - 90 °С при 90% відносної вологості.

Для програмування ESP32-камери знадобляться наступні компоненти:

- ESP32-CAM з OV2640;

- TTL програматор;

- з'єднувальні дроти типу "мама-мама"

Налаштування необхідного ПО і прошивку ESP32 розділимо на кілька етапів:

1. Встановлення доповнення ESP32. Використовуємо Arduino IDE для програмування плати ESP32-CAM. Приклад коду CameraWebServer, наведено у рисунку 1.3.

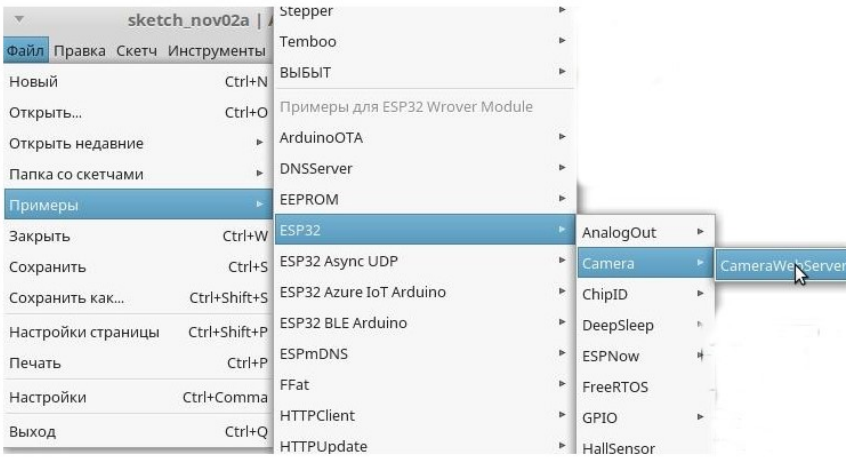

Рисунок 1.3 - Пример кода CameraWebServer

Скетч роботи с камерой ESP32 приведено у рисунку 1.4.

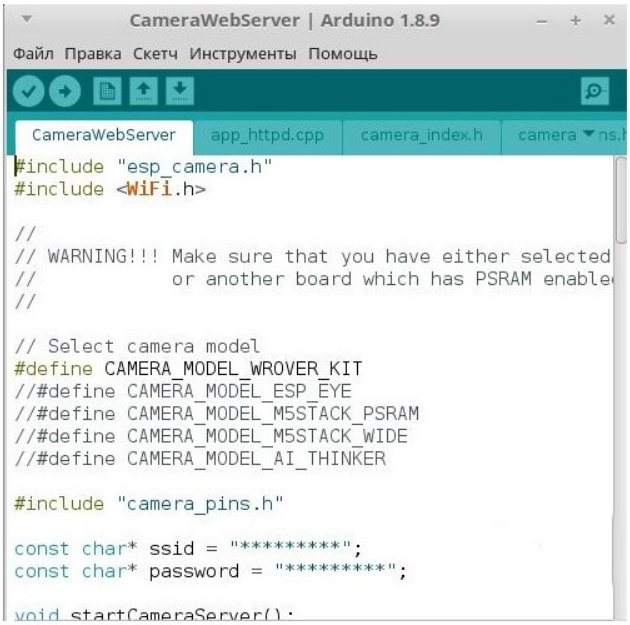

Рисунок 1.4 - Скетч роботы с камерой ESP32

2. Прошивка ESP32-CAM

Для прошивки використовуємо TTL програматор.

Підключаємо по схемі, яка наведена у рисунку1.5.

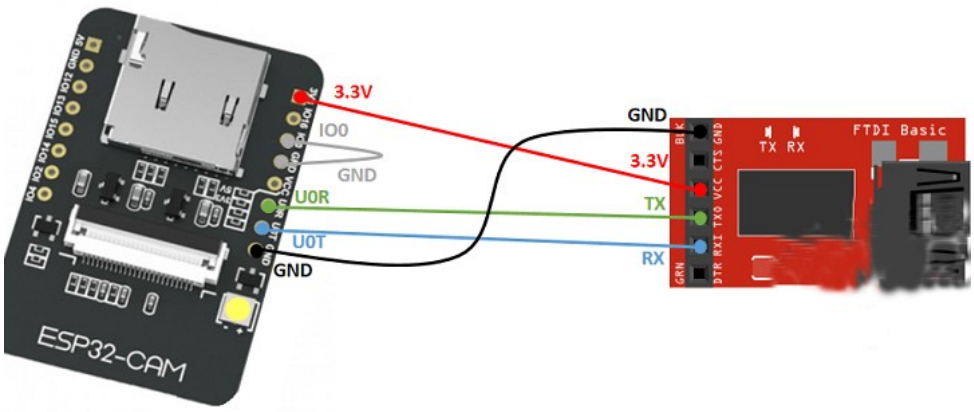

Рисунок 1.5 – Подключение ESP32

GPIO 0 підключений до GND. Щоб завантажити код виконуємо наступні дії:

- переходимо в меню «Інструменти> Плати» і вибираємо модуль ESP32 Wrover;

- переходимо в меню «Інструменти> порти», вибираємо COM-порт, до якого підключений ESP32;

- в меню «Інструменти> Partition Scheme», вибираємо "Huge APP (3MB No OTA)";

- натискаємо кнопку ESP32-CAM on-board RESET;

- натискаємо кнопку «Завантаження», щоб завантажити код, який наведено у рисунку1.6.

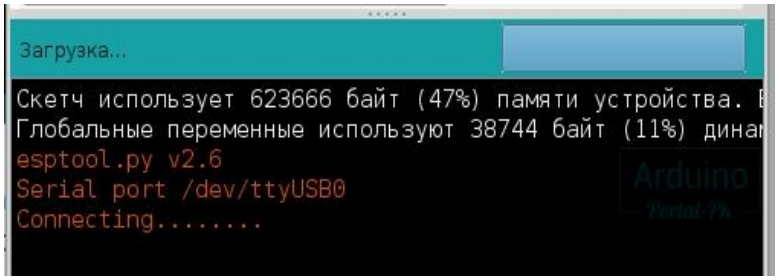

Рисунок 1.6 – Загрузка кода

3. Отримання IP-адреси і підключення до камери

Після завантаження коду відключаємо GPIO 0 від GND. Підключаємо живлення на 5 В.

Відкриваємо послідовний монітор зі швидкістю передачі даних 115200, натискаємо кнопку ESP32-CAM on-board Reset і IP-адреса ESP32 з'явиться в послідовному моніторі.

Доступ до сервера потокової передачі камери в мережі отримано. Відкриваємо браузер і вводимо IP-адреса ESP32-CAM. Натискаємо кнопку Start Streaming і починаємо потокову передачу відео. Приклад наведено у рисунку1.7.

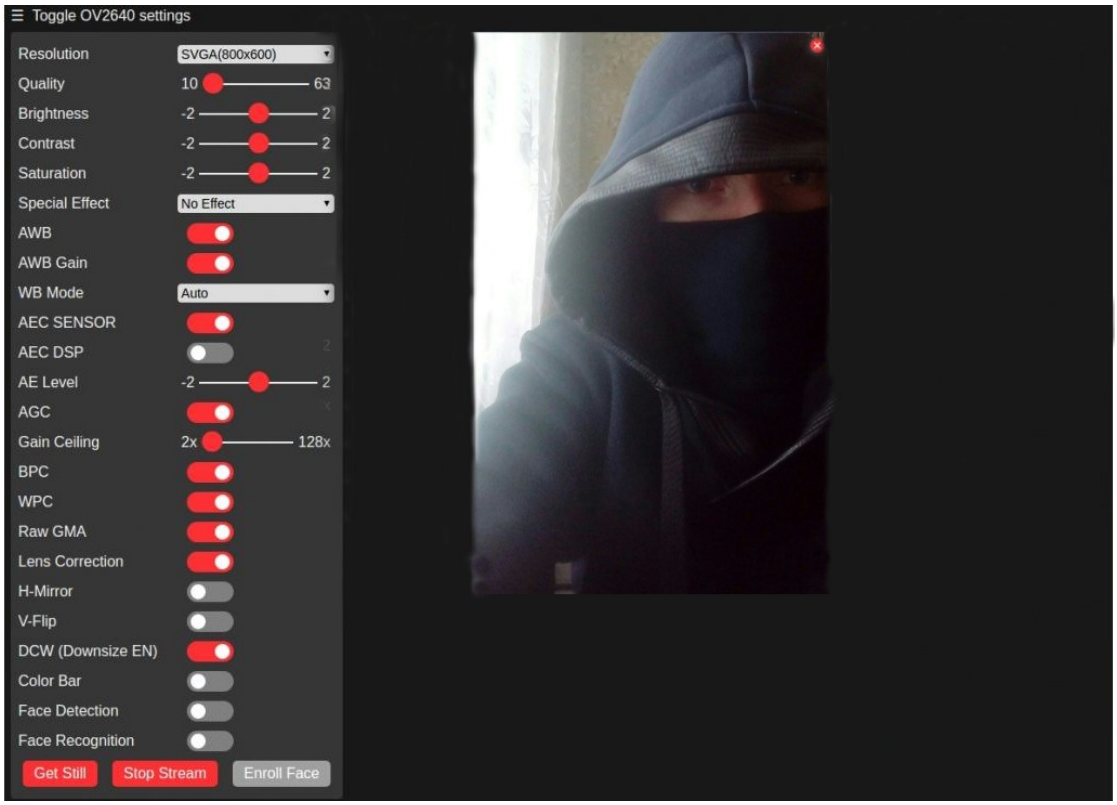

Рисунок 1.7 – Потоковое видео

2 УПРАВЛІННЯ ESP32 ЗА ДОПОМОГОЮ BLUETOOTH ТА RASPBERRY PI

2.1 Управління ESP32 по Bluetooth з передаванням відео

Підключаємо модуль ESP32 к COM порту. Перед заливкою скетча вибираємо плату, як наведено у рисунку 2.1.

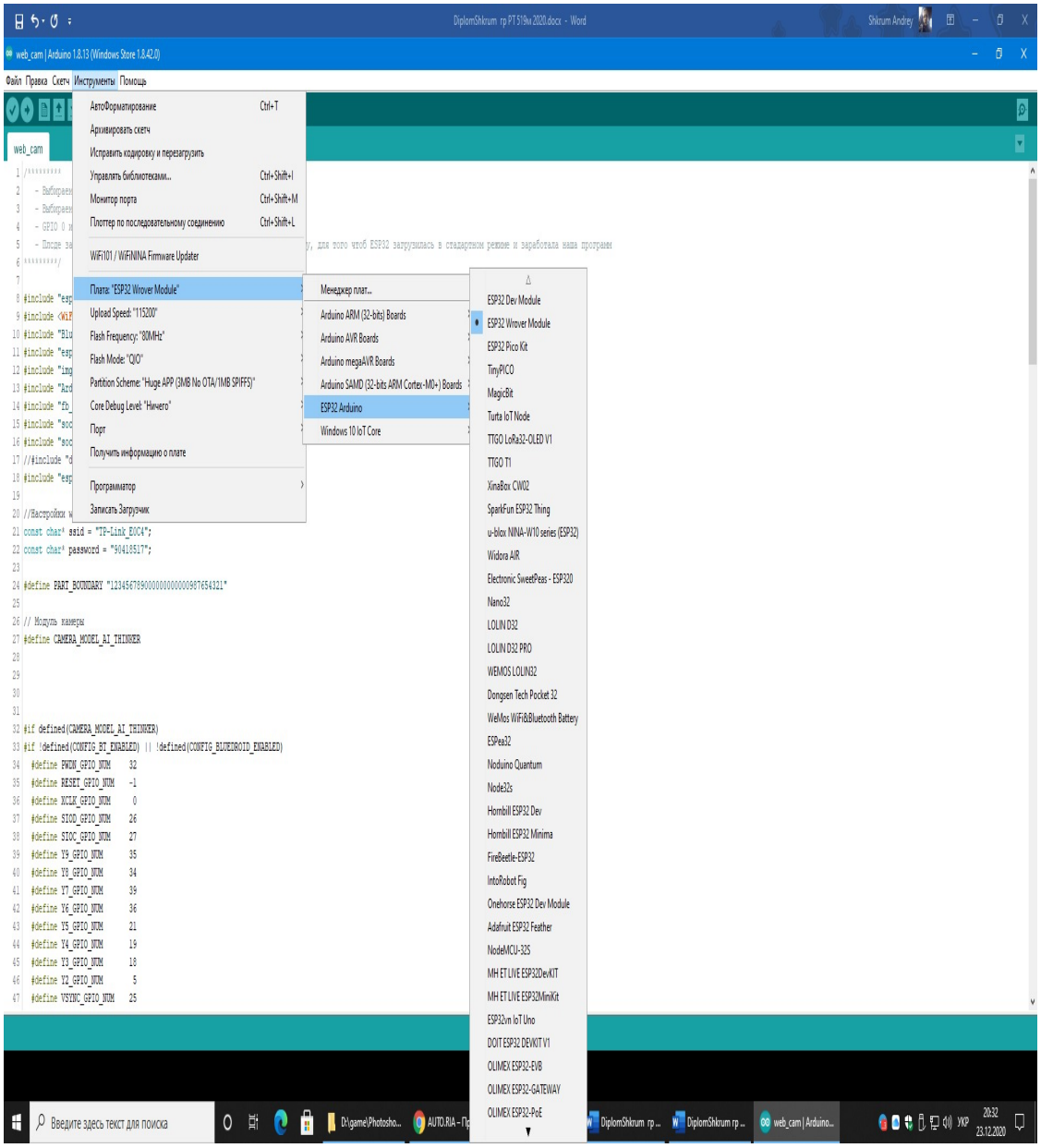

Рисунок 2.1 - Вибір плати

Далі завантажуємо код програми, якій наведено у рисунку 2.2.

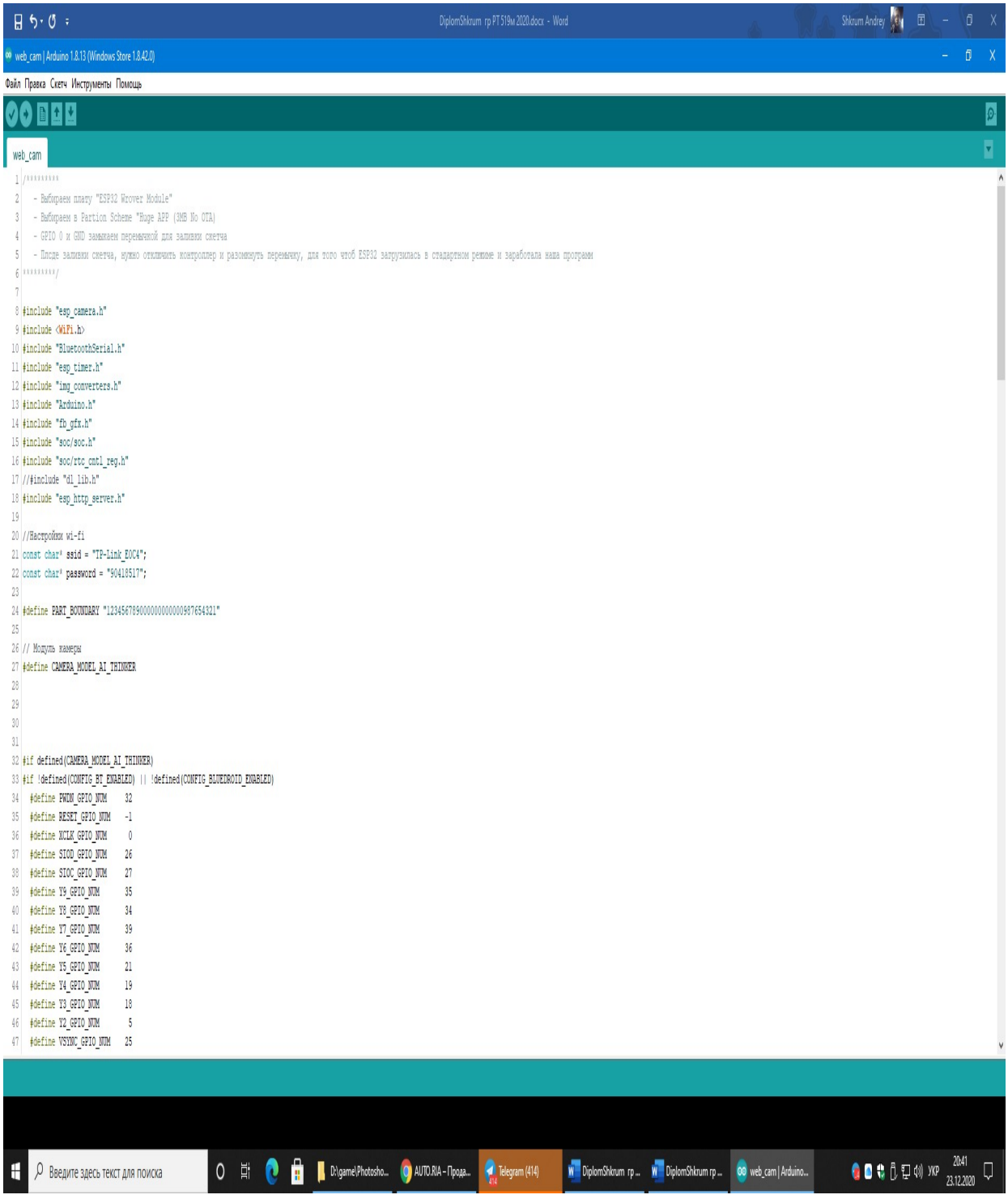

Рисунок 2.2 – Код програми

Запускаємо термінал-додаток, як наведено у рисунку 2.3.

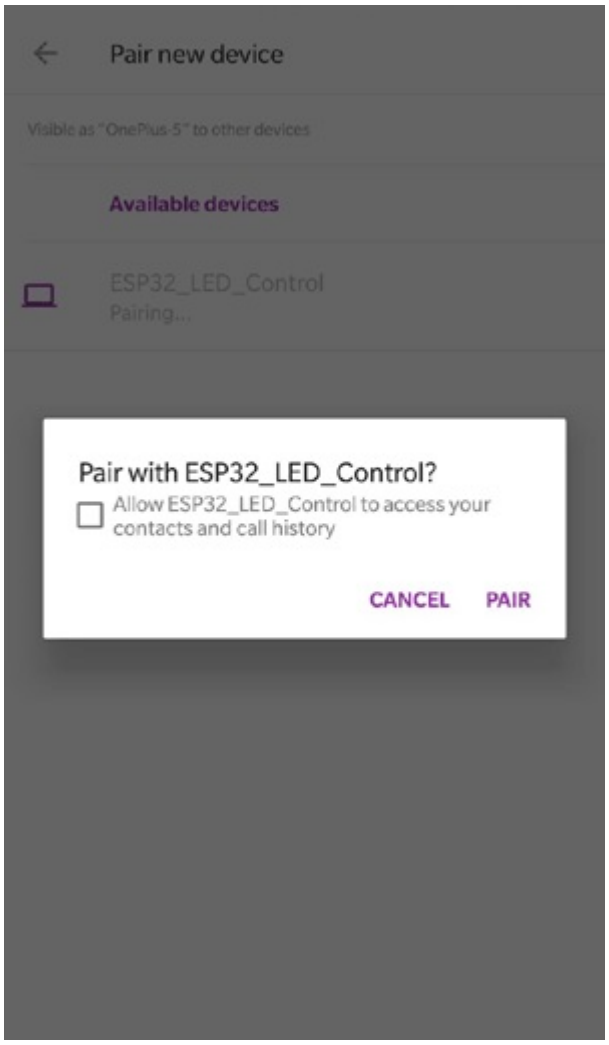

Рисунок 2.3 – Термінал-додаток

Після завантаження коду відкриваємо Serial Monitor зі швидкістю 115200 бод. Натискаємо кнопку ESP32 Enable.

Через кілька секунд отримуємо повідомлення: «Пристрій запущено, тепер ви можете підключити його до Bluetooth! ».

Зображення наведено у рисунку 2.4.

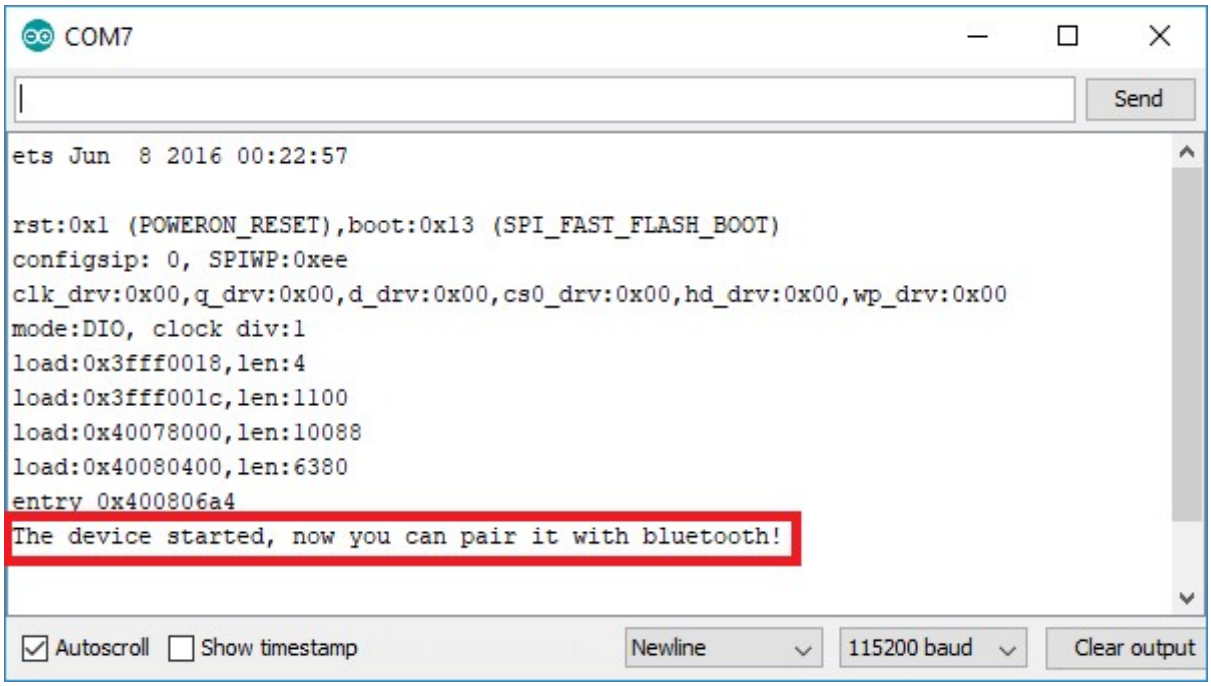

Рисунок 2.4 – Повідомлення

Відкриваємо програму «Послідовний Bluetooth-термінал».

Щоб підключитися до ESP32 в перший раз, необхідно підключити новий пристрій.

Потрібно перейти до пристроїв, як наведено у рисунку 2.5.

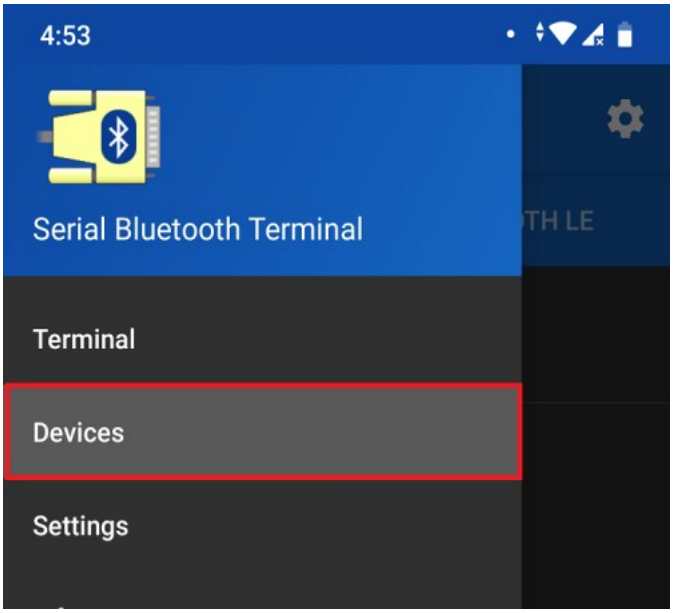

Рисунок 2.5 – Перехід до пристроїв

Натискаємо піктограму налаштувань і вибираємо «Підключити новий пристрій». Отримаємо список з доступними пристроями Bluetooth, включаючи ESP32test. Список наведено у рисунку 2.6.

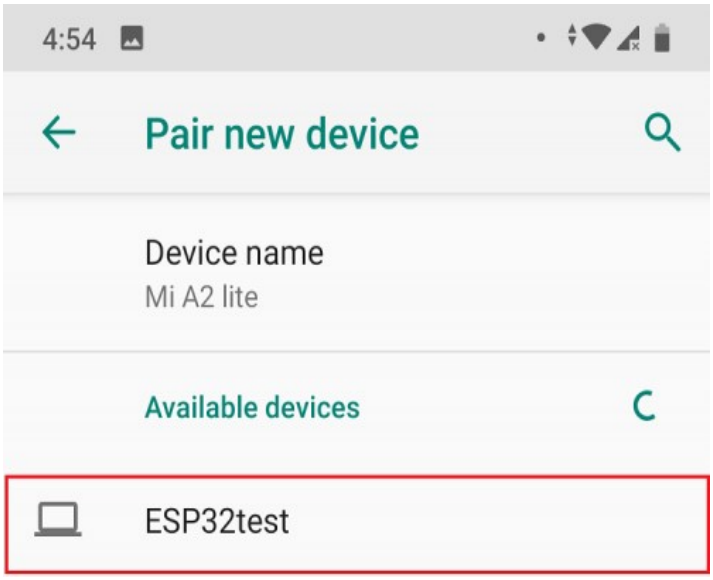

Рисунок 2.6 – Список з доступними пристроями

Повертаємось до послідовного терміналу Bluetooth. Натискаємо піктограму вгорі, щоб підключитися до ESP32. Отримаємо повідомлення «Підключено», як наведено у рисунку 2.7.

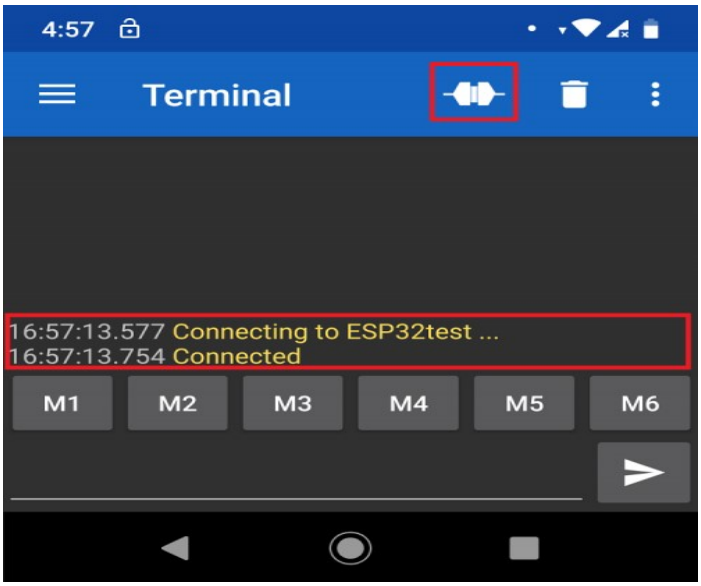

Рисунок 2.7 - Повідомлення «Підключено»

Вводимо в додатку Serial Bluetooth Terminal слово «Hello», як наведено у рисунку 2.8.

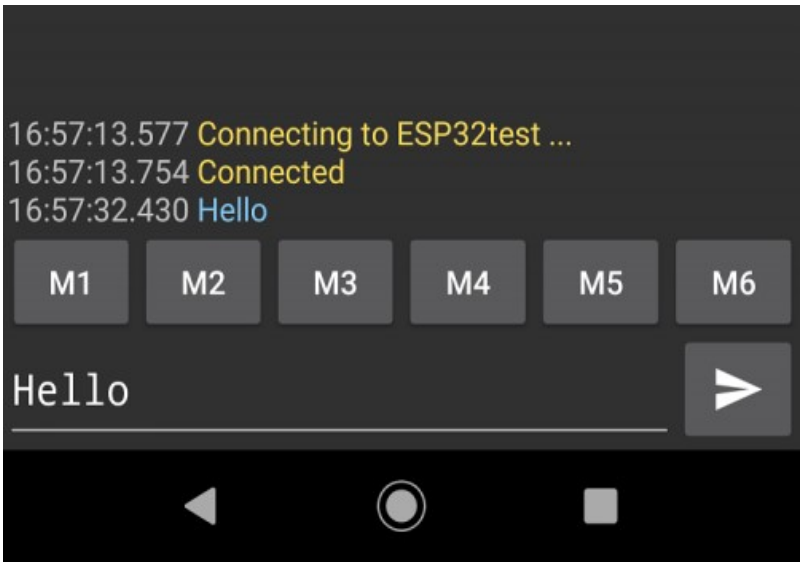

Рисунок 2.8 - Додаток Serial Bluetooth Terminal

Отримуємо повідомлення у послідовному моніторі Arduino IDE, яке наведено у рисунку 2.9.

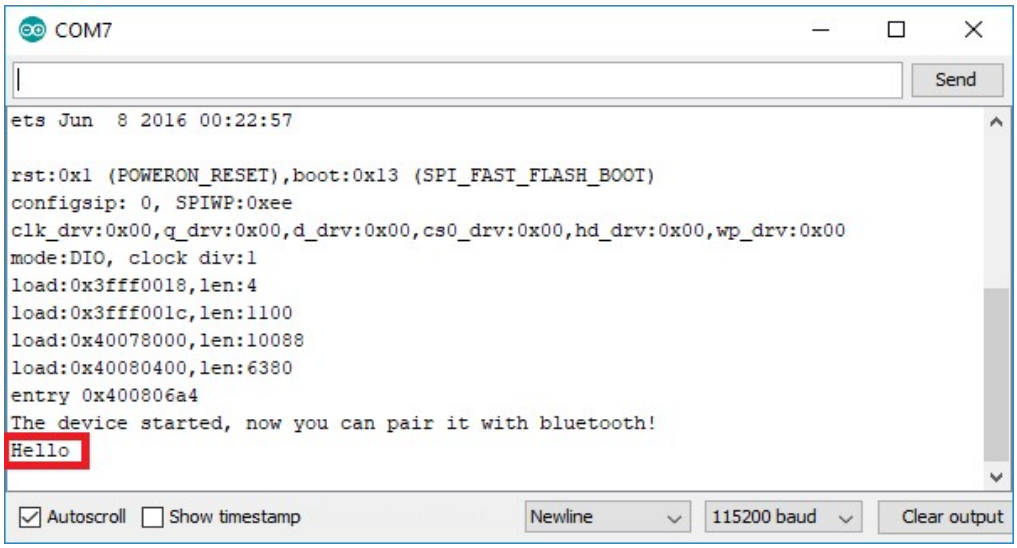

Рисунок 2.9 - Повідомлення у послідовному моніторі Arduino IDE

Далі наведено код програми підключення камери

```
// Модуль камери
#define CAMERA_MODEL_AI_THINKER

#if defined(CAMERA_MODEL_AI_THINKER) // Вибираємо модуль камери
#if!defined(CONFIG_BT_ENABLED) // Конфігурація Bluetooth ввімкнена || !defined(CONFIG_BLUEDROID_ENABLED) // Конфігурація Bluetooth ввімкнена
  #define PWDN_GPIO_NUM     32 // Встановлення затримки
  #define RESET_GPIO_NUM    -1 // Таймінг пристроїв
  #define XCLK_GPIO_NUM      0 // Таймінг пристроїв
  #define SIOD_GPIO_NUM     26 // Таймінг пристроїв
  #define SIOC_GPIO_NUM     27 // Таймінг пристроїв
  #define Y9_GPIO_NUM       35 // Таймінг пристроїв
  #define Y8_GPIO_NUM       34 // Таймінг пристроїв
  #define Y7_GPIO_NUM       39 // Таймінг пристроїв
  #define Y6_GPIO_NUM       36 // Таймінг пристроїв
  #define Y5_GPIO_NUM       21 // Таймінг пристроїв
  #define Y4_GPIO_NUM       19 // Таймінг пристроїв
  #define Y3_GPIO_NUM       18 // Таймінг пристроїв
  #define Y2_GPIO_NUM        5 // Таймінг пристроїв
  #define VSYNC_GPIO_NUM    25 // Таймінг пристроїв
  #define HREF_GPIO_NUM     23 // Таймінг пристроїв
  #define PCLK_GPIO_NUM     22 // Таймінг пристроїв
#else
  #error "Модель камери не вибрано"
  #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it :// Bluetooth не ввімкнен. Запустить меню конфігурації
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY; // Запис потоку даних
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n"; // Запис константи
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";
```

```
httpd_handle_t stream_httpd = NULL; //Обробка потока даних

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL; // Обнулення камери
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0; // Розмір зображення
  uint8_t * _jpg_buf = NULL; // Обнулення буфера обміну
  char * part_buf[64]; // Значення розміру буфера

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE); // Запис потоку контента
  if(res != ESP_OK){
    return res;
  }

// Робота камери

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed"); // Не вдалося зафіксувати камеру
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
```

Блок-схема роботи серевера відео наведена у рисунку 2.10.

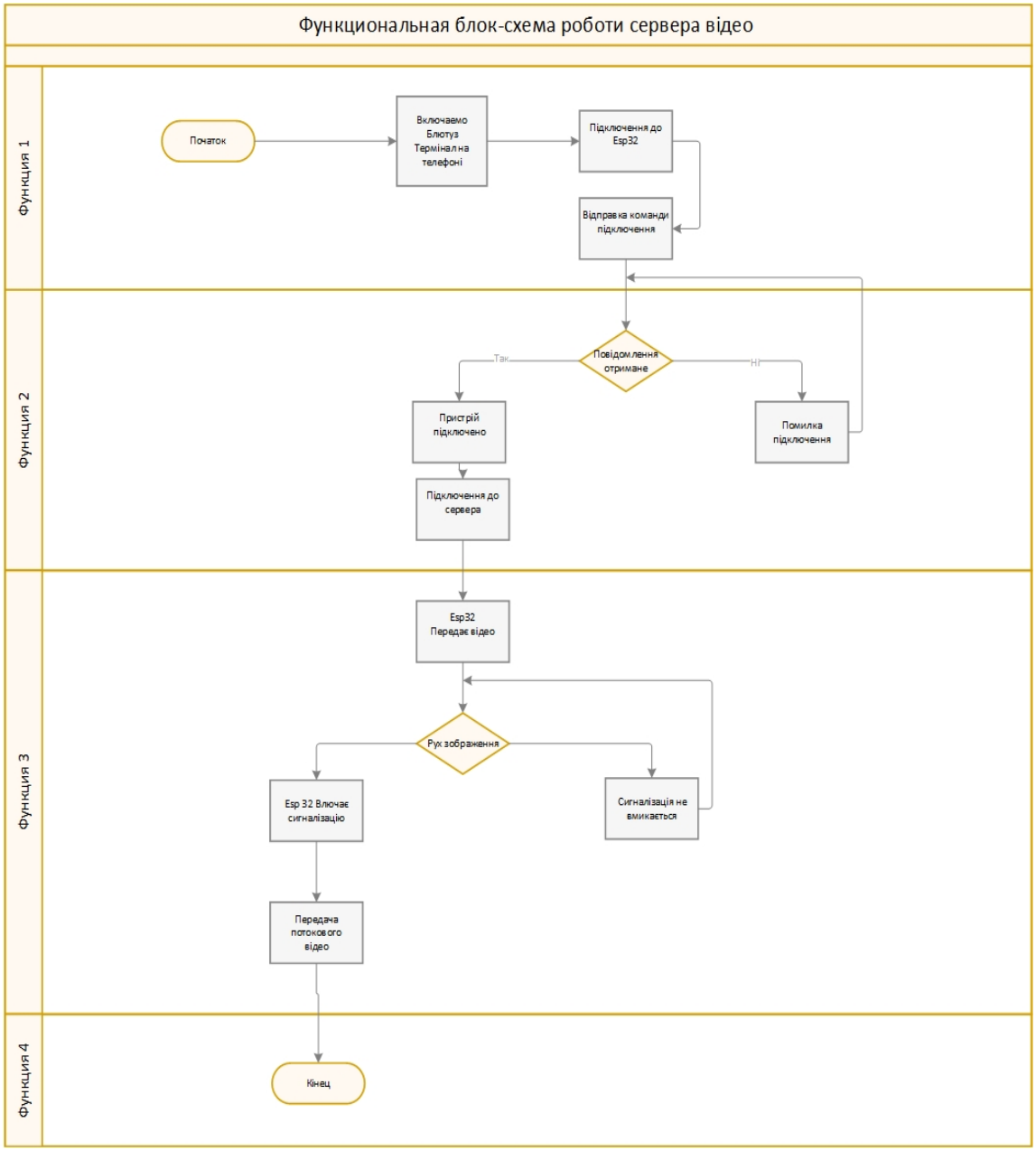

Рисунок 2.10 - Блок-схема роботи серевера відео

HttpServer також підтримує протокол WebSocket для створення постійного з'єднання між клієнтом (браузером) та ESP32. За допомогою WebSocket можливо надсилати дані в будь-якому напрямку. Примітка: ці класи використовують обробку класів C ++, що означає, що потрібно ввімкнути цю можливість за допомогою "make menuconfig".

Це можна знайти за адресою: Параметри компілятора -> Увімкнути винятки C ++ .

Щоб використовувати сервер HTTP, на найпростішому рівні створюємо екземпляр класу HttpServer і запускаємо такі процеси:

- HttpServer httpServer ();
- httpServer.start (80);
- call to start () - бере номер порту, який хочемо прослухати для вхідних запитів.

Команда start () не блокує, що означає, що сервер HTTP запускається, а потім прослуховує вхідні запити у фоновому режимі. Якщо потрібно визначити функції зворотного виклику, що викликаються при надходженні запиту, потрібно звернутися до функції addPathHandler () - екземпляр об'єкта HttpServer.

Синтаксис цієї команди: addPathHandler (метод std :: string, std :: string path, handlerFunction). Параметр методу - це метод HTTP, який обробляється. Зазвичай це GET або POST. Параметр path - це вхідний шлях, що міститься в запиті, який спричинить зворотний виклик. Звернемо увагу, що зворотний виклик буде ініційований лише тоді, коли і метод, і шлях збігаються.

Функція зворотного виклику має такий підпис: void handlerFunction (HttpRequest * pRequest, HttpResponse * pResponse).

Ось приклад функції зворотного виклику:

```
static void helloWorldHandler (HttpRequest * pRequest, HttpResponse * pResponse) {pResponse-> HTTPRESPOTS , "ГАРАЗД");
 pResponse->addHeader(HttpRequest HTTP_HEADER_CONTENT_TYPE, "text / plain");
pResponse-> sendData ("Привіт назад");
pResponse-> close_cpp ();
}
```

Тут ми визначаємо HTTP-сервер для обробки запиту:

HttpServer * pHttpServer = new HttpServer ();

pHttpServer-> addPathHandler (HttpRequest :: HTTP_METHOD_GET, "/ helloWorld", helloWorldHandler);

pHttpServer-> start (80);

реєструємо обробник за шляхом "/ helloWorld".

Таким чином, запит браузера, націлений на: http: // <ESP32_IP> / helloWorld, ініціює обробку. Об'єкт HttpRequest описує характер запиту, а об'єкт HttpResponse дозволяє надати відповідь.

Методами HttpRequest є:

• std :: рядок getBody () - отримати тіло повідомлення запиту для PUT та POST;

• std :: string getHeader (std :: string name) - отримати значення іменованого заголовка;

• std :: map <std :: рядок, std :: string> getHeaders () - отримати карту всіх заголовків та їх значень;

• std :: string getMethod () - отримати метод, який переданий у запиті;

• std :: string getPath () - отримати шлях до запиту;

• std :: map <std :: string, std :: string> getQuery () - отримати частини запиту рядка запиту;

• Socket getSocket () - отримати базовий сокет TCP / IP;

• std :: string getVersion () - отримайте версію HTTP, передану в запиті;

• WebSocket*getWebSocket () - отримати об’єкт WebSocket (припускаючи, що запитом було створення WebSocket);

• bool isWebsocket () - повернути true, якщо викликаний запит створення нового веб-сокета;

• std :: map <std :: string, std :: string> parseForm () - проаналізуємо дані , щоб повернути карту пар імен / значень форми;

• std :: vector < std :: string> pathSplit () - розбиваємо шлях на складові частини;

• std :: string urlDecode (std :: string str) - розшифровуємо рядок або тіло URL-адреси.

Методи в HttpResponse:

• void addHeader (std :: ім'я рядка, std :: значення рядка) - додайте заголовок до відповіді;

• void close () - закрийте / заповніть відповідь;

• std :: string getHeader (std :: string name) - отримати значення заголовка, який раніше додавали;

• std :: map <std :: string, std :: string> getHeaders () - отримати всі заголовки попередньо додані;

• void sendData (std :: string data) - надіслати дані клієнту;

• void setStatus (int status, std :: string message) - встановити значення стану (наприклад, 200) для відповіді.

WebSocket в обробнику зворотного виклику HTTP можемо запитати вхідний запит і чи стосується це нового створення WebSocket.

Для цього використовується виклик HttpServer # isWebSocket (). Якщо це повертає істину, то це є WebSocket. Отримавши WebSocket, не буде передано об’єкт HttpResponse, оскільки більше не обробляємо просту транзакцію запиту / відповіді.

Натомість працюємо зі стійким зв’язком між HttpServer та клієнтом. Якщо є WebSocket, обмін протоколом веб-сокета вже здійснено, і не потрібно виконувати будь-яку іншу роботу для обміну даними. При появі виклика через нову WebSocket, викликаємо функцію HttpServer # getWebSocket (), яка поверне посилання на об'єкт WebSocketobject. Об'єкт WebSocket має наступні методи:

• void close () - закрити веб-сокет;

• Socket getSocket () - отримати базовий сокет TCP / IP;

• void send (std :: string data) - надіслати дані партнеру;

• void setHandler (обробник WebSocketHandler) - встановлення обробника для подій зворотного виклику.

Оскільки вхідна передача даних WebSocket може відбуватися в будь-який час, реєструємо обробник/зворотний виклик, щоб отримувати інформацію про надходження нових даних. Реєструємо цей обробник за допомогою функції setHandler (). Це як вхідний екземпляр класу типу WebSocketHandler, який є класом з віртуальними компонентами.

Можливо надати реалізації для:

• void onClose () - викликається, коли WebSocket закритий;

• void onError (std :: string error) - викликається, коли виявляється помилка з WebSocket;

• void onMessage (WebSocketInputStreambuf * streambuf) - викликається при отриманні нового повідомлення.

За замовчуванням передбачени реалізації для функцій, які не потрібно кодувати.

Зворотний виклик onMessage () потребує невеликого пояснення. У стандартній бібліотеці C ++ маємо концепцію потоків даних.

Для нашого зворотного виклику onMessage () можемо не хотіти отримувати всі дані та зберігати їх в оперативній пам’яті. Наприклад, якщо отримати вміст файлу розміром у мегабайт від партнера, то зіткнемося із справжніми проблемами, оскільки ESP32 має лише 512 КБ оперативної пам'яті.

Є об’єкт, який реалізує інтерфейс std :: streambuf, архітектований стандартною бібліотекою C ++.

З цього об’єкта можемо створити вхідний потік std :: istream і почати витягувати дані, як потрібно. Також можемо використати форму, де можемо створити вихідний потік і передати std :: streambufinstance до цього вихідного потоку.

Результатом є те, що вихідний потік буде працювати з streambuf, щоб споживати представлений вміст і передавати його до місця призначення потоку.

Наприклад: void onMessage (WebSocketInputStreambuf * webSocketStreambuf) {// Створити вихідний потік oStream << webSocketStreambuf;}

Цей приклад не зчитує дані, передані з веб-сокета в оперативну пам'ять, перед тим як надсилати їх у вихідний потік. Натомість він буде зчитувати частину, видаляти частину і повторювати, поки не будуть витрачені всі дані потоку з веб-сокета.

WebSocket File Transfer.

Для того щоб завантажити файл у файлову систему ESP32 можливо використовувати HTTP POST. Можливо створити клас, який може отримувати вхідні файли через WebSocket. Отримавши на HTTP-сервері нового клієнта, ініційованого WebSocket, можемо пов’язати WebSocket з екземпляром класу під назвою WebSocketFileTransfer.

WebSocketFileTransfer буде обробляти вхідні файли і зберігати їх результати у файловій системі ESP32.

Наприклад:

WebSocketFileTransfer webSocketFileTransfer ("/ spiflash");

void uploadOpenHandler (HttpRequest * pRequest, HttpResponse * pResponse) {if (pRequest-> isWebSocket ()) {webSocketFileTransfer.start (pRequest-> getWebSocket ());

}}

HttpServer * pHttpServer = new HttpServer (); pHttpServer-> addPathHandler ("GET", "/ upload", uploadOpenHandler).

Розглянемо формат вхідних даних. Два або більше повідомлень будуть отримані через WebSocket. Перше повідомлення - це рядок JSON, який описує файл і буде отриман. Друге повідомлення - це вміст даних самого файлу. Це може бути декілька повідомлень, що містять частини файлу, або одне повідомлення, що містить весь файл.

Формат JSON: {"name": <fileName to be created>, "length": <length of the file in bytes> // Це необов’язково.}

Прикладом використання цієї функції може бути передача ZIP-файлу з ПК на ESP32. Наприклад, можливо написати програму Node.js, яка читає ZIP, розпаковує zip і для кожного файлу, що міститься всередині, надсилає його через WebSocket на ESP32. Коли запускаємо програму, результатом буде копія вмісту з ZIP у файловій системі ESP32.

Лістинг програми керування по Bluetooth приведено у Додатку А.

2.2 Віддалене управління за допомогою Raspberry Pi

Перші Raspberry Pi з’явилися у 2012 році, вони поклали основу мінікомп’ютерам потужніших за Arduino.

Отримати доступ до Raspberry Pi можливо у віддаленому режимі. [7]

Для отримання доступу до Raspberry Pi за межами домашньої мережі потрібно отримати IP-адресу, а також виконати налаштування параметрів на своєму маршрутизаторі.

Запускаємо Raspberry Pi та підключаємось до мережі. При підключенні Raspberry Pi до Інтернету змінюємо пароль.

Віддалене управління Raspberry Pi дозволяє підключатися до різних датчиків, двигунів, камерам, портативним літальним апаратам.

Працює цей девайс завдяки операційній системі Linux, бо ця система є більш оптимальною для пристрою такого сегменту.

Raspberry Pi має сорококонтактний роз’єм GPIO, це дає доступ до більшості периферійних пристроїв низького рівня.

Віддалене управління дає змогу не приєднувати клавіатуру, мишу, або монітор для з’єднання з робочим столом (хоча можливість підключення цих пристроїв передбачені).

Щоб отримати віддалений доступ до комп'ютера мережі та доступу до Raspberry Pi використовуємо VNC, VNC Connect. Після налаштування отримуємо доступ до графічного інтерфейсу Raspberry Pi за допомогою програми VNC Viewer.

Можливо підключитися до Raspberry Pi у Windows віддалено. [7]

Для налаштування виконуємо кілька настроек:

- налаштуємо вікна для прийняття віддалених підключень;
- знаходимо IP-адрес ПК з Windows;
- встановлюємо програмне забезпечення протоколу віддаленого робочого столу (RDP) на Raspbian;
- підключаємось до комп'ютера через Raspberry Pi.

Підключаємо Raspberry Pi до ПК у послідовності:

1) включаємо настройку віддаленого помічника:

- вибираємо властивості;
- знаходимо «Дистанційні настройки» та «Дозволити підключення»;
- натискаємо «Додати чек»;
- натискаємо «Застосовувати та підтвердити», «Перевірити Дозволити» віддалене управління комп'ютером;
- натискаємо «Добре» щоб закрити вікно властивостей, потім закриваєм вікно системи.

2) знаходимо IP-адресу пристрою Windows:

- знаходимо IP-адресу комп'ютера з Windows;
- перевіряємо список для поточного з'єднання;
- Wi-Fi вказано як адаптер бездротової локальної мережі;
- знаходимо ім'я хоста ПК через системне вікно і IP-адресу. Це буде в форматі 192.168.0.x або ж 192.168.1.x.

3) встановлюємо програмне забезпечення RDP на Raspberry Pi:

- завантажуємо комп'ютер і підключаємо його до локальної мережі, потім відкриваємо термінал і обновляємо Raspbian;

- вводимо команди по черзі, система Raspbian оновлена. Перезапускаємо Pi за допомогою «sudo reboot».

4) Встановлюємо програму для віддаленого робочого столу.

2.3 Управління по Bluetooth з Raspberry Pi web інтерфейс

Raspberry Pi відмінно взаємодіє з зовнішнім світом завдяки Bluetooth і іншим засобам обміну інформацією. Завдяки Bluetooth Raspberry Pi можна використовувати як повноцінне цифрове джерело.

Одноплатний ARM-комп'ютер Raspberry Pi - це один з лідерів у своєму сегменті. Raspberry Pi дає можливість підключати різну периферію. Це різні виконавчі пристрої, сенсори і багато інших девайсів, що працюють від електромережі. Raspberry Pi відмінно взаємодіє з зовнішнім світом завдяки великій кількості інших стандартів:

- USB і Ethernet;
- роз'єм HDMI і композитний вихід;
- Wi-Fi з Bluetooth.

На платі знаходиться швидкий мобільний процесор з відеосистемою, яка декодує і виводить на екран відео в Full HD якісті. Така платформа дозволяє реалізувати безліч вбудованих проектів. Завдяки наявності композитного виведення і HDMI, до плати можна підключити будь-який сучасний монітор, а через USB-роз'єми навіть клавіатуру з мишею. [14]

Raspberry Pi використовує звичайну micro SD флеш-карту.

Програмна частина складається із програмної оболонки (IDE) для написання програм, їх компіляції та програмування апаратури. Апаратна частина це набір змонтованих друкованих плат. Одноплатні комп'ютери не потребують установки додаткових периферійних плат.

Одноплатні комп’ютери є компактними, всі компоненти розташовуваються на одній платі.

Іноді користувачі використовують Raspberry Pi як сервер, він споживає малу кількість енергії і є безшумним. Завдяки невеликому розміру Raspberry Pi вбудовують в різні корпуси і використовуються в необхідних цілях.

Для роботи з Raspberry Pi необхідні:

- SD-карта, з якої буде завантажена операційна система;
- монітор або телевізор з роз'ємами HDMI, DVI або RCA;
- USB-клавіатура;
- USB-миша;
- кабель живлення або акумулятор micro-USB

Встановлюємо на Raspberry Pi операційну систему з сайту виробника і завантажуємо на SD карту.

Управління здійснюємо через смартфон або планшет.

Завантажуємо Raspberry Pi і підключаємо Bluetooth USB модуль.

У Raspbian модуль працює та додаткових драйверів не потрібно встановлювати.

Встановлюємо необхідно ПО:

//sudo apt-get install bluetooth bluez-utils bluez-compat

Отримуємо список доступних Bluetooth пристроїв, до яких може підключитися Raspberry Pi, смартфон робимо видимим для інших BT пристроїв у настройках:

//hcitool scan

У відповідь на цю команду з’явиться список доступних для сполучення пристроїв. В списку адресів вибираємо необхідний.

У відповідь в консолі Raspberry Pi повинен з’явитися запит підтвердження, одночасно на екрані смартфона з’являється запит на дозвіл сполучення:

//RequestConfirmation (/org/bluez/2184/hci0/dev_20_F3_A3_E2_D7_49, 100111) Confirm passkey (yes/no):

Вибираємо “yes” і натискаємо кнопку “Сполучення” для цього повідомлення на смартфоні. Потім скрипт bluez-simple-agent повинен вивести напис:

//Release New device (/org/bluez/2184/hci0/dev_20_F3_A3_E2_D7_49)

Лістинг програми web сервера з RASPBERRY PI приведено у Додатку Б

2.4 Розробка інтерфейса з Raspberry Pi

При створенні веб-інтерфейсу використовуються такі технології:

- HTML;
- CSS;
- PHP;

Мова розмітки HTML - "основа" веб-сторінки. HTML-тегами задаються параграфи та гіперпосилання.

CSS - каскадні таблиці стилів. За допомогою таблиць стилів CSS задаємо шрифти і їх розмір, вертикальні і горизонтальні меню.

Веб-інтерфейс представляє доступ до всіх функцій сервера на Raspberry Pi у зручному вигляді, та відображається чітко на мобільних пристроях.

Забезпечуємо адаптивну верстку сторінки.

Створюємо веб-інтерфейс на HTM, вміст веб-сторінки знаходиться між тегами <html> (відкриває тег) і </ html> (закриває тег).

У середині шапка сторінки і тіло сторінки. У шапці сторінки міститься службова інформація. А тіло сторінки бачимо на екрані у браузері.

В папці / var / www / html / на Raspberry Pi створюємо (або відкриваємо для редагування) файл index.html:

sudo nano /var/www/html/index.html

Поставимо «відкриває тег» <html> і запишемо шапку сайту:

<Html>

<Head>

```
<Meta http-equiv = "Content-Type" content = "text / html; charset = utf-8" />
<Title> Raspberry Pi </ title>
</ Head>
```

Створюємо тіло сторінки:

```
<Body>
<H1 align = "center"> Домашній сервер на Raspberry Pi 3 </ h1>
<Nav role = 'navigation'>
  <a href="ССИЛКА1"> Медіасервер Plex </a>
  <a href="ССИЛКА2"> Електронна бібліотека </a>
  <a href="ССИЛКА3"> RPi-Monitor </a>
  <a href="ССИЛКА4"> Transmission </a>
  <a href="ССИЛКА5"> Nextcloud </a>

</ Nav>

<Ul id = "navbar">
     <Li> <a href="ССИЛКА6"> [Status] </a> </ li>
     <Li> <a href="ССИЛКА7"> [Reboot] </a> </ li>
     <Li> <a href="ССИЛКА8"> [Transmission reload] </a> </ li>
     <Li> <a href="ССИЛКА9"> [CUPS] </a> </ li>
   </ Ul>

</ Body>
</ Html>
```

Заголовок знаходиться зверху – «Raspberry Pi 3». Він укладений в теги заголовка першого рівня H1 і знаходиться по центру сторінки.

Розглянемо два меню. Перше з них розташовано вертикально, а друге горизонтально. Обидва меню «відкривають» і «закривають» тегі. В кінці

«закриває» тег </ html>, усередині якого знаходяться і шапка сторінки (head) і тіло сторінки (body). У цьому вся суть HTML-розмітки.

Основу сторінки створено. Включаємо каталог електронної бібліотеки за допомогою вертикального меню.

За вертикальним меню йде меню горизонтальне, яке відображається одним рядком.

У горизонтальному меню знаходяться кнопки виведення короткого статусу системи, перезавантаження, перезапуску Transmission, доступу до панелі управління і управління CUPS.

Стандартними засобами HTML зробити звернення до порта неможливо. Тому задіємо яваскрипти, які підставляють номер порта безпосередньо в момент кліка.

Вертикальне меню після прописування посилань:

 Медіасервер Plex

 Електронна бібліотека

 RPi-Monitor

 Transmission

 Nextcloud

Прописані посилання на пункти додаткового горизонтального меню:

<Li> [Status] </ li>

<Li> [Reboot] </ li>

<Li> [Transmission reload] </ li>

<Li> [CUPS] </ li>

Вносимо поправки в тіло сторінки index.html, і процес створення основи веб-інтерфейсу закінчен.

Додамо інтерфейсу іконку (favicon), яка буде відображатися в закладках і в заголовку вкладки в браузері.

Для створення іконки знаходимо підходящу картинку і користуємось одним з численних онлайн-сервісів по генерації favicon.

Створення іконки робимо за допомогою realfavicongenerator.net.

Можна поправити стандартні параметри, після чого натискаємо на "Generate your favicons and HTML code" і сервіс видає архів з файлами іконок і HTML-код для вставки на сторінку.

Код буде приблизно таким:

<Link rel = "apple-touch-icon" sizes = "180x180" href = "/ apple-touch-icon.png">

<Link rel = "icon" type = "image / png" sizes = "32x32" href = "/ favicon-32x32.png">

<Link rel = "icon" type = "image / png" sizes = "16x16" href = "/ favicon-16x16.png">

<link rel="manifest" href="/manifest.json">

<link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">

<meta name="theme-color" content="#ffffff">

Вставляємо його в шапку файла index.html, а завантажений з сервісу архів з іконками розпаковуємо в папку / var / www / html /.

Робимо кнопку перезавантаження на PHP.

Зробимо кнопку, натискання якої в веб-інтерфейсі Raspberry Pi дасть команду на перезавантаження.

Створюємо php-файл:

sudo nano /var/www/html/com-reboot.php

Вписуємо в нього наступне:

<? Php

echo shell_exec ("sudo reboot");

?>

Створюємо php-файл, який виводить короткий статус:

sudo nano /var/www/html/status.php

І впишемо в нього код:

```
<? Php
   echo "<b> Uptime: </ b>. shell_exec ("uptime"). "<br />";
   echo "<b> System Info: </ b>. shell_exec ("uname -a"). "<br />";
?>
```

Виводиться на екран текст "Uptime:", який виконує консольную команду "uptime" і виводить на екран відповідь.

Створюємо файл:

```
sudo nano /var/www/html/com-transreload.php
```

Вписуємо в нього:

```
<? Php
echo shell_exec ("sudo /etc/init.d/transmission-daemon stop");
echo shell_exec ("sudo /etc/init.d/transmission-daemon start");
?>
```

Всі дії, пов'язані з роботою веб-сервера в Raspbian виконуються від користувача www-data.

Скористаємося іншим способом: дамо користувачеві www-data доступ до виконання суворо обумовленого списку команд з правами суперкористувача без запиту пароля користувача root.

Для цього відредагуємо файл настройок:

```
sudo nano / etc / sudoers
```

І впишемо в нього наступні рядки:

```
% Www-data ALL = NOPASSWD: / sbin / reboot
% Www-data ALL = NOPASSWD: /etc/init.d/transmission-daemon stop
% Www-data ALL = NOPASSWD: /etc/init.d/transmission-daemon start
```

На сервері зможуть виконуватися команди "sudo reboot", "sudo /etc/init.d/transmission-daemon stop" і "sudo /etc/init.d/transmission-daemon start".

Якщо надалі розширити функціональність веб-інтерфейсу управління, то треба прописувати права на виконання кожної з цих команд для www-data в файл / etc / sudoers.

Надаємо веб-інтерфейсу остаточний вигляд через CSS-стиль.

Надаємо інтерфейсу остаточний вигляд.

Робимо за допомогою CSS-стилів.

Спочатку робимо так, щоб всі тексти в інтерфейсі відображалися красивим тонким шрифтом.

За еталон вибираємо шрифт, який використовується корпорацією Apple в своєму дизайні. Він називається San Francisco і недоступний для вільного користування.

Але є і загальнодоступні шрифти. Виберемо безкоштовний шрифт Roboto.

Підключимо його до веб-інтерфейсу. В шапці файлу index.html у будь - якому місці після відкриваючого тега <head> вставляємо наступний рядок:

<link href="https://fonts.googleapis.com/css?family=Roboto:100,300" rel="stylesheet">

Тепер шрифт можна використовувати на сторінці.

Тепер сформуємо таблицю стилів CSS.

<style type="text/css"> //Стиль тексту

```
body{
background:#f1f1f1;
}

header, nav {
 text-align: center; //Текст по центру
 margin: 20px 0;
 padding: 10px;
```

```
}

nav a {
 font-family:'Roboto', Helvetica Neue,helvetica,arial,verdana,sans-serif;
 font-weight:200;
 font-size: 2.4em; //Розмір
 text-decoration: none; //Декорації тексту: нема
 background: #333;
 border-radius: 5px;
 color: white; //Колір:білий
 padding: 3px 8px;
 display: block;
}

p
font-family:'Roboto', Helvetica Neue,helvetica,arial,verdana,sans-serif;
font-weight:200;
font-size: 2.8em;
line-height: 1.333em; /* 16px/12 =1.33em*/
margin-bottom: 1.25em; /* 15px/12 =1.25em*/
}

h1{
font-family:'Roboto', Helvetica Neue,helvetica,arial,verdana,sans-serif;
margin:0px auto 0 auto;
font-size: 2.8em;
text-align:center;
font-weight:200;
color:#4b4b4b;
```

```
}

#navbar {
 font-family:'Roboto', Helvetica Neue,helvetica,arial,verdana,sans-serif;
 font-weight:200;
 font-size: 2.0em;
 text-align: center;
 text-decoration: none;
 background: #333;
 border-radius: 5px;
 color: white;
 padding: 3px 8px;
 }

#navbar li {display: inline;}

#navbar a {
 color: #fff;
 padding: 3px 8px;
 text-decoration: none;
 display: inline;
 width: 100px;
}

</style>
```

Код потрібно вставити в шапку файлу index.html, між тегами <head> і </head>.

Він прописує колір фону, вказує що тексти слід відображати підключеним шрифтом Roboto, визначає розміри і форму меню.

Веб-інтерфейс виглядає в браузері комп'ютера, як наведено у рисунку 2.11. Він доступен до введення у адресний рядок ip-адреси Raspberry Pi.

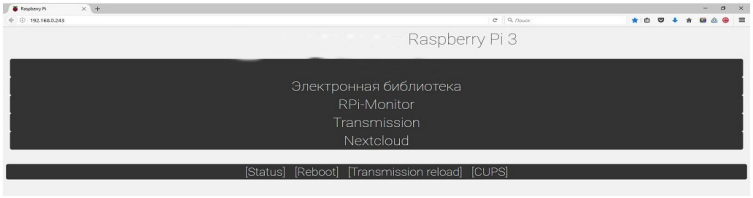

Рисунок 2.11 - Веб-інтерфейс в браузері комп'ютера

На екрані айфона будет зображення, як у рисунку 2.12.

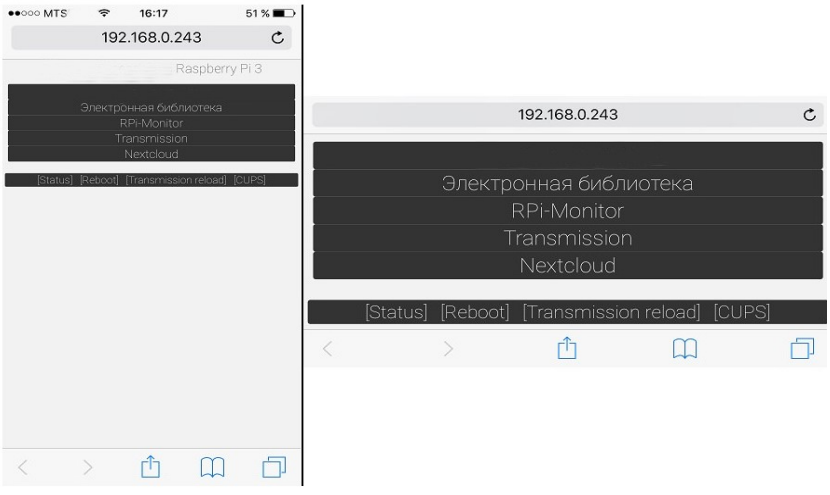

Рисунок 2.12 – Зображення на екрані айфона

Додаємо іконку для доступа до веб-інтерфейсу прямо на робочий стіл смартфона.

Додаємо ярлик для доступа до створеного інтерфейсу Raspberry Pi на екран смартфона або планшета.

Приклад OS і браузера Safari виглядає як наведено у рисунку 2.13.

Рисунок 2.13 - Додавання ярлика для доступа на екран смартфона

Відкриваємо сторінку в браузері. Натискаємо на іконку "Поділитися" в нижньому меню. Вибираємо пункт "На екран додому" на робочому столі. Потім тиснемо "Додати" і отримуємо результат.

Ярлик поміщаємо на робочий стіл у разі використання Android і Google Chrome.

2.5 Опис кода програми Bluetooth Server

Bluetooth BLE

Використання низькорівневих функцій ESP-IDF для побудови рішення BLE вважається складним. Існує багато концепцій, які слід розглянути, а також вхідні події, які потрібно обробляти. Це здається ідеальним для інкапсуляції класу вищого рівня для спрощення завдань. З цією метою створюємо набір класів C ++, які забезпечують всі необхідні можливості BLE з більш простою моделлю використання.

При використанні цих класів використовуємо make menuconfig для налаштування середовища ESP-IDF. По-перше, потрібно переконатися, що

Bluetooth увімкнено. Починаємо робити menuconfig і перейходимо до Componentconfig і вибераємо Bluetooth.

Далі детально ознайомлюємось із записом Bluetooth і переконуємось, що вибрано «Стек Bluetooth Bluedroid увімкнено»

Збережимо всі зміни, відновлені у вихідному коді. Маємо дві частини: одна - це BLE-сервер, а інша - BLE-клієнт. BLE-сервер також відомий як BLE-периферія. За допомогою сервера BLE можливо використовувати одну або декілька служб, де кожна служба має одну або кілька характеристик, і кожна характеристика може мати нуль або більше дескрипторів.

У моделі C ++ створюємо концепцію класів, що представляють ці елементи:

• BLEServer - моделює сервер;

• BLEService - моделює послугу.

Належить серверу BLES:

• BLEC характеристика - моделює характеристику.

Належить службі BLES:

• BLEDescriptor - моделює дескриптор, володіє BLEC характеристикою.

А також:

• BLEAdvertising – моделює ідентіфікацію.

На високому рівні BLE стає:

// Ініціалізація середовища BLEBLEDevice :: init ("Ім'я сервера");

// Створюємо серверBLEServer * pServer = BLEDevice :: createServer ();

// Створюємо службуBLEService * pService = pServer-> createService (ServiceUUID);

// Створюємо характеристикуBLECharacteristic * pCharacteristic = pService-> createCharacteristic (CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, CharacteristicUUID,

CharacteristicUUID, CharacteristicUUID, CharacteristicUUID, / Встановіть характеристику valuepCharacteristic-> setValue ("Hello world");

// Запустіть службу Service-> start ().

Далі створюємо сервер, створюємо службу, створюємо характеристику на сервісу, встановлюємо значення для характеристики.

Далі наведено частину коду роботи сервера.

```
/*
 * BLEService.h
 *
 *  Created on: Mar 25, 2020
 *      Author: Shkrum
 */

#ifndef COMPONENTS_CPP_UTILS_BLESERVICE_H_
#define COMPONENTS_CPP_UTILS_BLESERVICE_H_
#include "sdkconfig.h"
#if defined(CONFIG_BT_ENABLED)

#include <esp_gatts_api.h>

#include "BLECharacteristic.h"
#include "BLEServer.h"
#include "BLEUUID.h"
#include "FreeRTOS.h"

class BLEServer;

/**
 * @brief A data mapping used to manage the set of %BLE characteristics kn
own to the server.
 */
class BLECharacteristicMap {
public:
   void setByUUID(BLECharacteristic* pCharacteristic, const char* uuid);
   void setByUUID(BLECharacteristic* pCharacteristic, BLEUUID uuid);
   void setByHandle(uint16_t handle, BLECharacteristic* pCharacteristic);
   BLECharacteristic* getByUUID(const char* uuid);
```

```
    BLECharacteristic* getByUUID(BLEUUID uuid);
    BLECharacteristic* getByHandle(uint16_t handle);
    BLECharacteristic* getFirst();
    BLECharacteristic* getNext();
    std::string toString();
    void handleGATTServerEvent(esp_gatts_cb_event_t event, esp_gatt_if_t g
atts_if, esp_ble_gatts_cb_param_t* param);
```

Блок-схема роботи сервера Bluetooth наведена у рисунку 2.14.

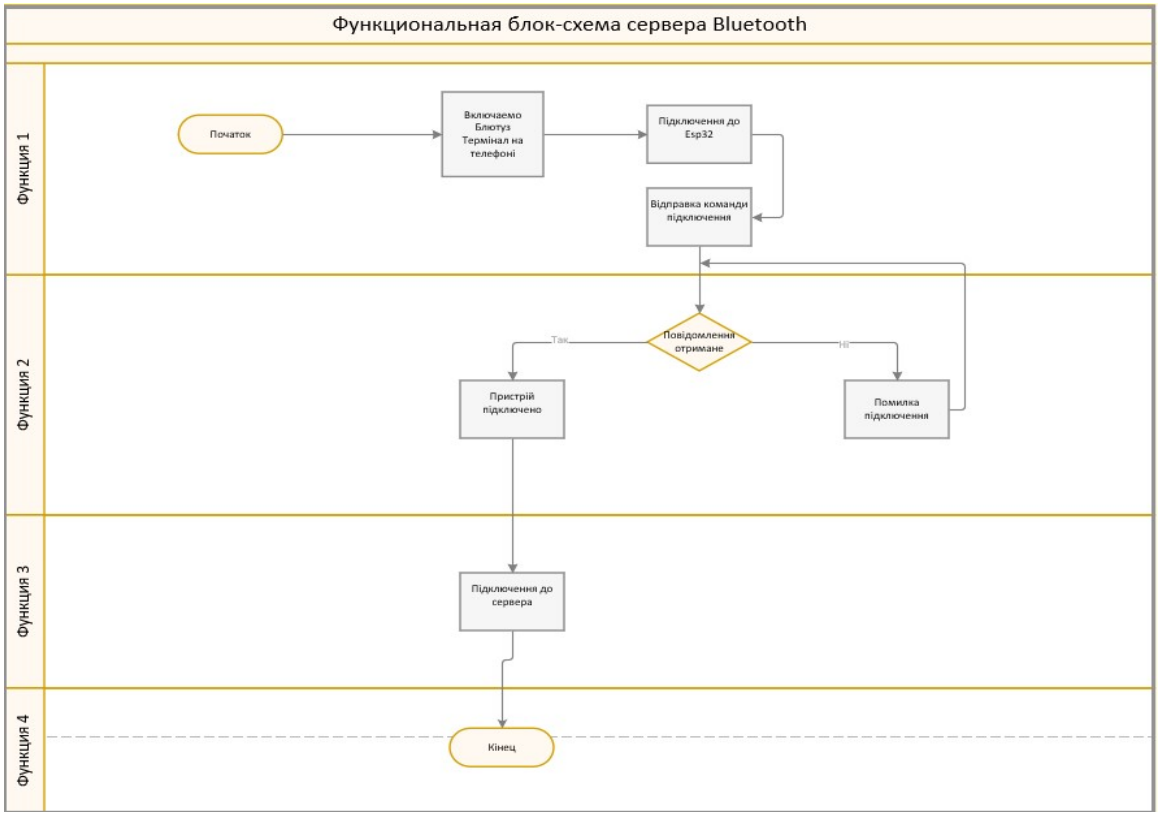

Рисунок 2.14 - Блок-схема роботи сервера Bluetooth

ESP32 BLE API, що постачаються ESP-IDF повинна мати код, необхідний для обробки подій. Це обробляється шляхом реалізації класів. Метою цих класів є ефективна обробка робочого процесу BLE.

Класи містять прості API високого рівня для 95% загальноприйнятих практик, а також забезпечують методи, які можна викликати для налаштування та адаптації операцій для більш рідкісних випадків.

Коли BLE-сервер запущено, пристрій однорангової мережі може зчитувати значення характеристики або встановлювати нове значення.

Після того, як запущено BLE Server, можемо попросити у нього об’єкт (BLEAdvertising): BLEAdvertising * pAdvertising = pServer-> getAdvertising (); pAdvertising-> start ().

Після виконання клієнт може динамічно знаходити сервер.

Клієнт може запросити з'єднання у будь-який час, якщо потрібно повідомити (або запам'ятати) адресу сервера BLE.

Можемо створити власний корисний набір функцій та пов’язати з тим, що пропонує ESP32. Це досягається за допомогою класу типу BLEAdvertisementData. Створюємо екземпляр цього класу, заповнюємо його корисним навантаженням :

BLEAdvertising * pAdvertising = pServer-> getAdvertising ();

BLEAdvertisementData advertisementData;

// встановлюємо властивості функції data

// перегляньте методи встановлення класу BLEAdvertisementData pAdvertising-> setAdvertisementData (advertisementData); pAdvertising-> start ().

Значення, яке зчитане з характеристики, забезпечує інформацію. Якщо характеристика відображає пульс, виміряний датчиком, то віддалений клієнт отримує значення пульсу, яке зчитано з характеристики. Значення буде внутрішньо змінюватися сервером або кожного разу, коли клієнт робить запит на зчитування, або приймається нове зчитування датчика.

Характеристика, що зберігається на сервері BLE, може представляти стан чогось, що можна активувати або змінити. Коли характеристика відображає стан замка дверей автомобіля, можливо прочитати характеристику, яка визначить, що двері заблоковані, коли виходимо, або, навпаки, встановимо значення характеристики в розблокований стан, що призведе до того, що сервер механічно розблокує двері.

Для моделювання цих понять у класах C ++ використовуємо уявлення про те, що клас C ++ можна підкласифікувати як більш спеціалізований. Клас під назвою BLECharacteristicCallbacks надає два методи, якими можна перезавантажити:

• onRead (BLECharacteristic * pCharacteristic) - викликається, коли запит на читання надходить від клієнта;

• onWrite (BLECharacteristic * pCharacteristic) - викликається, коли надходить запит на запис, ініційований клієнтом.

Нове значення вже було встановлено в характеристиці, і власна логіка коду може читати значення та виконувати певну дію. Щоб використовувати це, можемо замінити функції у власному класі C ++, який підкласує передбачений клас MyCallback: public BLECharateristicCallback

{void onRead (BLECharacteristic * pCharacteristic) {// Зробіть щось до завершення читання.

}

void onWrite (BLECharacteristic * pCharacteristic) {// Зробіть щось, тому що було написано нове значення.

}}

Щоб використати цю техніку, повідомляємо BLECharacteristic про обробку зворотного дзвінка.

Наприклад: pCharacteristic-> setCallbacks (new MyCallback ()).

Ось приклад, який передає час від запуску кожного разу, коли клієнт запитує значення: class MyCallbackHandler: public BLECharacteristicCallbacks

{void onRead (BLECharacteristic * pCharacteristic) {struct timeval tv; gettimeofday (& tv, nullptr);

std :: ostringstream os; os << "Час:" << tv.tv_sec; pCharacteristic-> setValue (os.str ());

}};

Подібно до характерних зворотних викликів, також є зворотні виклики сервера. Вони повідомляють код про події підключення та відключення клієнта. Вони підкласифікуються з класу BLEServerCallbacksclass, який має віртуальні методи:

• onConnect (BLEServer * pServer) - викликається при встановленні з'єднання;

• onDisconnect (BLEServer * pServer) - викликається при відключенні. показання датчика.

Сервер може отримувати запити на читання для отримання поточного значення або може отримувати запити на запис для встановлення поточного значення. Є ще одна операція, яка викликається сервером на стороні сервера та асинхронно для клієнта.

Ця операція називається "повідомити". Вона використовується для сигналізації (або повідомлення) клієнту про те, що значення характеристики змінилося.

Клієнт отримає індикаційну подію, що повідомить, що зміни відбулися. На сервері можемо показати вказівку, що має відбутися, викликаючи: pCharacteristic-> notify ().

Поточним значенням характеристики буде значення, яке передається рівному. Можемо з'єднати цей виклик із попереднім запитом для встановлення нового значення:

pCharacteristic-> setValue ("HighTemp");

pCharacteristic-> notify ().

Подібна функція notify () називається indic (). Різниця між ними полягає в тому, що вказівник () отримує підтвердження, в той час як повідомлення () не отримує підтвердження. Поєднаний з ідеєю вказівок/сповіщень, є архітектурним BLE-дескриптором під назвою "Клієнтська характеристична конфігурація", який має UUID 0x2902.

Він містить два окремі бітові поля, які можна вмикати та вимикати. Одне бітове поле регулює сповіщення, а інше - індикації. Якщо відповідний біт увімкнено, тоді сервер може / не може надіслати відповідний push.

Якщо біт Notifications увімкнено, сервер може / не може надсилати сповіщення. Основна мета дескриптора - дозволити партнеру (клієнту) вимагати, щоб сервер фактично надсилав сповіщення або вказівки.

Наведемо приклад: є BLE-сервер, який може публікувати сповіщення при зміні даних. Припустимо що клієнт BLE підключається, але насправді не зацікавлений у їх отриманні. Якщо сервер BLE виконує сповіщення та надсилає дані, це витратить зусилля / енергію, оскільки клієнт не хоче або не може використовувати інформацію.

Потрібно щоб клієнт повідомив сервер, що він хоче надсилати нові дані. І саме тут дескриптор вступає в гру.

Якщо дескриптор присутній у характеристиці, тоді клієнт може віддалено змінити прапорець бітів, щоб дозволити або вимкнути сповіщення та індикації. Оскільки прапор дескриптора зберігається на сервері, сервер може вільно перевірити прапор перед виконанням радіопередачі. Це означає, що клієнт може вмикати або вимикати своє бажання отримувати сповіщення, і сервер повинен виконувати ці запити.

Специфікація BLE обмежує максимальний обсяг даних, який можна надіслати через повідомлення або індикацію, до 20 байт або менше. Якщо значення характеристики перевищує цю суму, тоді буде передано лише перші 20 байт даних. Ймовірно, що сервер BLE-програми буде мати власний набір характеристик.

За допомогою характеристики можемо встановити і отримати значення як двійковий фрагмент сховища, однак це може бути низьким рівнем. Альтернативою є використання можливостей мови C ++ та моделювання власної спеціалізованої характеристики як підкласу характеристик BLEC.

Потім це може інкапсулювати ґеттер і сеттер нижчого рівня із власною індивідуальною версією.

Наприклад, якщо ваша характеристика представляла температуру, можливо створити: class MyTemperatureCharacteristic:

public BLECharacteristic

{MyTemperature (): BleCharacteristic (BLEUUID (MYUUID)) {setTemperature (0,0);

}

void setTemperature (double temp) {setValue (& temp, sizeof temp);

}

подвійний getTemperature () { return * (double *) getValue ();

}}

БЛІ Клієнт

Тепер звернемо увагу на другу половину нашої історії, а саме на те, як бути клієнтом. ESP32 не розміщує послуги, а натомість хоче бути споживачем послуг, розміщених в інших місцях.

Далі подію можна розбити на дві підподії, а саме сканування та взаємодію. Якщо припустимо, що ESP32 запускається і хоче бути клієнтом віддаленого BLE-сервера, то він повинен підключитися до цього сервера. Для того, щоб підключитися до сервера, потрібно знати адресу цільового сервера.

Адреса - це 6-байтове значення, яке зазвичай пишеться у формінn: nn: nn: nn: nn. Хоча в принципі це може бути жорстко закодовано у програмі або введено вручну через якусь іншу історію взаємодії.

Натомість виконуємо процедуру, відому як сканування.

Коли ми виконуємо сканування BLE, отримуємо невеликі записи даних, які завжди містять адресу рекламодавця, а іноді і додаткову інформацію, такі як послуги, які вони надають, або інші описові предмети. Ця інформація надходить до нас пасивно.

Якщо потрібно дізнатись більше про пристрої, можемо підключитися до них і запитати докладнішу інформацію про те, що вони можуть зробити. Це принцип сканування.

У C ++ BLE моделювання зроблено за допомогою BLEScan. Отримуємо екземпляр, за допомогою запиту: BLEScan * pMyScan = BLEDevice :: getScan ().

Об'єкт, який нам повертається, є одностороннім.

Це було вибрано, тому що хочемо сканувати лише один раз на ESP32 у будь-який момент часу. Об’єкт BLEScan для початку сканування викликаємо методом start (), вказуючи інтервал сканування. Інтервал вимірюється в секундах.

Наприклад: pMyScan.start (30); // Сканування протягом 30 секунд.

Це блокуючий дзвінок. Повернеться після закінчення періоду сканування.

Якщо використовуємо BLEScanResults, то можемо запитати, скільки результатів він містить (через виклик getCount ()).

Перед викликом start () можемо зареєструвати функцію зворотного виклику, яка буде викликана для кожного знайденого однорангового пристрою. Тут працює абстрактний клас C ++ під назвою BLEAdvertisedDeviceCallbacks. У ньому є метод onResult (), який буде викликаний для кожного унікального результату, знайденого під час сканування.

Один і той же виявлений пристрій не передається двічі. Цей метод передається у функцію зворотного виклику як екземпляр об’єкту типу BLEAdvertisedDevice.

Наприклад: клас MyCallbacks:

public BLEAdvertisedDeviceCallbacks

{void onResult (BLEAdvertisedDevice advertisedDevice)

{// Зробити щось із знайденим пристроєм

}}

BLEDevice :: init ("");

```
BLESan * pMyScan = BLEDevice :: getScan ();
 pMyScan -> setAdvertisedDeviceCallbacks (new MyCallbacks ());
pMyScan-> start ();
```

Це означає, що для кожного унікального пристрою, який знаходить сканування, буде здійснено виклик onResult (), далі передається екземпляру BLEAdvertisedDevice, який описує цей пристрій.

Пристрій точно скаже його BLE-адресу.

Можливі атрибути, про які може повідомити пристрій, і на даний момент змодельовано наступну підмножину:

• зовнішній вигляд;

• дані виробника;

• назва;

• RSSI;

• первинна послуга UUID;

• передача живлення.

Наприклад, у нашій обробці можемо кодувати:

```
if (advertisedDevice.hasServiceUUID ()) {BLEUUID service = advertisedDevice. getServiceUUID ();
if (service.equals (BLEUUID ((uint16_t) 0x1802)) {// ми знайшли корисний пристрій .
}}
```

Якщо хочемо закінчити сканування раніше, можемо викликати метод stop () об'єкта BLEScan. Посилання на BLEScan зручно і доступне в об'єкті BLEAdvertisedDevice.advertisedDevice.getScan () -> stop ().

Закінчуємо ідентифікацією пристрою до якого ми хочемо підключитися.

Ключовим елементом тепер стає адреса пристрою, яку можемо отримати за допомогою getAddress ().

Тепер можемо звернути свою увагу на фактичне підключення до сервера. Клієнт BLE змодельований як клас BLEClient.

Отримуємо непідключений екземпляр, запитуючи ESP32 за один:

BLEClient * pMyClient = BLEDevice :: createClient ();

Далі вимагаємо з'єднання:

pMyClient-> підключення (адреса);

Connect () - це блокуючий дзвінок, і, повернувшись, ми зв’язані.

На сервері BLE може бути кілька служб. Можемо представити ще дві моделі: BLERemoteService та BLERemoteCharacteristic.

Після підключення до сервера BLE можемо попросити посилання на службу BLERemoteService, що моделює послугу на цьому сервері:

BLERemoteService*pMyRemoteService=pClient->getService serviceUUID); і як тільки отримаємо посилання на службу, можемо запитати службу для посилання на бажану характеристику;

BLERemoteCharacteristic * pMyRemoteCharacteristic= pMyRemoteService-> getCharacteristic (характеристикаUUID); можемо працювати зі значеннями, пов’язаними з характеристикою:

std :: string myValue = pMyRemoteCharacteristic-> readValue (readValue); прочитати значення;

pMyRemoteCharacteristic-> writeValue ("abc"); записати нове значення.

Складемо це в контексті:

// Створити clientBLEClient * pMyClient = BLEDevice :: createClient ();

// Підключити клієнта до сервера pMyClient-> connect (адреса) ;

// Отримати посилання на певну віддалену службу на сервері BLERemoteService * pMyRemoteService = pClient-> getService (serviceUUID);

// Отримати посилання на конкретну віддалену характеристику, що належить службі BLERemoteCharacteristic * pMyRemoteCharacteristic = pMyRemoteServiceter>>;

// Отримуємо поточне значення віддаленої характеристики. Std :: string myValue = pMyRemoteCharacteristic-> readValue ().

Об'єкт BLERemoteCharacteristic надає метод, який називається registerForNotify (callbackHandler). При виклику реєструємо функцію обробника зворотного виклику з таким підписом: void notifyCallback (BLERemoteCharacteristic * pBLERemoteCharacteristic, uint8_t * data, size_t length, bool isNotify). Ця функція викликається, коли сервер BLE передає повідомлення повідомлення. Функція викликається із посиланням на характеристику, пов’язану із сповіщенням, разом із даними та довжиною даних, що передаються сервером. Якщо хочемо скасувати реєстрацію для сповіщень, можемо знову звернутись registerForNotify (), але цього разу передамо NULL як функцію зворотного виклику:

- HttpServer.

До класів C ++ входить простий HTTP-сервер. Це може бути використано для обслуговування файлів з локальної файлової системи, що міститься на ESP32.

Лістінг програми приведено у Додату В.

3 ЕКОНОМІЧНІЙ РОЗДІЛ

Ідея магістерського проекту: дослідження можливостей одночасного керування по відеозв’язку ESP32 по Bluetooth та Raspberry Pi .

Зміст ідеї: передача відео на мікроконтролері ESP32 через Bluetooth на телефон або на комп’ютер бездротовим чином.

Впровадження системи передачі відео дає користувачам бачити все на телефоні в будь якій час, та приймати рішення згідно ситуації. В поєднанні з ефективними методами обробки великих обсягів відеокамери це дозволить знизити ризики аварійних ситуацій.

Перспективи втілення ідеї у практичній діяльності великі, це може бути використано як для домашнього користування, так і для різних сфер діяльності, де потрібно виконувати контроль за процесами та діями.

В даний час розробляється велика кількість різноманітних систем моніторингу для різних областей. Однак вони не поширені і знаходяться на стадії проектування повноцінної системи.

Серед існуючих аналогів можна виділити наступні системи моніторингу:

- "Умний дом",
- "Управління роботамі".

Яскравим прикладом, у всіх сенсах цього слова, можна вважати світлодіодний браслет IoT групи Alibaba. Кожен браслет працює як «піксель», що приймає команди для координованого управління світлодіодним світлом. Це дозволяє формувати «живий і бездротовий екран». Також значної популярності набули такі проект, як:

- DingTalk's M1 (біометрична система відстеження відвідуваності);
- LIFX Mini (серія дистанційно керованих світлодіодних ламп);
- Pium (домашній аромат та аромотерапія).

Подібні пристрої можуть використовуватися у медицині, школах, лабораторіях, системах розумного будинку.

Мікроконтролери можна зустріти в багатьох сучасних приладах, таких як телефони, пральні машини, вони відповідають за роботу двигунів і систем гальмування сучасних автомобілів, з їх допомогою створюються системи контролю і системи збору інформації.

А завдяки тому, що мікроконтролер володіє можливістю передачі інформації за допомогою Wi-Fi та Bluetooth, це дає додаткову можливість для покращення використання. Саме завдяки цим чинникам обумовлений потенціал і використання даного мікроконтролера у промисловості.

Встановивши систему відеоспостереження можливо: здійснювати моніторинг виробничого процесу на кожному з його етапів, мати повну та задокументовану інформацію про нещасні випадки на виробництві; запобігати порушенням якості виробництва, умов праці, техніки безпеки; запобігати розкраданню та псуванню вироблених товарів та продуктів, виробничого майна.

Магістерська робота визначається як науково-дослідницька, основним завданням якої є створення програми управління, тому визначимо керівника проекту як стейкхолдера.

Команда складається з двох працівників даного проекту: керівник проекту та програміст.

Визначення трудомісткості та тривалості

Весь комплекс програмування можна розділити на етапи. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. У науковом дослідженні приймають участь програміст протягом 2.5 місяці і керівник протягом 1 місяця. Тривалість робіт визначають за формулою 3.1:

$$T_{\text{ц}} = \frac{Q}{R} \tag{3.1}$$

де $T_{ц}$ - тривалість циклу, днів;

Q - трудомісткість, людино-днів;

R - кількість виконавців, чол.

$$T_{ц} = \frac{58}{2} = 29$$

Отримана інформація зведена в табл. 3.1

Таблиця 3.1 - Завдання та обов'язки по програмуванню

Найменування роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.-дні	%к підсумку		
1. Отримання технічного завдання	10	13,9	Програміст керівник	5
2. Огляд мікроконтролера ESP32, Bluetooth	10	13,9	Програміст керівник	5
3. Огляд літературних джерел	8	11,1	Програміст керівник	4
4. Розробка інтерфейса и Raspberry Pi	10	13,9	Програміст	10
5. Розробка програмного забезпечення	20	27,8	Програміст	20
6. Тестування ПО	14	19,4	Програміст	14
Разом	72	100		58

Визначення витрат на програмування

Для визначення витрат на програмування складається калькуляція вартісної вартості робіт, яка включає наступні статті:

- основна заробітна плата;

- додаткова заробітна плата;
- єдиний соціальний внесок (ЄСВ);
- витрати на спеціальне обладнання;
- матеріали і комплектуючі вироби;
- накладні витрати;
- податки.

Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих в проекте. Розрахунок зарплати ведеться на підставі даних про трудомісткості, представлених в табл. 3.3.

Таблиця 3.3 - Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн.	Кількість місяців роботи	Сума ЗП, грн.
Програміст	1	8000	58	23200
Керівник	1	9000	14	6300
Разом	2			29500

Планова кількість робочих днів у місяці – 20.

Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою 3.2:

$$ЗП_{доп} = ЗП_{осн} \cdot 0,1 \quad (3.2)$$

Підставивши величину основної заробітної плати в формулу 3.2, отримуємо:

$$ЗП_{доп} = 29500 \cdot 0,1 = 2950 \text{ грн}$$

Відрахування на єдиний соціальний внесок

Вони становлять 22% і беруться від основної та додаткової заробітної плати і розраховують за формулою 3. 3.

$$ОТ = (ЗП_{осн} + ЗП_{доп}) \cdot 0,22 \quad (3.3)$$

$$ОТ = (29500 + 2950) \cdot 0,22 = 7139 \text{ грн.}$$

Визначення затрат на матеріали

У цю статтю включають вартість основних і допоміжних матеріалів, напівфабрикатів, що купуються, і комплектуючих виробів. Транспортно-заготовчі витрати приймають рівними 3-10% від вартості матеріалів. Використовується 3 найменування матеріалів: мікроконтролер с видеокамерой – 489 грн., USB перехідник – 169грн., папір - 105 грн. (1 пачка).

Витрати на комплектуючі розраховують за формулою 3.4:

$$M = \sum_{i=1}^{n}(Ц_i \cdot N_i \cdot (1 + K_{т.з.}) - Ц_{io} \cdot N_{io}), \quad (3.4)$$

де M - витрати на покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{т.з.}$ - коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ - ціна і-го найменування напівфабрикату і комплектуючого, грн.;

N_i - потреба в і-му напівфабрикаті і комплектуючому;

$Ц_{io}$ - вартість зворотних відходів і-го найменування комплектуючого, грн.;

N_{io} - кількість зворотних відходів і-го найменування;

n - кількість найменувань напівфабрикатів і комплектуючих.

$$Ц_{io} = 0;\ N_{io} = 0;\ K_{т.з.} = 0{,}05;$$

$$М = (1 + 0{,}05)\cdot(489+169+105) = 801{,}15 \text{ грн.}$$

Разом, витрати на матеріали становлять 801,15 грн.

Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання.

Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для програмування. Для чого необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування» виходячи з якої визначиться вартість одного машино-години роботи ПК. У табл. 3.4 наведене обладнання яке використовується при роботі.

Таблиця 3.4 – Спеціальне обладнання

Матеріали	Одиниця виміру	Кіль кість	Ціна за одиницю, грн.	Загальна вартість, грн.
ПК RTLINE Gaming X47 v36 (X47v36)	шт.	1	24000	24000
Стіл комп’ютерний	шт.	1	950	950
Крісло комп’ютерне	шт.	2	1100	2200
Транспортно-підготовчі роботи 5%				1357,50
Разом				28507,50

Амортизаційні відрахування визначають за формулою 3.5:

$$A = Ф_б \cdot \frac{Н_а}{100}, \quad (3.5)$$

де $Ф_б$ - балансова вартість обчислювальної техніки, грн .;

$Н_а$ - норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, для ПК 50%.

Варто зазначити, що з 23.05.20 р. стартували зміни до пп. 14.1.138 Податкового кодексу, згідно з якими вартісний критерій ля основних засобів змінено з 6000 грн на 20000 грн. Тож, ті матеріальні цінності, строк корисного використання яких – більше року, але вартість менша чи дорівнює 20000 грн, визнаються малоцінними необоротними матеріальними активами (далі – МНМА). Для них у пп. 138.3.3 ПК передбачено групу 11 «Малоцінні необоротні матеріальні активи», термін використання для яких не визначено. Отже, нарахування амортизації здійснюється за принципом 50/50 (при надходженні - 50% і 50% - при списанні) або ж 100% при надходженні.

Балансова вартість обчислювальної техніки становить 28507,50 грн.

Отримуємо:

$$А = 28507{,}50 \cdot 0{,}5 = 14853{,}75 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію і допоміжні комплектуючі, розрахунок виконують за формулою 3.6:

$$C_е = N_н \cdot Ф_{еф} \cdot K_{зв} \cdot K_{зм} \cdot Ц_е, \quad (3.6)$$

де $N_н$ - номінальна потужність ЕОМ, кВт;

$Ф_{еф}$ - річний ефективний фонд часу роботи ЕОМ, машино-год;

$К_{зв}$ - середній коефіцієнт завантаження за часом;

$К_{ш}$ - коефіцієнт завантаження по потужності;

$Ц_{е}$ - ціна одного кВт-год електроенергії, грн./(кВт-ч).

Номінальна потужність ЕОМ - 0,5 кВт. Ефективний фонд часу роботи ЕОМ становить 464 годин. Середні коефіцієнти завантаження за часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 1,68 грн.

Отримуємо:

$$C_{е} = 0,5 \cdot 464 \cdot 0,9 \cdot 0,6 \cdot 1,68 = 210,47 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 855,23 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 8798,59 грн., 5% від суми складають 439,93 грн.

Розраховані статті витрат на утримання і експлуатацію устаткування внесені в табл. 3.4.

Таблиця 3.4 - Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	14853,75
Поточний ремонт обладнання	855,23
Інші витрати	439,93
Витрати по електропостачанню	210,47
Разом	16359,38

Витрати на оплату машинного часу ЕОМ для налагодження програмних засобів визначаються за формулою 3.7:

$$C_{мо} = P_{екс} \cdot t_{мо}, \quad (3.7)$$

де $C_{мо}$ - витрати на оплату машинного часу, грн.;

$P_{екс}$ - експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год.;

$t_{мо}$ - машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу розраховуємо діленням суми витрат за кошторисом «Витрати на утримання і експлуатацію устаткування» (табл. 3.4) на ефективний фонд часу роботи ЕОМ. Ефективний фонд часу роботи ЕОМ дорівнює 464 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$P_{екс} = 16359{,}38/464 = 35{,}26 \text{ грн./машино-год}$$

ЕОМ експлуатується 58 днів в одну зміну, що становить в сумі 290 годин. Таким чином, витрати на оплату машинного часу складуть:

$$С_{мо} = 35{,}26 \cdot 290 = 10389{,}40 \text{ грн.}$$

Розрахунок загальновиробничих витрат

Загальновиробничі витрати включаються до вартості програмування непрямим шляхом - у відсотках до основної заробітної плати розробників. В даному випадку загальновиробничі витрати становлять 40% до основної заробітної плати розробників, що складає 11600 грн.

Результати визначення витрат на програмування у вигляді калькуляції кошторисної вартості робіт наведені в табл. 3.5.

Таблиця 3.5 - Калькуляція кошторисної вартості робіт з програмування

№	Найменування статей витрат	Сума, грн.	Питома вага до підсумку,%
1	Основна заробітна плата	29500	27,5
2	Додаткова заробітна плата	2950	2,7
3	ЄСВ	7139	6,6
4	Матеріали	801,15	0,7
5	Витрати на спец. обладнання	28507,50	26,6
6	Інші прямі витрати	16359,38	9,7
7	Витрати на оплату машиного часу	10389,40	15,2
8	Загальновиробничі витрати	11600	11
9	Разом ($S_{пр}$)	107446,03	100

Розрахунок техніко-економічної ефективності

Для теоретичних досліджень у більшості випадків важко чи навіть неможливо розрахувати економічний ефект, тому визначаємо техніко-економічну ефективність з урахуванням наступних показників:

- важливості дослідження;
- складності розробки;
- результативності й можливості використання.

Результативність НДР визначається по повноті рішень поставленого завдання: отриманий результат відповідає планованому, задовільний (часткове рішення) чи негативний.

Аналіз залежності між цими показниками й витратами на їхнє досягнення дає можливість кількісної оцінки техніко-економічної ефективності теоретичних НДР і визначається за формулою (3.8):

$$K_{НДР} = \frac{J^n \cdot R \cdot T}{B_{НДР} \cdot t_{НДР}}, \quad (3.8)$$

де $K_{НДР}$- рівень ефективності дослідження (коефіцієнт техніко-економічної ефективності МР):

J^n - важливість роботи;

R - результативність роботи;

T - технічна складність виконання НДР;

$B_{НДР}$ - витрати на проведення НДР:

n - показник використання результатів НДР:

n = 0 - результати НДР не використовуються;

n = 1 - результати НДР використовуються частково;

n = 2 - результати НДР використовуються в дослідно-конструкторських роботах (ДКР);

n = 3 - результати НДР можуть бути використані без проведення ДКР.

Для НДР, у яких $B_{НДР}$ >30 тис. грн. і $t_{НДР} \leq 2$ років, можна застосовувати такі значення оцінних факторів наведених в табл. 3.6

Таблиця 3.6 – Значення оцінних факторів

Оцінні фактори	J	R	T	C	$t_ф$	n
Припустимі значення	2…5	1…4	1…3	-	-	1…8
Прийняті значення	5	2	3	-	-	6

Згідно значень з таблиці оцінних факторів, отримуємо такий вираз:

$$К_{ндр} = \frac{5^6 \cdot 2 \cdot 3}{107446{,}03 \cdot 0{,}72} = 1{,}21$$

Таким чином, так як коефіцієнт техніко-економічної ефективності НДР $К_{НДР} \geq 1$, в нашому випадку рівний $К_{НДР} = 1{,}21$, то дослідницька робота вважається ефективною.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз потенційних небезпек

Тема дипломного проекту «Система віддаленого управління ESP32 по Bluetooth та Raspberry Pi», тому розглянемо робоче місце інженера-радіотехніка. Роботи проводяться в приміщенні лабораторії, основні потенційні небезпеки та шкідливі виробничі чинники:

- потенційні небезпеки фізичного характеру: незадовільна організація робочого місця, зокрема не раціональне розташування технологічного обладнання та захаращеність робочої зони при роботі з персональним комп'ютером та приладами;
- небезпека ураження електричним струмом, внаслідок недотримання правил електробезпеки;
- розумове та емоційне перевантаження внаслідок недотримання змін режимів праці та відпочинку, що визиває зниження працездатності та стомлення або підвищене психофізіологічне навантаження;
- негативний вплив недостатнього освітлення робочої зони на зір;
- негативний вплив підвищеного рівня шуму на психоемоційний стан, пов'язаний з використанням кондиціонерів, освітлювальних приладів;
- негативний вплив електромагнітних полів, які створюються офісним обладнанням, на організм працюючого;
- негативний вплив на опорно-руховий апарат працюючого при недостатній рухливості протягом робочого часу;
- негативний вплив повітряного середовища, внаслідок несправності системи вентиляції;
- нервово-психічні перевантаження внаслідок контактів з колегами по роботі, керівництвом;

- не виконання вимог до обладнання і організації робочих місць користувачів ПК та вимог до режиму праці й відпочинку;
- небезпека загоряння, яка може бути у зв'язку із несправністю електричного обладнання, порушення правил протипожежної безпеки;
- неправильні дії персоналу у надзвичайних ситуаціях.

4.2 Заходи по забезпеченню техніки безпеки

Комп'ютерне обладнання потрібно підключи до електромережі за допомогою тільки справних засобів заводського виготовлення.

При використанні штепсельних з'єднань та електророзеток, повинні бути спеціальні контакти для підключення нульового захисного провідника. Приєднання нульового захисного провідника повинно виконувати раніше, ніж приєднання фазового та нульового робочого провідників. Але порядок роз'єднання при відключенні повинен бути бути зворотним.

Не дозволяється підключати комп'ютерну техніку до звичайної двопровідної електромережі. Заборонено використання перехідних пристроїв.

Приміщення лабораторії відноситься до приміщень де не має підвищеної небезпеки ураження електричним струмом.

Обладнання живиться від змінного струму 220 В від мережі. Захист працівників від ураження електричним струмом повинно відповідати требованиям СОУ 31.2-21677681-19:2009:«Випробування та контроль стану пристроїв заземлення електроустановок. Типова інструкція, затвердженого наказом Міністерства палива та енергетики України від 29 грудня 2009 року № 772».

Відповідати требованиям повинні стаціонарні комп'ютери та освітлювальні прилади, кондиціонери, опалювальні пристрої, ноутбуки, сканери.

Способ захисту людини від ураження електричним струмом відповідає вимогам згідно з СОУ 31.2-21677681-19:2009: «Випробування та контроль

стану пристроїв заземлення електроустановок. Типова інструкція, затвердженого наказом Міністерства палива та енергетики України від 29 грудня 2009 року № 772».

Згідно ПУЕ («Правила устрою електроустановок») виконані заходи з електробезпеки:

- захист від випадкового дотику до струмопровідних частин за допомогою їх ізоляції та захисних оболонок.

Защитное заземление: у лабораторії влаштовано занулення.

З усіма працівниками, які прийняти на роботу, а також у період роботи проводяться навчання, інструктажі з питань:

- охорони праці (вступний, первинний, повторний, позаплановий, цільовий);
- надання першої допомоги потерпілим від нещасних випадків;
- правила поводження при виникненні аварій;
- навчання з правил електробезпеки;
- перевірка знань та атестація персоналу на отримання та підвищення групи з електробезпеки.

Виконані основні вимоги до моніторів відповідно розділу «Мінімальні вимоги з охорони праці», директиви ЕС 90/270 ЕЕС:

- символи на екрані повинні бути чіткі і добре розрізнятися;
- зображення позбавлене блимання;
- яскравість та контрастність легко регулюються;
- екрани вільні від відблисків і відбиття;
- випромінювання знижені до надзвичайно малих рівнів;
- при розміщенні декількох робочих місць в одному приміщенні мінімальна площа для одного робочого місця складає 6 м2;
- екран монітора має бути розміщений на оптимальній відстані від очей користувача (60-70 см), але не ближче 50 см;

- екран повинен бути розташований у вертикальній площині під кутом + 30° до нормальної лінії погляду працівника;
- поверхня клавіатури має бути з антистатичними властивостями;
- під час роботи необхідно робити перерви для розвантаження очей.

Робота за ПК викликає загальну втому, яка може призвести до пошкоджень різних ділянок хребта — шийного, грудного, попереково-крижового. Якщо руки під час друкування мають неправильне положення , це буде призводить до хронічних захворювань кисті. Клавіатура має розташовуватися на відстані 10-15 см від краю столу.

Нервове напруження також впливає на серцево-судинну систему. У працівників збільшується артеріальний тиск і частота пульсу. Це може привести до небажаних захворювань, таких як инсульт, порушення мозгового кровообігу.

Також тривала та інтенсивна робота провокують появлення синдрому комп'ютерного стресу (СКС). В результаті працівники відчують головний біль, запалення очей, алергією, дратівливість, млявість і депресією.

Кожні 2 години на 15 хвилин у роботі повинні бути перерви. Для зняття напруги м'язів спини, шиї, рук і ніг повинно проводити виконання фізичних вправ 2-3 рази протягом робочого часу. Згідно вимог до режиму праці та відпочинку при роботі з ПК ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ».

До робочого місця з комп’ютером повино надходити свіже повітря.

Необхідно провітрювати приміщення, та робити вологе прибирання не рідше одного разу на день. Обов’язково слід протирати поверхню столу серветкою з антистатиком. Монітори, ПК, клавіатура також повинні протиратися спеціальними засобами.

Матеріали, які були використані для ремонта та оздоблення приміщень можуть мати хімічні речовини у повітрі робочої зони. Серед них синтетичні

матеріали для підлоги (ворсоніт, ковролін, лінолеум), полімерні матеріали для оздоблення стін. Вміст формальдегіду у повітрі робочої зони повинен бути на рівні 0,5 ГДК.

Нормативним значенням еквівалентного рівня звуку для програмістів є 50 дБА.

4.3 Заходи з виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для приміщення лабораторії обладнаної ПК з ВДТ розроблені відповідно до вимог Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», МЮУ 06.05.2014 р. за № 472/25249, ДСанПіН 3.3.2.007-98 «Державні стандартні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» і НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин».

Технічні заходи передбачають:

- систематичне підтримання чистоти у приміщеннях і на робочих місцях;
- забезпечення санітарно-гігієнічних вимог до повітря виробничого середовища;
- наявність систем вентиляції та кондиціювання робочих місць зі шкідливими умовами праці;
- забезпечення захисту працюючих від шуму, ультра- та інфразвуку, вібрації, різних видів випромінювання.

Показники мікроклімату в лабораторії відповідають встановленим санітарно-гігієнічним вимогам ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень», ГОСТ 12.1.005-88 (1991) «ССБТ. Общие

санитарно-гигиенические требования к воздуху рабочей зоны» і ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень».

Роботи в приміщені лабораторії повинні мати наступні оптимальні значення параметрів мікроклімату:

- у холодний період року: температура 21-23°С;
- відносна вологість: 40 -60%;
- швидкість переміщення повітря: 0,1 м/с;
- у теплий період року: температура 22-24°С;
- відносна вологість: 40- 60%;
- швидкість переміщення повітря: 0,2 м/с.

Згідно зі ст.7 Закону України «Про забезпечення санітарного й епідеміологічного благополуччя населення», завданням адміністрації є розробка і проведення санітарних і протиепідеміологічних заходів. Адміністрація виконує контроль за підтримкою вимог санітарних норм та інформувати державну санітарну епідеміологічну службу про надзвичайні ситуації.

Згідно ДБН В.2.5-28-2006 «Природне і штучне освітлення» в офісних приміщеннях використовується природне та штучне освітлення. Природне освітлення здійснюється через світлові прорізи, воні забезпечують коефіцієнт природної освітленості (КПО) не нижче 1,5%. Для захисту від прямих сонячних променів використовують сонцезахисні пристрої, на вікнах встановлюють жалюзі або штори, щоб уникнути відблисків на поверхні екранів.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300–500 лк.

Як джерела світла для штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛБ.

Згідно ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» формується рівень звукового тиску.

Зниження рівня шуму в приміщенні здійснюються за допомогою:

- використання більш сучасного обладнання;
- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;
- переведення жорсткого диска в режим сну (Standby), якщо комп'ютер не працює протягом визначеного часу;
- використання блоків живлення ПК з вентиляторами на гумових підвісках.

Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99, затверджені постановою Головного державного санітарного лікаря України від 01.12.99 р. № 39;

Рівні шуму та вібрації на робочих місцях осіб, що працюють з ПК, визначаються відповідно до ДСанПіН 3.3.2.007-98.

Вимоги щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних та магнітних полів встановлюються відповідно до ДСанПіН 3.3.2.007-98, а також вимог до роботодавців щодо захисту працівників від шкідливого впливу електромагнітних полів, затверджених наказом Міненергетики від 05.02.2014 р. № 99, ДСанПіН 3.3.6.096-2002.

Інтенсивність потоків ультрафіолетового випромінювання не повинна перевищувати допустимих значень, відповідно до СН 4557–88.

Внаслідок роботи з ПК, на працівників негативно впливають електромагнітні випромінювання. Згідно НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» та ДСанПіН 3.3.6.096-2002 «Державні санітарні норми і правила при роботі з

джерелами електромагнітних полів», а також ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», на робочих місцях обладнаних встановлюють рідкокристалічні монітори , які не є джерелами рентгенівського та електромагнітного випромінювань.

Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- розміщення виробничих меблів з урахуванням характеристик людини;
- раціональна компоновка обладнання на робочих місцях;
- врахування характеру і особливостей трудової діяльності.

Правила охорони праці під час експлуатації електронно-обчислювальних машин, затверджені наказом Держгірпромнагляду від 26.03.2010 р. № 65.

Комп'ютеризовані робочі місця краще розмістити рядами вздовж стіни з вікнами. Це дасть змогу унеможливити дзеркальне відбиття на екрані ВДТ джерел природного світла (вікон) та не буде потрапляти у очі операторів.

Вимоги до виробничих приміщень для експлуатації ПК:

- об'ємно-планувальні рішення будівель та приміщень для роботи з ПК мають відповідати вимогам ДСанПіН 3.3.2.007–98;
- розміщення робочих місць з ПК у підвальних приміщеннях, на цокольних поверхах заборонено;
- площа на одне робоче місце становить не менше ніж 6,0 м3, а об'єм – не менше ніж 20,0 м3;
- приміщення для роботи з ПК повинні мати природне та штучне освітлення відповідно до СНиП П-4–79;
- природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче, ніж 1,5%;

- виробничі приміщення повинні обладнуватись шафами для зберігання документів, полицями, стелажами;
- приміщення із ПК повинні бути оснащені аптечками першої медичної допомоги;
- при приміщеннях із ПК мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження.

Гігієнічні вимоги до параметрів приміщень із ПК:

- гігієнічні вимоги включають вимоги до параметрів мікроклімату, освітлення, шуму й вібрації, рівнів електромагнітного та іонізуючого випромінювання;
- з метою зменшення негативного впливу монотонності є доцільним чергувати операції обробки тексту і числових даних, та ввод даних з редагуванням текстів.

Для зниження нервово-емоційного напруження, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, доцільні перерви використовувати для виконання комплексу вправ, приклади яких також наведено в ДСанПіН 3.3.2.007-98.

Повітрообмін – це часткова або повна заміна забрудненого повітря в приміщенні свіжим і чистим зовнішнім.

Величина потрібного повітрообміну залежить від призначення приміщення, кількості шкідливих речовин, що виділяються в приміщення, гранично допустимої концентрації (ГДК) шкідливих речовин.

Розрахуємо повітрообмін для приміщення цеху з новим обладнання.

Для приміщення з нормальним мікрокліматом, без виділення шкідливих речовин, потрібний повітрообмін L ($м^3$/год) визначають кількома засобами, розглянемо розрахунок по тепловиділенню для цеху.

Розміри цеху: 40х60х6, обладнання потужністю 100 кВт/год., працює 120 чол., 16 вікон розміром 3х2 м з подвійним склом, потужність світильників загального освітлення 12 кВт/год., повітря яке надходить ззовні 2400 кг/год,

температура повітря для холодного періоду $t_з$ = -14°С; для теплого періоду $t_л$ = + 25°С.

Розрахунок за надлишковим тепловиділенням

Повітрообмін для видалення надлишкового тепла визначається за формулою (4.1):

$$L = \frac{Q_{\text{надл}}}{c \cdot \rho_{\text{пр}} \cdot (t_{\text{вид}} - t_{\text{пр}})}, \quad (4.1)$$

де $Q_{надл}$ - надлишкове тепловиділення, кКал/год;

$\rho_{пр} = 1{,}2$кг/м^3;

$t_{вид} = t_{р.з.} + \Delta t(6-2) = 25 + 2 \cdot (6-2) = 33°C$;

$t_{р.з.}$- температура повітря в робочій зоні, °С;

$\Delta t = (1-5)°$С/м– температурний градієнт;

H - висота приміщення, м.

Визначення надлишкового тепловиділення $Q_{надл}$ визначається за формулою (4.2):

$$Q_{надл} = Q - Q_{вих}, \quad (4.2)$$

де Q – загальна кількість тепла, яка надходить до приміщення, кКал/год;

$Q_{вих}$ – загальна кількість тепла, яка відводиться з приміщення, кКал/год.

Визначення загальної кількісті тепла, яке надходить до приміщення визначається за формулою (4.3):

$$Q = Q_{дв} + Q_{осв} + Q_{сон} + Q_л, \quad (4.3)$$

де $Q_{дв}$ - виділення тепла від електродвигунів, кКал/год;

$Q_{осв}$ - виділення тепла від освітлюваних приладів, кКал/год;

$Q_{сон}$ - виділення тепла від сонячної радіації, кКал/год;

$Q_л$ - виділення тепла від працюючих людей. кКал/год.

Визначення загальної кількісті тепла, яка відводиться з приміщення, визначається за формулою (4.4):

$$Q_{вих} = Q_{дод} + Q_{втр}, \quad (4.4)$$

де $Q_{дод}$ - втрати тепла на нагрів повітря, яке надходить у приміщення, кКал/год;

$Q_{втр}$ - втрати тепла через конструкції цеху, кКал/год.

Визначення виділення тепла від електродвигунів визначається за формулою (4.5):

$$Q_{дв} = \sum N \cdot 860 \cdot \psi_1 \cdot \psi_2 \cdot \psi_3 \cdot \psi_4, \quad (4.5)$$

де $\sum N$ - сумарна потужність двигунів, кКал/год;

860 – теплоелектричний еквівалент;

ψ_1 - середній ККД електродвигунів;

ψ_2 - коефіцієнт використання двигунів;

ψ_3 - коефіцієнт одночасності роботи двигунів;

ψ_4 - коефіцієнт, який характеризує перехід механічної енергії в теплову.

При роботі обладнання без спеціального охолодження $\psi_1 \cdot \psi_2 \cdot \psi_3 \cdot \psi_4$ =0,1.

$$Q_{дв}=100*860*0,1= 8600 \text{ кКал/год}$$

Визначення виділення тепла від освітлюваних приладів визначається за формулою (4.6):

$$Q_{осв} = \sum N \cdot 860 \qquad (4.6)$$

де $\sum N$- потужність світильників загального освітлення, кВт/год.

$$Q_{осв} = 12 \cdot 860 = 10320 \text{ кКал/год.}$$

Визначення виділення тепла від сонячної радіації визначається за формулою (4.7):

$$Q_{сон} = Q_o + Q_n, \qquad (4.7)$$

де $Q_o = q_o \cdot A_o \cdot F_o$ - надходження тепла через засклені отвори, кКал/год;

$Q_n = q_n \cdot K_n \cdot F_n$ - надходження тепла через перекриття цеху, кКал/год;

q_o, q_n - величини радіації;

F_o, F_n - площі засклених поверхонь та перекриття відповідно, м^2;

A_o- коефіцієнт, який враховує вид засклених отворів, A_o для вікон = 1,15

K_n - коефіцієнт теплопередачі перекриття.

$$Q_o = 125 \cdot 16 \cdot 6 \cdot 1{,}15 = 13800 \text{ кКал/год.}$$

$$Q_n = 18 \cdot 0{,}75 \cdot (40 \cdot 60) = 32400 \text{ кКал/год.}$$

$$Q_{сон} = 13800 + 32400 = 46200 \text{ кКал/год.}$$

$$Q_л = 120 \cdot 225 = 27000 \text{ кКал/год.}$$

Сумарне надходження тепла:

- для холодного періоду $Q = 8600 + 10320 + 27000 = 45920$ кКал/год;

- для теплого періоду $Q = 45920 + 46200 = 92120$ кКал/год.

Визначення втрати тепла на нагрів повітря, яке надходить у приміщення визначається за формулою (4.8):

$$Q_{дод} = 0{,}24 \cdot G \cdot (t_в - t_з),\ \text{кКал/год} \qquad (4.8)$$

$$Q_{дод} = 0{,}24 \cdot 2400 \cdot (25 + 14) = 22464\ \text{кКал/год}$$

Визначення втрати тепла через конструкції цеху, визначаються за формулою (4.9):

$$Q_{втр} = \sum F \cdot n \cdot K_n \cdot (t_в - t_з), \qquad (4.9)$$

де F - площа поверхні конструкцій цеху, м^2;

n =0,6 – коефіцієнт;

K_n- коефіцієнт тепловіддачі конструкцій.

Втрати тепла через конструкції цеху розраховуємо тільки для холодного періоду року.

Розрахунок для засклених отворів:

$Q_{втр} = 96 \cdot 0{,}6 \cdot 2{,}5 \cdot (25 + 14) = 5616$ кКал/год.

Розрахунок для підлоги:

$Q_{втр} = 2400 \cdot 0{,}6 \cdot 0{,}19 \cdot (25 + 14) = 10670$ кКал/год.

Розрахунок для стін:

$Q_{втр} = 1104 \cdot 0{,}6 \cdot 0{,}67 \cdot (25 + 14) = 17309$ кКал/год.

Розрахунок для стелі:

$Q_{втр} = 2400 \cdot 0{,}6 \cdot 0{,}75 \cdot (25 + 14) = 42120$ кКал/год.

Сумарні втрати

$Q_{втр} = 5616 + 10670 + 17309 + 42120 = 75715$ кКал/год.

Розрахунок надлишкової кількості тепла:

- для холодного періоду року:

$$Q_{надл} = 45920 - 75715 - 22464 = -52259\ \text{кКал/год}$$

- для теплого періоду:

$Q_{надл} = 92120$ кКал/год.

Повітрообмін для видалення надлишкового тепла складає:

$$L = \frac{92120}{0{,}24 \cdot 1{,}2 \cdot (33-25)} = 39983 \text{ м}^3/\text{год}$$

4.4 Заходи безпеки у надзвичайних ситуаціях

4.4.1 Заходи з пожежної безпеки

Заходи з пожежної безпеки відповідають документу «Документ - 948/2019 «Про невідкладні заходи щодо запобігання пожежній небезпеці в Україні», поточна редакція — Прийняття від 24.12.2019.

Наказом МВС України від 30.12.2014 № 1417 «Про затвердження правил пожежної безпеки в Україні» зі змінами від 31.07.2017 № 657 та наказом ДСНС України від 12.07.2016 № 335 затверджений примірний Перелік документів з питань цивільного захисту.

Працівники об’єкта мають бути ознайомлені з цими вимогами на інструктажах або під час проходження пожежно-технічного мінімуму.

Для кожного приміщення об’єкта мають бути розроблені та затверджені керівником об’єкта або уповноваженою ним посадовою особою інструкції про заходи пожежної безпеки.

Знаки безпеки, їх кількість, а також місця їх встановлення повинні відповідати ДСТУ ISO 6309:2007 «Протипожежний захист. Знаки безпеки. Форма та колір» (ISO 6309:1987, IDT) та ГОСТ 12.4.026-15 «ССБТ. Цвета сигнальные, знаки безопасности и разметка сигнальная».

Метою пожежної безпеки об’єкта є попередження виникнення пожежі, а у випадку виникнення пожежі – обмеження її розповсюдження, своєчасне виявлення, гасіння пожежі, захист людей і матеріальних цінностей.

Комплекс протипожежних заходів для виробничого приміщення обладнаного ПК з ВДТ розроблений згідно вимог НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні».

Для співробітників дуже важливо виконання елементарних правил пожежної безпеки під час перебування на робочому місці.

Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати.

Засоби протипожежного захисту слід утримувати у справному стані. Усі працівники повинні вміти користуватись наявними вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження Відстань від найбільш віддаленого місця приміщення до місця розташування вогнегасника не повинна перевищувати 20 м.

Комплекс протипожежних заходів для виробничого приміщення (дослідницької лабораторії, конструкторського бюро тощо) обладнаного ПК з ВДТ розроблений згідно вимог НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні».

Створюються графіки проходження інструктажів з пожежної безпеки співробітників.

У лабораторному приміщенні знаходяться дерев'яні меблі, електронна апаратура, паперові носії інформації.

У разі виникнення пожежі в приміщенні лабораторії для евакуації персоналу відповідно до вимог ДБН В. 1.1-7-2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидва боки приміщення, з одного боку вікно (на пожежну драбину), а з іншого - вхідні двері. Згідно п. 2.29 СНиП 2.09.02-85 * «Виробничі будівлі», відстань від

найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Устаткування, силові і освітлювальні мережі приміщення лабораторії відповідають вимогам пожежної безпеки, так як виконані відповідно до вимог НПАОП 40.1-1.32-01 «Правила улаштування електроустановок. Електрообладнання спеціальних установок», і мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежної небезпеки П-II А, до якого належить приміщення.

Крім цього, згідно з вимогами ДБН В.2.5-56-2014 «Системи протипожежного захисту», приміщення лабораторії обладнано автоматичними пожежними сповіщувачами ДІП-1, які забезпечують виявлення димових ознак виникнення пожежі.

4.4.2 Заходи з цивильного захисту

Електромагнітний імпульс - це сильні перемінні електромагнітні поля, виникають внаслідок дії гамма-променів на атоми навколишнього середовища і утворення потоку електронів та плюсових іонів. Електромагнітний імпульс діє на дротові та кабельні лінії електропередач і зв'язку, збільшує в них напругу, внаслідок чого пробивається ізоляція, пошкоджуються вхідні елементи апаратури, вигоряють плавкі вставки, а також уражуються люди, які обслуговують ці засоби. Тривалість уражуючої дії ЕМІ - декілька мілісекунд.

Електромагнітний імпульс, який виникає під час ядерного вибуху більшість своєї енергії переносить в електромагнітних хвилях з частотою в діапазоні від 3 Гц до 30 кГц за напруженості магнітного поля, що досягає 50000 В/м.

Зміна властивостей іоносфери Землі, викликана ЕМІ призводить до появи завад у радіозв'язку.

Найкращим захистом від електромагнітного імпульсу є відключення цих пристроїв від мереж при наближенні грози.

При наземному і низькому повітряному вибухах вплив ураження від електромагнітного імпульсу спостерігається на відстані до декількох кілометрів від епіцентру вибуху.

Дія ураження від електромагнітного імпульсу проявляється стосовно до радіоелектронної та електротехнічної апаратури, транспортних засобів та інших об'єктів, що використовують електричну енергію. Струми і напруги можуть викликати пробій ізоляції, пошкодження трансформаторів, псування напівпровідникових приладів, перегорання плавких вставок та інших елементів радіотехнічних пристроїв.

Уражаюча дія ЕМІ в приземній області та на землі пов'язана з акумулюванням його енергії довгими металевими предметами, антенами, лініями електропередачі і зв'язку. У них виникають сильні наведені струми, які руйнують підключене електронне та інше чутливе устаткування. У районі дії ЕМІ безпосередній контакт людини зі струмопровідними предметами дуже небезпечний.

ЕМІ уражає радіоелектронну і радіотехнічну апаратуру. Найбільш уразливими елементами обладнання є напівпровідникові прилади - транзистори, діоди, кремневі випрямлячі, цифрові процесори, управляючі й контрольні прилади. Чутливі до пошкодження ЕМІ транзистори звукової частоти, перемикаючі транзистори, інтегруючі ланцюги та ін.

ЕМІ небезпечний і за наявності міцних споруд, які розраховані на стійкість протиударної хвилі наземного ядерного вибуху.

Сучасний рівень знань про природу і властивості ЕМІ дає можливість розробити захист від нього і впровадити заходи захисту, до яких входять:

- схеми, стійкі до електромагнітної інтерференції;
- радіоелектронні елементи стійкі до ЕМІ;
- екранування окремих пристроїв або цілих електронних систем.

Нові технології пропонують протидії потужному ЕМІ – радіоізотопно-плазмову технологію, яка забезпечує поглинання потужних ЕМІ у широкому діапазоні частот.

З великою часткою ймовірності невеликі системи не будуть порушені ЕМІ, якщо вони ізольовані від мережі живлення. Під час вступу попередження потрібно вимкнуть усі підключені до електричної розетки прилади та пристрої. Відключити вентиляцію і термостати. Весь будинок потрібно відключити від загальної мережі.

Усі електронні пристрої, електроприлади, якщо вони не використовуються потрібно відключити; не залишайте принтери і сканери в режимі очікування. Використовуйте короткі кабелі для роботи, компоненти з автономними батареями, рамкові антени. Підключіти всі проводи заземлення до однієї спільної точки заземлення.

Слід заздалегідь продумати захисну систему. Наприклад, резервний генератор, не буде пошкоджений сонячної бурею, але ЕМІ може пошкодити чутливі електронні контролери, так що екранування є доцільним.

ВИСНОВКИ

Для реалізації поставленої задачі було обрано мікроконтролер ESP32.

В ході виконання дипломного проекту були виконані всі поставлені завдання та були отримані наступні результати:

– проведено аналіз використання мікроконтролера ESP32, Bluetooth з Raspberry Pi;

– виконано розробку програми передавання відео с ESP32 по wifi з управлінням ESP32 по Bluetooth;

– проведено встановлення ESP32 в середу Arduino IDE;

– проведено відлагодження програми;

– проведено тестування програми.

ПЕРЕЛІК ПОСИЛАНЬ

1. Веб-сервер для потокового видео на базе ESP32-CAM – Режим доступу: https://voltiq.ru/esp32-cam-video-web-server/ (дата звернення: 20.112020)

2. Микроконтроллер ESP32 и проекты Arduino. URL – Режим доступу: https://arduinomaster.ru/platy-arduino/esp32-arduino-raspinovka-arduino-ide/ (дата звернення:15.11.2020)

3. Начало работы с ESP32 Bluetooth в Arduino IDE – Режим доступу: https://diytech.ru/projects/nachalo-raboty-s-esp32-bluetooth-v-arduino-ide (дата звернення: 25.11.2020)

4. Связь ESP32 со смартфоном по Bluetooth через Arduino – Режим доступу: https://voltiq.ru/esp-32-serial-bluetooth-in-arduino-ide/ (дата звернення: 28.11.2020)

5. Совмещенные чип и модуль Bluetooth/Wi-Fi производства Espressif Systems – Режим доступу: https://wireless-e.ru/radiomoduli/esp32/ (дата звернення: 02.12.2020)

6. Установка ESP32 в Arduino IDE – Режим доступу: https://voltiq.ru/instruction-installing-esp32-board-in-arduino-ide-for-windows (дата звернення: 30.11.2020)

7. Як віддалено керувати Raspberry Pi з будь-якої точки світу – Режим доступу:http://isearch.kiev.ua/uk-searchpractice-methodsinstruments-1938-as-remotely-manage-raspberry-pi-from-anywhere-in-the-world (дата звернення: 30.11.2020)

8. Arduino YUN – Режим доступу: http://digitrode.ru/articles/1668-esp32-i-bluetooth-svyaz-po-bluetooth-serial-v-arduino-ide.html (дата звернення: 16.11.2020)

9. ESP32 - Вікіпедія – Режим доступу: https://uk.wikipedia.org/wiki/ESP32 (дата звернення: 02.12.2020)

10. ESP32 Datasheet. Espressif Systems. 2017-03-06. – Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 02.12.2020)

11. ESP32 Bluetooth Classic с Arduino IDE – Режим доступу: https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide (дата вернення: 02.12.2020)

12. RASPBERRY PI – Режим доступу: https://www.raspberrypi.org-products/raspberry-pi-3-model-b (дата звернення: 03.12.2020)

13. SparkFun ESP32 Thing – Режим доступу: https://www.sparkfun.com - products/13907 (дата звернення:27.11.2020)

14. How to setup Bluetooth on a Raspberry Pi 3 – Режим доступу: https://www.cnet.com/how-to/how-to-setup-bluetooth-on-a-raspberry-pi-3/

ДОДАТОК А

Лістинг програми керування по Bluetooth

```
/*********
  - Вибираємо плату "ESP32 Wrover Module"
  - Вибираємо у  Partion Scheme "Huge APP (3MB No OTA)
  - GPIO 0 и GND замикаємо перемичкою для заливки скетчу
  - Після заливки скетчу, потрібно відключити контролер і розімкнути перемичку, для того щоб ESP32 завантажилася в стадартному режимі і запрацювала  програма
*********/

#include "BluetoothSerial.h":// Підключаємо бібліотеку Bluetooth

#if!defined(CONFIG_BT_ENABLED) // Конфігурація Bluetooth ввімкнена
|!defined(CONFIG_BLUEDROID_ENABLED) // Конфігурація ввімкнена
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it // Bluetooth не ввімкнен. Запустить меню конфігурації
#endif

BluetoothSerial SerialBT; // створіть екземпляр Bluetooth послідовний

void setup () {// Вводимо налаштування
 Serial.begin(115200); // Форматувати послідовний зв'язок зі швидкістю
 SerialBT.begin("ESP32test"); //Bluetooth device name //Змінюємо ім’я
 Serial.println("The device started, now you can pair it with bluetooth!"); // Пристрій запущено
}

void loop() {
 if (Serial.available()) {:// перевіряємо, чи надходять байти в послідовний порт
  SerialBT.write(Serial.read()); // відправляє, повертає дані через послідовний порт Bluetooth.
 }
 if (SerialBT.available()) {: // перевіряє доступні байти для читання в послідовному порту Bluetooth.
  Serial.write(SerialBT.read()); // записуємо байти в Serial Monitor.
```

```
  }
  delay(20); // Встановлення затримки
}
```

ДОДАТОК Б

Листинг об’єднання кода Bluetooth та web-сервера

```
/*********
  - Вибираємо плату "ESP32 Wrover Module"
  - Вибираємо у  Partion Scheme "Huge APP (3MB No OTA)
  - GPIO 0 и GND замикаємо перемичкою для заливки скетчу
  - Після заливки скетчу, потрібно відключити контролер і розімкнути перемичку, для того щоб ESP32 завантажилася в стадартному режимі і запрацювала  програма
*********/
#include "Utils.h" //Включає бібліотеку Утиліти
#include "esp_camera.h"//Включає бібліотеку Камера
#include <BLEDevice.h>//Включає Блютуз Пристрій
#include <BLEUtils.h>//Включає Блютуз Утиліти
#include <BLEServer.h>//Включає Блютуз Сервер
#include <WiFi.h>//Включає бібліотеку Вай-Фай

#define CAMERA_MODEL_AI_THINKER//Вибираемо модуль камери

#include "camera_pins.h"//Включає затримку камери

const char* ssid = "fallenfallen";//Вводимо логин
const char* password = "12021975";//Вводимо пароль
BLECharacteristic *pCharacteristic;//Блютуз характеристики

void startCameraServer();// Запускается КамераСервер

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"//Визначає характеристики

class BLECb: public BLECharacteristicCallbacks {//Класифіцируе зворотні характеристики блютуса
  void onWrite(BLECharacteristic *pCharacteristic) //Вводить та записує характеристики
  {
    Byte b;//Число байтів
    uint8_t* value = pCharacteristic->getData();//Отримуемо пакет данних
```

```
    if(value!=nullptr)//Якщо значення нуль
      b.byte = value[0];//Значення байтів

    if(b.bit1)// Якщо байтів один
       digitalWrite(4, HIGH);//Електроний ввод, чотири, Високий
    else //Ще
      digitalWrite(4, LOW);//Електроний ввод(чотири, низько)

    Serial.print(b.bit1);//Серійний друк(1 біт)
    Serial.print(b.bit2); Серійний друк(2 біт)
    Serial.print(b.bit3); Серійний друк(3 біт)
    Serial.print(b.bit4); Серійний друк(4 біт)
    Serial.print(b.bit5); Серійний друк(5 біт)
    Serial.print(b.bit6); Серійний друк(6 біт)
    Serial.print(b.bit7); Серійний друк(7 біт)
    Serial.print(b.bit8); Серійний друк(8 біт)
    Serial.println();
   }
};

void setup() {//Вводится налоштування
  pinMode (4, OUTPUT);//Встановлюемо режим роботи- на виході чотири
  Serial.begin(115200);//Встановлюемо таймінг
  Serial.setDebugOutput(true);//
  Serial.println();
  camera_config_t config;//Конфігурація камери
  config.ledc_channel = LEDC_CHANNEL_0;//Дісплей каналів
  config.ledc_timer = LEDC_TIMER_0;//Дісплей таймера, нуль
  config.pin_d0 = Y2_GPIO_NUM;//Конфігурація піна
  config.pin_d1 = Y3_GPIO_NUM;// Конфігурація піна
  config.pin_d2 = Y4_GPIO_NUM; //Конфігурація піна
  config.pin_d3 = Y5_GPIO_NUM; //Конфігурація піна
  config.pin_d4 = Y6_GPIO_NUM; //Конфігурація піна
  config.pin_d5 = Y7_GPIO_NUM; //Конфігурація піна
  config.pin_d6 = Y8_GPIO_NUM; //Конфігурація піна
  config.pin_d7 = Y9_GPIO_NUM; //Конфігурація піна
  config.pin_xclk = XCLK_GPIO_NUM;//Конфігурація піна
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
```

```
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;//
config.pixel_format = PIXFORMAT_JPEG;//Формат зображення
//init with high specs to pre-allocate larger buffers //
if (psramFound()) {
  config.frame_size = FRAMESIZE_UXGA;//
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}

sensor_t * s = esp_camera_sensor_get();
//initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1);//flip it back
  s->set_brightness(s, 1);//up the blightness just a bit
  s->set_saturation(s, -2);//lower the saturation
}
//drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);

WiFi.begin(ssid, password); //ВайФай, логин, пароль

while (WiFi.status() != WL_CONNECTED) {
  delay(500);// ВайФай статус, подключение, ограничение 500
```

```
  Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");//ВайФай підключення

 startCameraServer();// запущена камера сервера

 Serial.print("Camera Ready! Use 'http://");// Камера готова, використовуйте домен
 Serial.print(WiFi.localIP());//ВайФай локальний IP
 Serial.println("' to connect");// підключення

 BLEDevice::init("QI_1");//Блютуз пристрій
 BLEServer *pServer = BLEDevice::createServer();//Блютуз сервер, створення сервера
 BLEService *pService = pServer->createService(SERVICE_UUID);//Створення сервера
 pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID,//Створення характеристики
BLECharacteristic::PROPERTY_READ
BLECharacteristic::PROPERTY_WRITE);

 pCharacteristic->setCallbacks(new BLECb());
 pService->start();
 BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
 pAdvertising->addServiceUUID(SERVICE_UUID);
 pAdvertising->setScanResponse(true);
 pAdvertising->setMinPreferred(0x06);
 pAdvertising->setMinPreferred(0x12);
 BLEDevice::startAdvertising();
}

void loop()
{
 delay(10000);//Затримка
}
```

ДОДАТОК В

Листинг кода Bluetooth- сервера

```
/*
 * BLEService.h
 *
 *  Created on: Mar 25, 2020
 *      Author: Shkrum
 */

#ifndef COMPONENTS_CPP_UTILS_BLESERVICE_H_
#define COMPONENTS_CPP_UTILS_BLESERVICE_H_
#include "sdkconfig.h"
#if defined(CONFIG_BT_ENABLED)

#include <esp_gatts_api.h>

#include "BLECharacteristic.h"
#include "BLEServer.h"
#include "BLEUUID.h"
#include "FreeRTOS.h"

class BLEServer;

/**
 * @brief A data mapping used to manage the set of %BLE characteristics kn
own to the server.
 */
class BLECharacteristicMap {
public:
   void setByUUID(BLECharacteristic* pCharacteristic, const char* uuid);
   void setByUUID(BLECharacteristic* pCharacteristic, BLEUUID uuid);
   void setByHandle(uint16_t handle, BLECharacteristic* pCharacteristic);
   BLECharacteristic* getByUUID(const char* uuid);
   BLECharacteristic* getByUUID(BLEUUID uuid);
   BLECharacteristic* getByHandle(uint16_t handle);
   BLECharacteristic* getFirst();
   BLECharacteristic* getNext();
```

```
   std::string toString();
   void handleGATTServerEvent(esp_gatts_cb_event_t event, esp_gatt_if_t g
atts_if, esp_ble_gatts_cb_param_t* param);

private:
   std::map<BLECharacteristic*, std::string> m_uuidMap;
   std::map<uint16_t, BLECharacteristic*> m_handleMap;
   std::map<BLECharacteristic*, std::string>::iterator m_iterator;
};

/**
 * @brief The model of a %BLE service.
 *
 */
class BLEService {
public:
   void              addCharacteristic(BLECharacteristic* pCharacteristic);
   BLECharacteristic* createCharacteristic(const char* uuid, uint32_t properti
es);
   BLECharacteristic* createCharacteristic(BLEUUID uuid, uint32_t properti
es);
   void              dump();
   void              executeCreate(BLEServer* pServer);
   void              executeDelete();
   BLECharacteristic* getCharacteristic(const char* uuid);
   BLECharacteristic* getCharacteristic(BLEUUID uuid);
   BLEUUID           getUUID();
   BLEServer*        getServer();
   void              start();
   void              stop();
   std::string       toString();
   uint16_t          getHandle();
   uint8_t           m_instId = 0;

private:
   BLEService(const char* uuid, uint16_t numHandles);
   BLEService(BLEUUID uuid, uint16_t numHandles);
   friend class BLEServer;
   friend class BLEServiceMap;
```

```
    friend class BLEDescriptor;
    friend class BLECharacteristic;
    friend class BLEDevice;

    BLECharacteristicMap m_characteristicMap;
    uint16_t             m_handle;
    BLECharacteristic*   m_lastCreatedCharacteristic = nullptr;
    BLEServer*           m_pServer = nullptr;
    BLEUUID              m_uuid;

    FreeRTOS::Semaphore  m_semaphoreCreateEvt = FreeRTOS::Semaphore(
"CreateEvt");
    FreeRTOS::Semaphore  m_semaphoreDeleteEvt = FreeRTOS::Semaphore(
"DeleteEvt");
    FreeRTOS::Semaphore  m_semaphoreStartEvt  = FreeRTOS::Semaphore("
StartEvt");
    FreeRTOS::Semaphore  m_semaphoreStopEvt   = FreeRTOS::Semaphore("
StopEvt");

    uint16_t             m_numHandles;

    BLECharacteristic* getLastCreatedCharacteristic();
    void handleGATTServerEvent(esp_gatts_cb_event_t event, esp_gatt_if_t g
atts_if, esp_ble_gatts_cb_param_t* param);
    void               setHandle(uint16_t handle);
    //void               setService(esp_gatt_srvc_id_t srvc_id);
}; // BLEService

#endif // CONFIG_BT_ENABLED
#endif /* COMPONENTS_CPP_UTILS_BLESERVICE_H_ */
```