КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«СИСТЕМА КЕРУВАННЯ "РОЗУМНИМ БУДИНКОМ" НА БАЗІ ESP32»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Аліна Квашенко _____

Науковий керівник:
кандидат фізико-математичних наук, асистент
Сергій Фесенко _____

Рецензент:
_____

Засвідчую, що у цій бакалаврській роботі немає запозичень з праць інших авторів без відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри
_____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Дипломна робота: 54 сторінки, 7 рисунків, 1 додаток, 11 таблиць, 9 джерел.

Мета роботи - проєктування розумного будинку, використовуючи платформу ESP32. Досягнення максимально можливої продуктивності та надійності системи. Мінімізація можливих проявів вразливостей автоматизованого будинку, що можуть виникнути при несумісності елементів як в програмній, так і апаратній частині проєкту.

Створення вебсторінки та мобільного додатку для дистанційного керування елементами будинку та відстеження показників датчиків. Дослідження методів передачі даних між мікроконтролером та електронним пристроєм користувача.

У першому розділі досліджено характеристики, принцип роботи, переваги та недоліки мікроконтролера ESP32 та електронних компонентів системи Smart House. Проведено порівняльну характеристику ESP32.

У другому розділі запрограмовано розумний будинок використовуючи мову C++. Створено вебсторінку та мобільний додаток, за допомогою яких користувачеві доступна можливість керування будинком та моніторинг показників температури, вологості та рівню CO_2 в повітрі.

Ключові слова: ESP32**,** Wi-Fi - модуль**,** мікросхема, електроніка, система, передача даних, вебсервер, мобільний додаток, AJAX, Node.js, React Native, GraphQL, датчик, ідентифікація, доступ.

ЗМІСТ

ВСТУП

Автоматизація технологічних процесів - це сукупність систем, що мінімізують участь людини в процесі виготовлення певної продукції. Вона відбувається шляхом поступового впровадження сучасних методів та обладнання. Найновіші технології оптимізують виробничі процеси та поліпшують якість товарів, що випускаються, при цьому можливість браку або допущення помилок людиною знижується до мінімуму.

Автоматизація може зачіпати як окремі технологічні процеси та елементи обладнання, так і бути основною системою діяльності, охоплюючи всі етапи виробництва.

Показники, що демонструють необхідність впровадження автономних технологій у робочий процес:

- зниження кількості робітників;
- мінімізація впливу людського фактору;
- підвищення обсягу товарних одиниць;
- зростання показників ефективності процесу виготовлення продукції;
- поліпшення якісних характеристик виробленого товару;
- зниження сировинних витрат;
- стабільне функціонування виробництва;
- усунення виконання задач людиною, що є фізично неможливі для неї або проводяться у критичних умовах;
- зменшення кількості відходів.

Очевидно, існують й недоліки автоматизації технічних процесів. Обладнання, що приймає участь у виробництві, часто має високу вартість і потребує систематичних перевірок та оновлень, а, отже, й фінансових затрат. Це необхідно не тільки для стабільності та надійності, а й для гарантування безпечних робочих умов для персоналу.

Варто згадати ще одну деталь – вразливість: автоматизована система може мати обмежений рівень інтелекту, тобто вона більш сприйнятлива до вчинення помилки за межами своєї безпосередньої сфери знань. Також, одним з недоліків може стати значне скорочення робочих місць для людей.

Сьогодні головна мета автоматизації змістилася в бік ширших питань, ніж вартість і час. Рішення щодо застосування систем управління приймаються все частіше, і не тільки в промисловості та у великих корпораціях.

Домашня автоматизація – це система пристроїв, застосування яких націлене на забезпечення захисту та розширення функціоналу будинку, оптимізацію побутових процесів, спрощення виконання повсякденних речей. Домашня автоматизація займається вивченням сучасних технологій для поліпшення якості життя людини.

За останні роки набула надзвичайної популярності концепція Smart House, де кожна конструкція є унікальною. Основними задачами при побудові є здатність автоматичного увімкнення світла та дистанційного відчинення дверей, здатність системи до кондиціонування й захисту оселі від крадіжки. Розумний будинок забезпечує не тільки комфорт та безпеку, а й краще енергозбереження.

РОЗДІЛ 1. АНАЛІЗ ESP32 ТА ОГЛЯД АПАРАТНОЇ ЧАСТИНИ ПРОЄКТУ

1.1. Загальний опис

ESP32 — це єдина комбінована Wi-Fi та Bluetooth мікросхема, що створена для досягнення високих показників потужності та радіочастотної продуктивності. Вона демонструє міцність, універсальність і надійність у різноманітних проєктах, при цьому, модуль потребує мінімального енергоспоживання. Апаратна частина IoT-платформи виконана на модулі ESP-WROOM-32 виробництва компанії Espressif. Мікросхема призначена для мобільної розробки та додатків Інтернету речей, є основою негабаритних електронних девайсів (wearable electronics). Побудова рішень на базі ESP32 не вимагає професійного обладнання.

ESP32 – це високоінтегроване рішення для IoT-додатків, що мають до 20 зовнішніх компонентів. Модуль містить антенний перемикач, підсилювач потужності, малошумний підсилювач, фільтри та модулі управління живленням. У ESP32 вбудовані вдосконалені калібрувальні схеми, що дозволяють усунути недоліки та швидко адаптуватися до зовнішніх змін. Для мінімізації кількості енергії, яку витрачає мікросхема, використовується низький робочий цикл. У ESP32 вбудовано чотири таймери загального призначення та 3 сторожових таймери.

1.2. Характеристики та принцип роботи

Основні характеристики Wi-Fi та Bluetooth

ESP32 підтримує протокол TCP/IP, Wi-Fi Multimedia (Wireless Multimedia Extensions) та стандарт Wi-Fi 802.11n. Швидкість передачі даних може досягати до 150 Мбіт/с.

У ESP32 впроваджено як класичний Bluetooth (Classic BT), так і Bluetooth Low Energy (BLE) версії 4.2. ESP32 Bluetooth надає три типи інтерфейсів хост-контролера (HCI): UART, SPI і VHCI (Virtual HCI), при цьому одночасно може використовуватися тільки один з них (за замовчуванням використовується UART).

Центральний процесор

ESP32 містить двоядерний процесор Xtensa 32-bit LX6 Dual Core 240 МГц з наступними характеристиками:

- 7-ступінчастий конвеєр для підтримки тактової частоти до 240 МГц;
- Підтримка Floating Point Unit;
- Підтримка інструкцій DSP, таких як 32-розрядний множник, 32-розрядний дільник і 40-розрядний MAC;
- Інтерфейс RAM/ROM Xtensa для команд і даних;
- Інтерфейс локальної пам'яті Xtensa для швидкого доступу до периферійних регістрів;
- Зовнішні та внутрішні джерела переривань.

Внутрішня пам'ять

ESP32 має внутрішню пам'ять, до якої входить:

- 448 Кбайт ПЗП (ROM) для функцій ядра та завантажень;
- 520 Кбайт SRAM для даних та команд;
- 8 Кбайт RTC FAST Memory, може бути використана для зберігання даних;
- 8 Кбайт RTC SLOW Memory, може бути доступна під час режиму глибокого сну;
- 1 Кбіт eFuse: 256 біт використовуються для системи (MAC-адреса і конфігурація мікросхеми), а решта 768 біт зарезервовані для клієнтських додатків, включаючи шифрування флеш-пам'яті;

- Вбудована флеш-пам’ять (4 Мбайт).

Зовнішня флеш-пам'ять і SRAM

ESP32 підтримує декілька зовнішніх мікросхем QSPI (Quad Serial Peripheral Interface) flash і SRAM. ESP32 також підтримує апаратне шифрування на основі AES для захисту програм і даних у флеш-пам'яті. ESP32 може отримати доступ до зовнішньої QSPI флеш-пам'яті і SRAM через високошвидкісний кеш. Підтримується SRAM об'ємом до 8 Мбайт.

Безпека

У ESP32 підтримується:

- Безпечне завантаження та перезавантаження;
- IEEE 802.11 безпека WFA, WPA/WPA2 (Wi-Fi protected access) та WAPI (WLAN Authentication and Privacy Infrastructure);
- Шифрування флеш-пам'яті;
- 1024-бітове OTP, до 768-бітове для клієнтів;
- Апаратне криптографічне прискорення: AES, SHA-2, RSA, ECC, RNG;
- 1024-bit OTP (one-time pad).

Характеристика пінів ESP32 на 38 контактів:

Назва	**Тип**	**Функції**
3V3	I/O	Напруга 3.3В
EN	I	Перезапуск ESP32
SP	I	GPIO36, ADC1_CH0, RTC_GPIO0
SN	I	GPIO39, ADC1_CH3, RTC_GPIO3
G34	I	GPIO34, ADC1_CH6, RTC_GPIO4
G35	I	GPIO35, ADC1_CH7, RTC_GPIO5
G32	I/O	GPIO32, ADC1_CH4, RTC_GPIO9, TOUCH9,

		32K_XP
G33	I/O	GPIO33, ADC1_CH5, RTC_GPIO8, TOUCH8, 32K_XN
G25	I/O	GPIO25, ADC2_CH8, RTC_GPIO6, DAC_1, EMAC_RXD0
G26	I/O	GPIO26, ADC2_CH9, RTC_GPIO7, DAC_2, EMAC_RXD1
G27	I/O	GPIO27, ADC2_CH7, RTC_GPIO17, TOUCH7, EMAC_RX_DV
G14	I/O	GPIO14, ADC2_CH6, RTC_GPIO16, TOUCH6, EMAC_TXD2, HSPICLK, HS2_CLK, SD_CLK, MTMS
G12	I/O	GPIO12, ADC2_CH5, RTC_GPIO15, TOUCH5, EMAC_TXD3, HSPIQ, HS2_DATA2, SD_DATA2, MTDI
GND	I/O	Заземлення (0В)
G13	I/O	GPIO13, ADC2_CH4, RTC_GPIO14, TOUCH4, EMAC_RX_ER, HSPID, HS2_DATA3, SD_DATA3, MTCK
SD2	I/O	GPIO9, HS1_DATA2, U1RXD, SD_DATA2, SPIHD
SD3	I/O	GPIO10, HS1_DATA3, U1TXD, SD_DATA3, SPIWP
CMD	I/O	GPIO11, HS1_CMD, U1RTS, SD_CMD, SPICS0
V5	I/O	Напруга 5В
G23	I/O	GPIO23, HS1_STROBE, VSPID
G22	I/O	GPIO22, U0RTS, VSPIWP, EMAC_TXD1
TXD	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
RXD	I/O	GPIO3, U0RXD, CLK_OUT2
G21	I/O	GPIO21, VSPIHD, EMAC_TX_EN
G19	I/O	GPIO19, U0CTS, VSPIQ, EMAC_TXD0

G18	I/O	GPIO18, HS1_DATA7, VSPICLK
G5	I/O	GPIO5, HS1_DATA6, VSPICS0, EMAC_RX_CLK
G17	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
G16	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
G4	I/O	GPIO4, ADC2_CH0, RTC_GPIO10, TOUCH0, EMAC_TX_ER, HSPIHD, HS2_DATA1, SD_DATA1
G0	I/O	GPIO0, ADC2_CH1, RTC_GPIO11, TOUCH1, EMAC_TX_CLK,CLK_OUT1,
G2	I/O	GPIO2, ADC2_CH2, RTC_GPIO12, TOUCH2, HSPIWP, HS2_DATA0, SD_DATA0
G15	I/O	GPIO15, ADC2_CH3, RTC_GPIO13, TOUCH3, EMAC_RXD3, HSPICS0, HS2_CMD, SD_CMD, MTDO
SD1	I/O	GPIO8, HS1_DATA1, U2CTS, SD_DATA1, SPID
SD0	I/O	GPIO7, HS1_DATA0, U2RTS, SD_DATA0, SPIQ
CLK	I/O	GPIO6, HS1_CLK, U1CTS, SD_CLK, SPICLK

Усі контакти ESP32 підтримують переривання. Максимальний струм на пінах: 12 мА. Аналоговий сигнал, що надходить до аналогового піну, має значення в діапазоні від «0» до «4095», де «0» – це 0 вольт, а «4095» – це 3.3 вольта.

Класифікація пінів загального призначення (GPIO):

- 21 цифровий контакт введення-виведення: 1-5, 12-19, 21-23, 25-27, 32 и 33. Логічний рівень одиниці - 3,3 В, нуля - 0 В.
- 4 цифрові контакти введення: 34, 35, 36 і 39. Можуть бути налаштовані тільки на вхід;

- 15 аналогових входів з 12-розрядним АЦП: 2, 4, 12-15, 25-27, 32-36 и 39. Дає змогу представити аналогову напругу в цифровому вигляді з розрядністю 12 біт;
- 2 аналогових виходи з 8-розрядним ЦАП: 25 (DAC1) і 26 (DAC2);
- Піни, призначені для роботи з внутрішньою флеш-пам'яттю (наприклад, при під'єднанні зовнішньої SD-карти): 6-11.

Технології та можливості пінів:

- LED та Motor PWM (16 каналів);
- 2xSPI: VSPI та HSPI;
- 2xADC: ADC1 та ADC2;
- TOUCH_X – сенсори дотику. Плата ESP32 має 10 контактів, які можуть використовуватися як touch-виводи, що здатні виявляти торкання рукою. У коді виводи починаються із символу T;
- DEEP SLEEP GPIO;
- Апаратна підтримка SD-карт;
- Підтримка протоколів/інтерфейсів I2C, DAC, SD, UART, CAN2.0, TX/RX;
- 1 host (SD/SDIO);
- 1 slave (SDIO/SPI).

1.3. Переваги та недоліки. Порівняльна характеристика

ESP32 VS ESP8266

ESP8266 (попередник ESP32) є бюджетним Wi-Fi модулем, що підходить для проєктів у сфері IoT й домашньої автоматизації. Головним недоліком модуля є недостатня кількість виводів, що обмежує застосування ESP8266 у реальних проєктах.

ESP32 має значну кількість удосконалень. Одна з них - більша кількість GPIO. Піни GPIO підтримують широкий спектр протоколів, такі як SPI, I2C, UART, ADC, DAC. Є можливість їх налаштування на програмному рівні, тобто, визначити, які з них будуть UART, I2C або SPI. Це можливо завдяки функції мультиплексування мікросхеми ESP32, яка дає змогу призначати кілька функцій одному й тому самому виводу. Крім того, ESP32 містить 10 ємнісних датчиків GPIO, які реагують на дотик і можуть використовуватися, наприклад, для запуску подій або виведення ESP32 з глибокого сну. У ESP32 також вбудовано датчик Холла й датчик температури.

Підсумуємо основні відмінності між ESP8266 та ESP32:

Характеристика	**ESP8266**	**ESP32**
Bluetooth	-	Bluetooth 4.2
SRAM	160 Кб	520 Кб
Частота процесора	160-240 МГц	80 МГц
GPIO	17	36
ШИМ	8 каналів	16 каналів
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
АЦП	10-бітний	12-бітний
Робоча температура	від -40 °C до 125 °C	від -40 °C до 125 °C
Ethernet MAC інтерфейс	-	1
Сенсор дотику	-	+
Датчик Холла	-	+
Датчик температури	-	+

Дані про робочу температуру обох пристроїв співпадають. Інформація взята з офіційного datasheet. Проте, заявлені мінімальне та максимальне значення температур є не робочими, а критичними. Рекомендований діапазон: від -15 до 60 градусів. Нестабільність або навіть неможливість роботи при

вищих або нижчих температурах пов'язані з фізичними процесами, що протікають у напівпровідниках.

Підсумовуючи характеристики обох плат, можна зробити висновок, що ESP32 істотно полегшує процес розробки завдяки численним вдосконаленням. Єдиним принциповим мінусом є ціна виробу, що зросла в порівнянні з попередником в середньому на 4$.

1.4. Опис електронних компонентів розумного будинку

1.4.1. Тактова кнопка

Кнопковий перемикач – найпростіший і найдоступніший з усіх видів датчиків. Механізм передає електричний сигнал іншим пристроям шляхом замикання або розмикання контактів. Натиснувши на нього, подається сигнал контролеру і, як наслідок, виникає певна дія, наприклад: вмикання світлодіоду, відтворення звуку, обертання певного рухомого механізму тощо.

Розділимо кнопки на дві групи:

- Кнопки перемикачі із фіксацією. Вони фіксуються у положенні, в якому їх залишили;
- Кнопки без фіксації (тактові). Вони повертаються у початковий стан після того, як їх відпустили.

При натисканні на кнопку відбувається замикання та розмикання пластин, як результат, можуть виникнути мікро-іскри, що провокують близько десятка перемикань за декілька мілісекунд. Це явище називається ефектом брязкоту, який необхідно враховувати, якщо поставлено задачу фіксувати або рахувати кожне натискання.

1.4.2. RGB світлодіод. Керування освітленням. ШІМ.

RGB-світлодіод складається з трьох одноколірних кристалів, що розташовані близько один до одного в одному корпусі. Назва RGB

розшифровується як: Red – червоний, Green – зелений, Blue – синій, де кожен кристал випромінює певний базовий колір.

Крім “ввімкнення” та “вимкнення” світлодіоду, є можливість зміни інтенсивності його кольору, що здійснюється шляхом регулювання яскравості випромінювання кожного із кристалів. У цих випадках застосовується ШІМ – сигнал (широтно-імпульсна модуляція). Принцип роботи полягає у періодичній зміні високого сигналу на низький, при цьому світлодіод періодично запалюється та гасне. Різне співвідношення часу ввімкненого та вимкненого стану світлодіоду сприймається оком людини як різна яскравість світіння. При використані цього способу регулювання потужності, втрати енергії мінімальні.

1.4.3. П’єзо-динамічний випромінювач

П’єзо-динамічний випромінювач (buzzer, зумер) - це компактний пристрій, який перетворює електричні коливання на звук.

П'єзодинаміки бувають активні та пасивні, ззовні вони схожі, але відрізняються за розмірами. Також, при підключенні пасивного п'єзодинаміку, не відтворюється жодного звуку, на відміну від активного. Звичайної подачі живлення не достатньо, потрібно використовувати програмні методи. Як правило, активний зумер пищить голосніше і виразніше за пасивний. Довга ніжка пристрою відповідає за напругу, коротка – заземлення.

Пристрої зустрічаються в модульному виконанні. У модуля три виводи, VCC відповідає за подачу напруги 3.3-5В, GND – заземлення, а вивід I/O з'єднується з цифровим піном контролюючої плати.

1.4.4. LCD дисплей

Дисплей — це електронний пристрій, призначений для візуального відображення інформації (текстової, графічної тощо). У даному проєкті використовується саме LCD-дисплей. Розглянемо детальніше.

LCD – рідкокристалічний дисплей, є типом плоского екрана, що набув широкого використання. Для отримання зображення LCD-екрани використовують рідкокристалічну матрицю із заднім підсвічуванням. При розробці LCD-дисплея використовується особлива речовина ціанофеніл. Вона, як кристал, може пропускати світловий фільтр.

Варто відмітити, що у LCD-технологію впроваджено відсутність мерехтіння й відносно невелике енергоспоживання. Подібний тип дисплеїв забезпечує відмінну передачу кольору, але, в порівнянні з конкурентами, характеризується низькою контрастністю. Крім того, LCD-екрани не здатні забезпечити відображення натурального чорного кольору, а при яскравому сонячному світлі картинка на них здається бляклою та розмитою.

1.4.5. Сервопривід (сервомотор)

Сервопривід - це механізм з електромотором з керуванням. Система має кілька складових частин. Привід - електромотор із редуктором. Часто швидкість обертання мотора буває занадто великою для практичного використання, тому, для зниження швидкості використовується редуктор. Вмикаючи і вимикаючи електромотор, можна обертати вихідний вал - кінцеву шестерню сервомотору.

Сервомотор бувають аналогові та цифрові. Розрізняються вони внутрішньою керуючою електронікою, а отже в способі обробки імпульсів і управлінні мотором.

Пристрій має 3 дроти:

- Червоний відповідає за живлення мотора, підключається до контакту 3.3-5V;
- Коричневий або чорний відповідає за заземлення;
- Жовтий або білий – сигнал, кут повороту мотору, підключається до цифрового виходу. Значення кута повороту встановлюється за допомогою ШІМ-сигналу.

1.4.6. Датчик температури та вологості DHT11

DHT11 – це цифровий датчик вологості та температури, що складається з термістора (застосовується в мікроелектроніці для контролю температури) та датчика вологості. Містить у собі АЦП, що слугує для перетворення аналогових значень вологості та температури. DHT11 не притаманні швидкодія та надзвичайна точність, однак, датчик достатньо простий у використанні, він підходить для контролю вологості та температури в невеликому приміщенні, йому притаманна довготривала стабільність роботи.

Характеристики DHT11:

- 3 контакти (GND, DATA, VCC);
- Живлення: 3,5 - 5,5 В;
- Діапазон визначення температури: 0 – 50° С;
- Похибка визначення температури: ±2° С;
- Діапазон визначення вологості: 20 – 90%;
- Похибка визначення вологості: 5%.

1.4.7. Датчик газу MQ-135

MQ-135 Gas Sensor - це простий та зручний у використанні датчик концентрації вуглекислого газу, який широко застосовують у робототехніці та системах автоматизації. Робоча напруга: 2.5-5 В. Чутливий до аміаку, оксиду азоту, алкоголю, бензолу, диму, вуглекислого газу.

Характеристики:

- Вихідна напруга залежить від концентрації вимірюваних газів;
- Швидка реакція і відновлення;
- Регульована чутливість;
- Напруга живлення нагрівача: 5 В;
- Напруга живлення датчика: 3,3-5 В;
- Струм, що споживається: 150 мА.

Датчик видає аналоговий сигнал, що пропорційний концентрації вуглекислого газу в одиницях виміру ppm (рівень концентрації газу в мільйонних частках). У газоаналізатор вбудовано нагрівальний елемент, який необхідний для хімічної реакції, що сприяє визначенню рівня забруднення повітря. Перед першим використанням рекомендується подати на датчик напругу і залишити в робочому стані на 1-2 доби. Ця процедура збільшить точність показань сенсора.

1.4.8. ІЧ датчик перешкоди YL-63

Інфрачервоне випромінювання - це електромагнітне випромінювання, яке випускає будь-яке нагріте до певної температури тіло.

Інфрачервоний датчик перешкод YL-63 – це цифровий пристрій, що застосовується, коли потрібно визначити наявність об'єкта, але точну відстань до нього знати необов'язково. Датчик складається з інфрачервоного випромінювача (білого кольору) і фотоприймача (чорного кольору).

ІЧ джерело випромінює ІЧ хвилі, які відбиваються від перешкоди та фіксуються фотоприймачем. Датчик побудований на основі компаратора LM393, який видає напругу на вихід за принципом: виявлено перешкоду - логічний рівень 1, не виявлено - логічний рівень - 0. Модуль має 3 виводи: VCC - живлення 3.3-5 В, GND – заземлення та OUT - цифровий вихід.

Характеристики:

- напруга живлення: 3.3-5 В;
- відстань виявлення перешкоди: 2 - 30 см;
- ефективний кут виявлення перешкоди: 35°;
- потенціометр для зміни чутливості;
- світлодіод індикації живлення;
- світлодіод індикації спрацьовування.

1.4.9. Система RFID

RFID (Radio Frequency Identification) – це метод автоматичної ідентифікації через радіосигнал. Він використовується в багатьох випадках безконтактного входу і доступу. RFID-система складається зі зчитувача, міток і програмного забезпечення. Мітка – це мікросхема, в якій зберігаються дані, а також антена для бездротової передачі інформації. Зовнішній зчитувач сканує пам'ять мітки та обробляє отримані дані.

Технологія має досить простий механізм роботи:

1. Інформація записується за допомогою радіохвиль;
2. Дані надходять на зчитувач за допомогою радіосигналу вбудованої антени;
3. Визначення випромінюваної частоти, налаштування і зчитування відомостей здійснюється автоматично завдяки сканеру.

Характеристики RFID-зчитувача RC522:

- Живлення 3,3 В;
- Дальність зчитування до 6 см;
- Призначений для зчитування та запису міток з частотою 13,56 МГц (діапазон HF).

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ПРОГРАМУВАННЯ СИСТЕМИ SMART HOUSE

2.1. Методи побудови розумного будинку

Для початку, необхідно зазначити, що розумний будинок – це модульна система, що передбачає інтеграцію численних під-систем, підбір яких залежить від бажань та потреб замовника. Незважаючи на різноманітність приладів і пристроїв, що впроваджуються у комплекс, автоматизовані системи можна класифікувати таким чином:

- Дротові (провідна система)
- Бездротові (безпровідна система)
- Централізовані
- Децентралізовані

Дротова система

При використанні даного методу побудови, всі керуючі системи пов'язані між собою загальною дротовою шиною, якою транслюються керуючі сигнали від датчиків до виконавчих пристроїв і навпаки. Для трансляції застосовують сигнальні кабелі, найчастіше використовують кручену пару.

Бездротова система

У подібній системі, керуючі сигнали надходять не по дротах, а по радіоканалу. Таке рішення значно скорочує час на монтаж системи загалом. Використання таких перемикачів допомагає розширити ряд завдань, які виконує будинок.

Централізована система

У випадку централізованості, налаштування та програмування системи здійснюється з єдиного модуля. Як правило, це програмований контролер, що здійснює управління приладами і пристроями, встановленими в будинку. Централізовані системи можуть бути побудовані на підставі або дротових, або бездротових мереж.

Децентралізована система

Слово "розподілений" описує даний метод побудови. Воно означає, що система не будується на одному центральному комп'ютері. Кожен пристрій, що використовується, оснащений мікропроцесором з автономною пам'яттю. Дотримуючись децентралізованості, вихід з ладу головного процесора не призведе до некерованості будинку. Тобто, розумні будинку розподіленої архітектури мають певну кількість вузлів, що зв'язані в загальну керовану мережу. Вузли мають доступ один до одного та обмінюються інформацією між собою. Це підвищує гнучкість та легкість масштабування. Даний метод ідеально підходить для проєктів великих корпорацій, що зосередженні на забезпеченні максимальної надійності та ефективності, яку вимагають клієнти.

У моєму випадку, Smart House на базі ESP32 є зменшеною та спрощеною версією справжнього будинку, він не пристосований до реальних умов та щоденних рутинних обов’язків. Розробка такого типу проєкту не потребує розподіленості, тому всі електронні компоненти підпорядковуються одному контролеру, що є мозком системи. Отже, результатом побудови є дротова централізована система.

2.2. Функціонал розумного будинку

Smart House реагує на певні події та безперестанно оновлює дані. Інформація передається з серверу та відображається на клієнтах вебдодатку та мобільного додатку.

Перелік електронних компонентів розумного будинку:

1. Датчик температури та вологості. Призначення: вимірювання температури та рівня вологості в кімнаті.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
DATA	GPIO4	Номер піну, до якого під’єднаний датчик
VCC	V5	Напруга 5В

2. Датчик газу. Призначення: вимірювання рівню газу в кімнаті.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
D0	GPIO33	Номер піну, до якого під’єднаний датчик
VCC	V5	Напруга 5В

3. Інфрачервоний датчик присутності. Призначення: фіксування руху.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
OUT	GPIO35	Номер піну, до якого під’єднаний датчик
VCC	V5	Напруга 5В

4. Тактова кнопка. Призначення: керування світлодіодом.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
OUT	GPIO15	Номер піну, до якого під’єднана кнопка
VCC	V5	Напруга 5В

5. RGB-світлодіод. Призначення: індикація, освітлення.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення

R	GPIO12	Номер піну, до якого під’єднаний контакт, що відповідає за червоний колір
G	GPIO13	Номер піну, до якого під’єднаний контакт, що відповідає за зелений колір
B	GPIO14	Номер піну, до якого під’єднаний контакт, що відповідає за синій колір

6. П’єзоелемент. Призначення: звуковий сигнал, сигналізація.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
DATA	GPIO32	Номер піну, до якого під’єднаний датчик
VCC	V5	Напруга 5В

7. RFID-зчитувач. Призначення: Зчитування UID картки при вході в будинок.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
SDA (SS) pin	GPIO5	Slave Select інтерфейсу SPI
SCK pin	GPIO18	Шина тактування інтерфейсу SPI

MOSI	GPIO23	Master Out, Slave In інтерфейсу SPI
MISO	GPIO19	Master In, Slave Out інтерфейсу SPI
IRQ	-	-
GND	GND	Заземлення
RST	GPIO27	Reset
3.3V	3.3V	Напруга 3.3В

8. LCD-дисплей. Призначення: відображення текстової інформації.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
VCC	V5	Напруга 5V
SDA	GPIO21	Канал даних інтерфейсу I2C
SCL	GPIO22	Канал синхронізації інтерфейсу I2C

9. Сервопривід. Призначення: рух вхідної двері.

Підключення:

Назва піну RC522	**ESP32**	**Призначення**
GND	GND	Заземлення
DATA	GPIO2	Номер піну, до якого під’єднаний сервомотор
VCC	V5	Напруга 5V

На рис. 1 зображено схему розумного будинку, а саме вигляд зверху. Основою будинку є велика та стандартна макетні плати.

На рис. 2, рис. 3 та рис. 4 зображено кінцеву версію розумного будинку, фотографії зверху, зліва та справа відповідно.

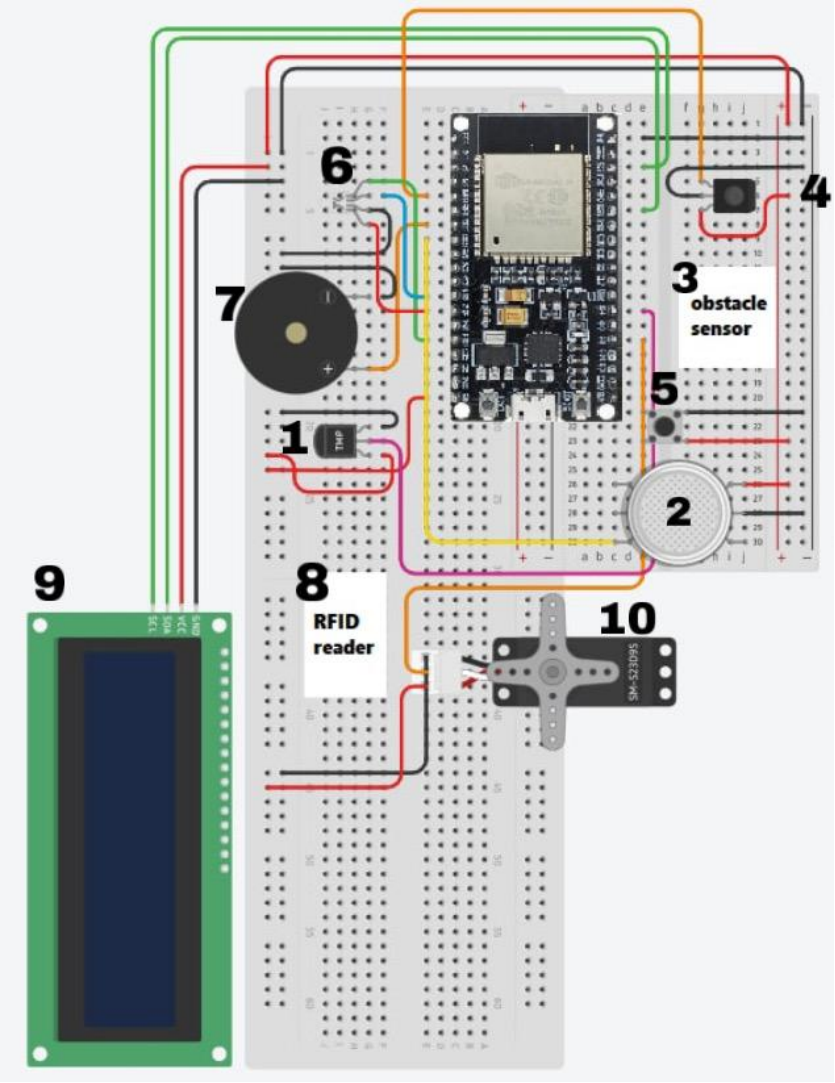

Рис. 1. Схема Smart House

Функціонал розумного будинку

1. Вхід до будинку. Ідентифікація за картою.

При зчитуванні мітки модулем RFID-RC522 (детальніше про нього у пункті 2.3.1), програма визначає UID карти та порівнює її з UID, прописаним у коді. Якщо картка підходить, користувачу надається можливість зайти в будинок. Кут рухомого елементу сервомотору повертається на 90 градусів. На LCD-дисплеї привітливо виводиться надпис “Welcome home!”. Якщо UID не вірний, кут залишається у стані 0 градусів. На LCD-дисплеї виводиться надпис “Who are you?”.

При цьому, колір світлодіоду залежить від наступних ситуацій:

a. Користувачеві відмовлено у доступі. Стан світлодіоду – червоний колір. Вимикається через 5 секунд;
b. Користувачеві дозволено увійти. Стан світлодіоду – зелений колір. Через 5 секунд колір змінюється на білий;

2. Вимірювання показників датчиками:
   - Показник температури та вологості;
   - Показник рівня CO_2 в будинку.

Обидва датчики (температури/вологості та рівню CO_2) регулярно оновлюють дані. Інформація перенаправляється на вебсервер та сервер мобільного додатку, де за допомогою вебсторінки та мобільного додатку користувач має можливість спостерігати за зміною показників. Детальніше про вебсторінку в пункті 2.4. Детальніше про мобільний додаток у пункті 2.5.

3. Інші функції електронних компонентів:
   - Перемикання стану світлодіоду тактовою кнопкою.

Після пропуску користувача до будинку, світлодіод набуває зеленого кольору, а через 5 секунд – білого. Для наступних вимкнень/увімкнень освітлення використовується чорна кнопка.

- Сигналізація при фіксуванні руху.

До схеми входить ІЧ датчик перешкоди, функція якого – визначати рух певного об’єкта. У програмі створено лічильник, який рахує кількість проходжень об’єкта перед датчиком. При визначенні лічильником трьох проходжень – активується сигналізація, п’єзодинамік видає звук 5 секунд.

На фото зображено кінцеву версію розумного будинку:

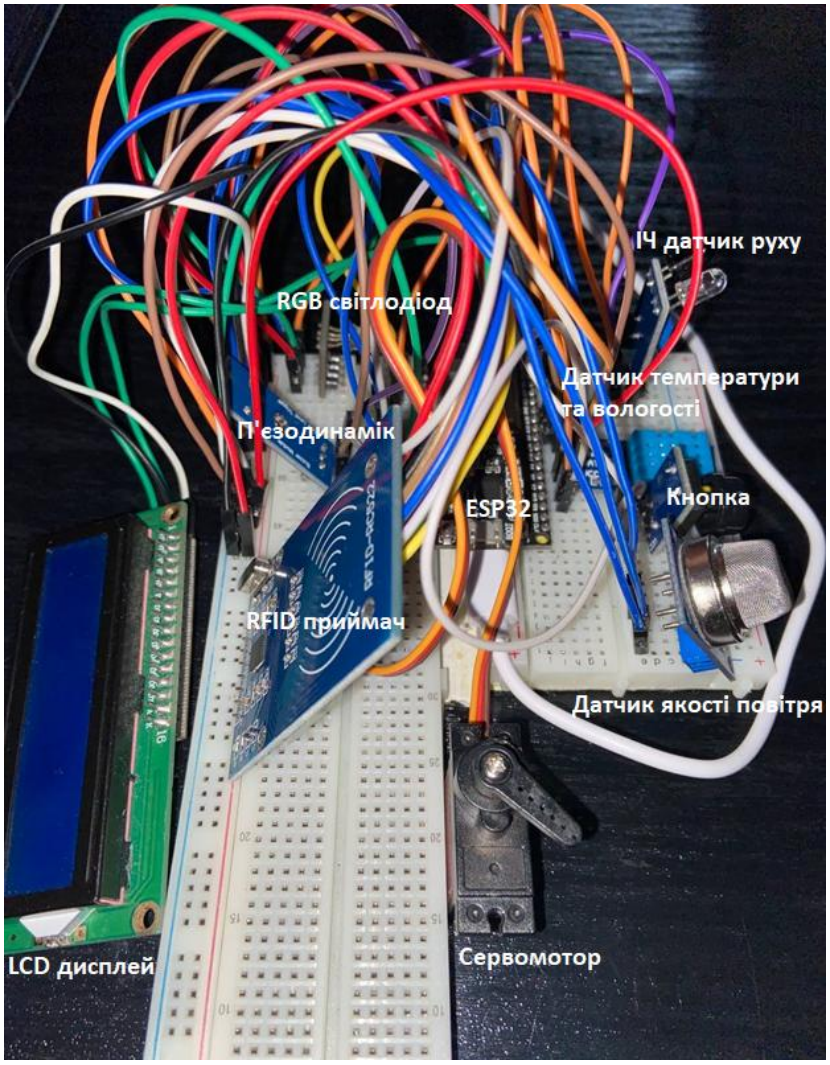

Рис. 2. Зовнішній вигляд системи зверху

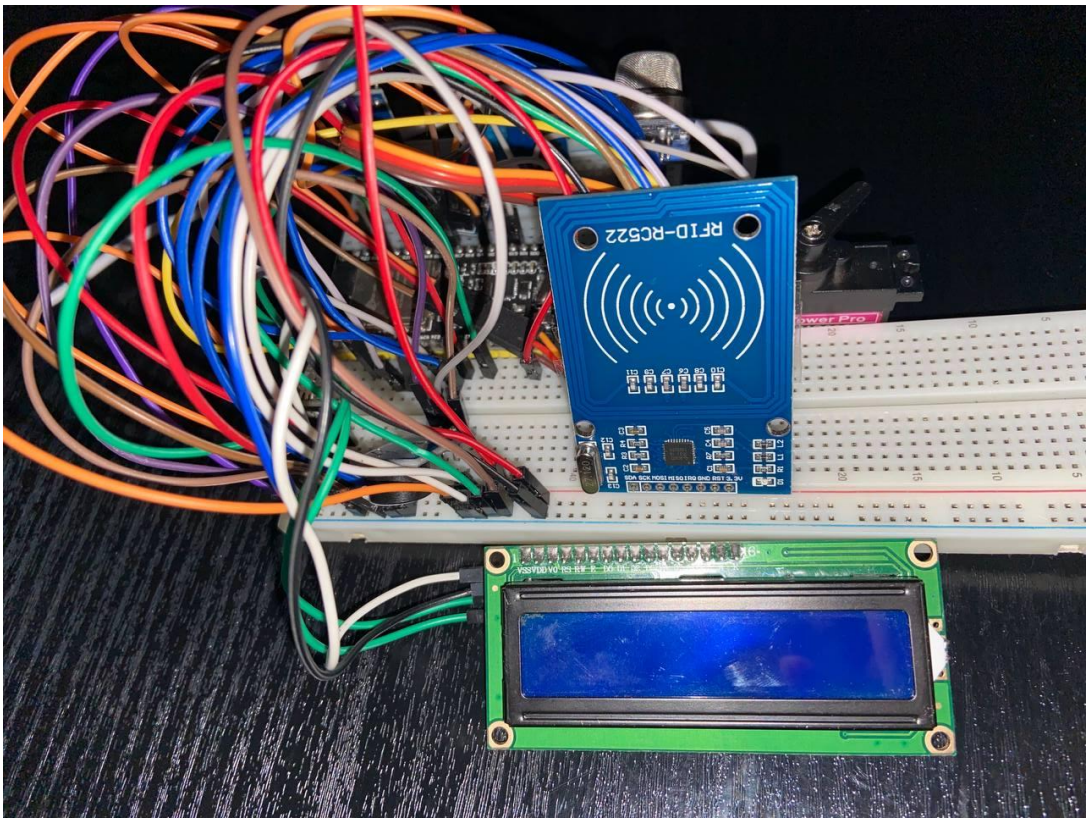

Рис. 3. Зовнішній вигляд системи зліва

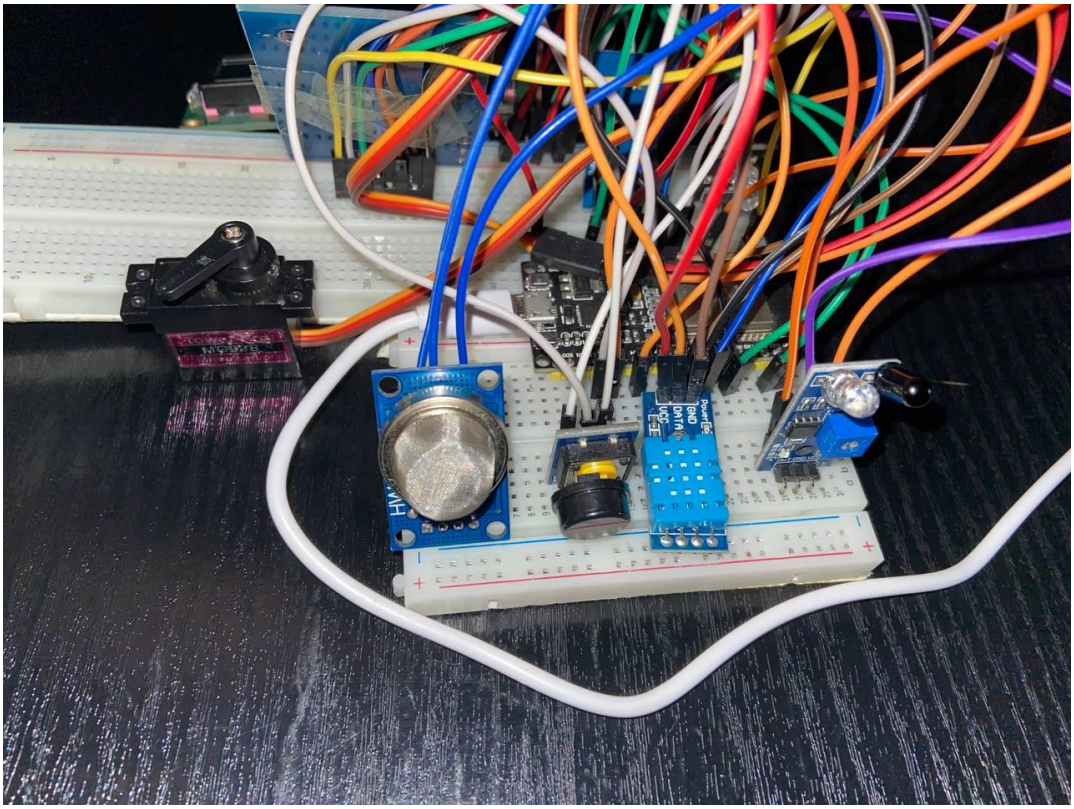

Рис. 4. Зовнішній вигляд системи справа

2.3. Пропуск користувача до будинку. RFID

Найперше, що необхідно зробити для отримання доступу до функцій розумного будинку – увійти до нього. Для пропуску користувача використовується RFID модуль RC522 на базі чипу MFRC522 та пасивна мітка (не має вбудованого джерела живлення).

Smart House моєї розробки має пасивну систему RFID, в якій використовується картка без внутрішнього джерела живлення. Живлення відбувається завдяки електромагнітної енергії, яку передає зчитувач RFID.

Для коректного зчитування UID картки та пропуску користувача до будинку необхідно прописати в програмі наступні моменти:

1. Визначення піну SDA(SS) RFID-зчитувача та присвоєння йому значення 5 (GPIO5). Визначення піну RST RFID-зчитувача та присвоєння йому значення 27 (GPIO27).
2. Ініціалізація змінної для зберігання значення UID потрібної картки.
3. Створення об’єкту RFID-зчитувача.
4. Ініціалізація MFRC522.
5. Додавання умови, при якій зчитувач не знаходить картку, спробувати зчитати знову. Пошук картки або мітки.
6. Запис UID зчитаної картки в змінну, значення якої надалі порівнюватиметься з ID потрібної картки.
7. Порівняння UID зчитаної картки зі змінною, де зберігається ID потрібної картки. Якщо значення співпадають, система пропускає користувача. В іншому випадку, система не дозволяє дверям розумного будинку відкритися.
8. Завершення сканування картки. Завершення роботи зі зчитувачем.

2.4. Дослідження методів передачі даних. Створення вебсервера

Одна з найголовніших деталей розумного будинку – коректний обмін інформацією. Для того, щоб регулярно отримувати оновлені дані від датчиків та мати можливість змінювати стан компонентів розумного будинку дистанційно, необхідно обрати метод передачі інформації.

Технології передачі даних ESP32

Мікроконтролер ESP32 підтримує бездротову передачу за допомогою технологій Bluetooth та WiFi. Обидва методи використовують радіосигнали для забезпечення з'єднання між електронними приладами.

При надсиланні або отриманні пакетів BT ESP32 не може прослуховувати або надсилати пакет WiFi. Причина: Wi-Fi та Bluetooth сигнали мають однакову частоту 2,4 ГГц та заважають один одному при паралельній роботі. Також, ці технології зазвичай мають різні призначення. Bluetooth використовується для підключення пристроїв малого діапазону для обміну даними, а Wifi забезпечує високошвидкісний доступ до Інтернету та надає доступ до більшої кількості користувачів та підключень.

Так як для передачі даних з ESP32 на вебсторінку необхідно використовувати технологію WiFi, її ж буде використано і для передачі інформації між ESP32 сервером та мобільним додатком для підтримання з’єднання без конфлікту радіохвиль WiFi та BT.

Архітектура клієнт-сервер

Розглянемо приклад, де користувачу, що виступає у ролі клієнта, потрібно змінити освітлення в будинку. Для дистанційного керування створимо вебсервер. Візуальна частина – вебсторінка (рис. 5) – є провідником між користувачем та ESP32. Для ввімкнення освітлення необхідно активувати перемикач біля пункту Lighting. Після натискання на кнопку підтвердження, клієнт надсилає запит (request) на сервер. Коли сервер отримає та виконає запит, клієнт, у свою чергу, отримає відповідь (response), про успішний (або ні) результат та побачить, що RGB світлодіод тепер світиться білим кольором.

Користувач може змінювати стан електронних компонентів використовуючи вебсторінку (рис. 5). Також, за допомогою простого та зрозумілого інтерфейсу сторінки, можна спостерігати за зміною показників температури, вологості та якості повітря.

Етапи створення вебсерверу:

1. Імпорт необхідних бібліотек.
2. Оголошення константи – назва мережі Wi-Fi. Оголошення константи – пароль.
3. Призначення порту прослуховування 80, що є TCP-портом, який використовується для незашифрованого трафіку HTTP.
4. Початок роботи із заданою мережею.
5. Створення кореневого шляху /. Запис статус запиту – 200(OK), типу сторінки - "text/html", контенту сторінки – масив index_html, та допоміжної функції, яка виконується при завантаженні HTML-сторінки.
6. Запуск та початок роботи вебсерверу.

Інформація зчитується з датчиків, після чого вона передається з ESP32 за допомогою технології AJAX (Asynchronous JavaScript and XML) та відображується на вебсторінці.

Отримання даних про рівень CO_2 в повітрі, виконання GET-запиту:

1. Створюється затримка в 10000 мілісекунд, тобто, дані, що надходять з датчиків, оновлюватимуться на сторінці кожні 10 секунд.
2. Для відправки GET-запиту використовується технологія AJAX та створюється новий XMLHttp запит.
3. Необхідно перевірити, чи статус відправки пакету даних дорівнює значенню “200”, а стан завершення відправки дорівнює значенню “4”.
4. Далі, перевіривши потрібні умови, змінюється текстове значення тегу, що відображає значення рівню CO_2 в повітрі.
5. Після цього, дані надсилаються на відповідну URL-адресу, а саме [‘ip-адреса веб-серверу’/air].

Відправка даних про стан світлодіоду:

1. Для відправки стану світлодіоду з вебсторінки на сервер використовується технологія AJAX.
2. Створюються новий XMLHttp запит.
3. Додається умова, що перевіряє, чи вибраний checkbox, що відповідає RGB світлодіоду. При виконанні умови, до URL-адреси додається параметр зі значенням 1. При невиконанні умови, до URL-адреси додається параметр зі значенням 0.
4. Дані надсилаються на відповідну URL-адресу, а саме [‘ip-адреса веб-серверу’/LEDweb?state=1] або [‘ip-адреса веб-серверу’/LEDweb?state=0].
5. При успішній відправці даних на сервер, у консоль виводиться повідомлення про завершення процесу.

Створено інтерфейс користувача для можливості виконання наступних дій:

1. Відслідковування показників температури, вологості та рівню CO_2 в повітрі;
2. Зміна режиму світлодіоду. Вмикання та вимикання освітлення;
3. Відкривання та закривання дверей розумного будинку. При відкриванні - кут сервомотору стає 90 градусів, при закриванні – 0 градусів;
4. Встановлення режиму “Вогонь у будинку” за допомогою перемикача, при якому спрацьовує сигналізація.

Для відправки оновлених даних на сервер необхідно натиснути кнопку “Send data”.

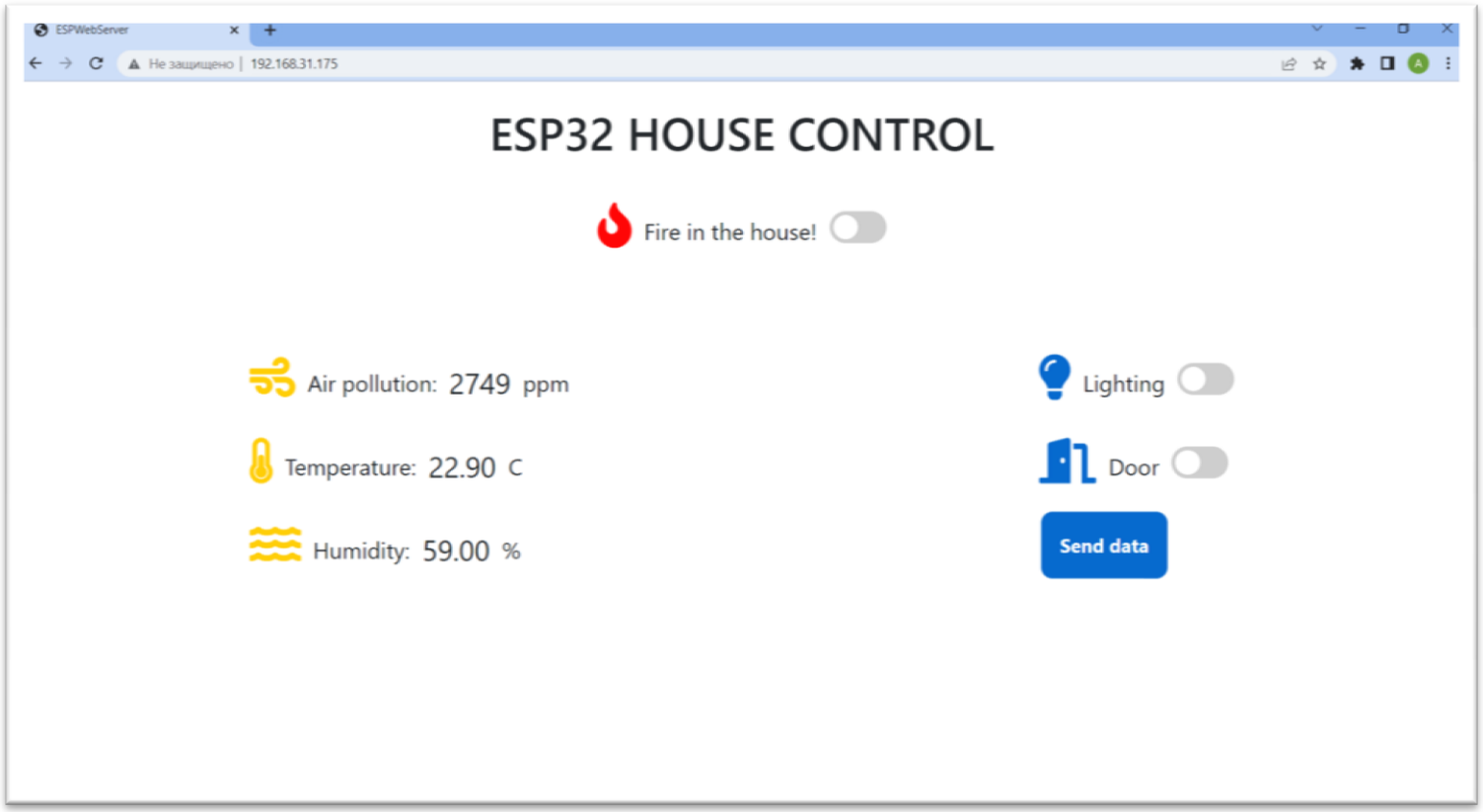

Рис. 5. Результат розробки вебсторінки

2.5. Створення мобільного додатку.

Для моніторингу інформації з датчиків та можливості керуванням освітленням та сервомотором створено мобільний додаток.

Клієнтська частина реалізована, використовуючи платформу Node.js та фреймворк React Native. Додаток складається з двох сторінок: головної (рис. 6) та інформаційної (рис. 7). На головній сторінці користувачу необхідно авторизуватися за допомогою відбитку пальця або паролю, який надається клієнту (власнику будинку) при оформленні документів на оренду або купівлю будинку. При коректному введені даних користувач може перейти на інформаційну сторінку за допомогою кнопки “GO TO DATA”, на якій відображаються показники температури, вологості та забруднення повітря.

Для повернення на головну сторінку потрібно натиснути кнопку “GO TO HOME”.

Серверна частина реалізована, використовуючи платформу Node.js та фреймворк Express.js. Для передачі даних між клієнтом та сервером використано мову запиту даних GraphQL. Серверна частина мобільного додатку надсилає та отримує дані з сервера ESP32.

1. Отримання даних з серверу ESP32
    - Відбувається за допомогою команди fetch, GET запит.
    - Дані з датчиків постійно оновлюються.
    - Сервер ESP32 надсилає оновленні дані на сервер мобільного додатку кожні 5 секунд, після чого дані транслюються на клієнті.
2. Надсилання даних на сервер ESP32
    - Відбувається за допомогою команди fetch, POST запит.
    - При зміні користувачем станом перемикачів світлодіоду та сервомотору оновленні дані надсилаються з клієнта на сервер.
    - Сервер мобільного додатку надсилає дані на сервер ESP32, після чого оновлюється стан освітлення або вхідних дверей.

Створено інтерфейс користувача для можливості виконання наступних дій:

1. Відслідковування показників температури, вологості та рівню CO_2 в повітрі;
2. Зміна режиму світлодіоду. Вмикання та вимикання освітлення;
3. Відкривання та закривання дверей розумного будинку. При відкриванні - кут сервомотору набуває значення 90 градусів, при закриванні – 0 градусів;

Мобільний додаток:

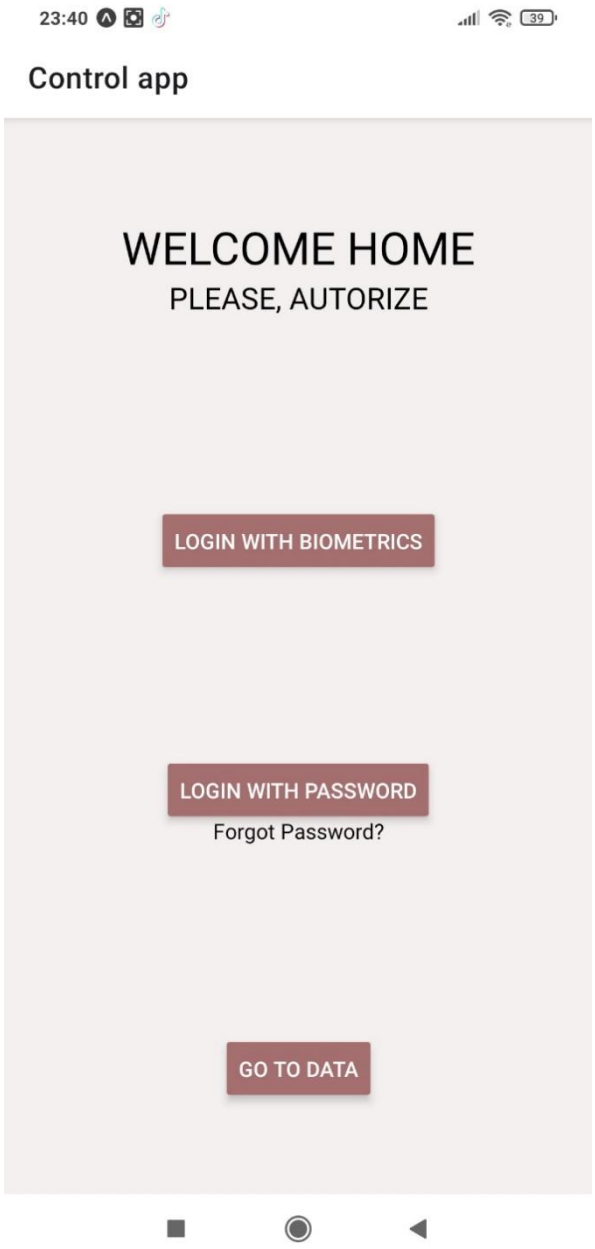

рис. 6. Головна сторінка

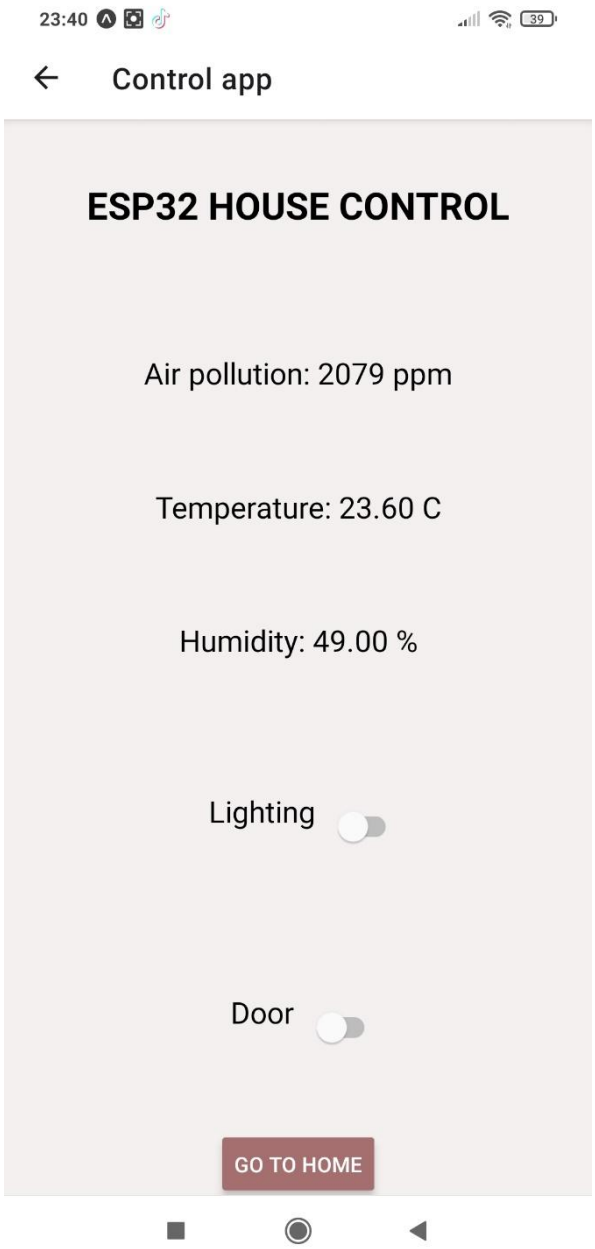

рис. 7. Інформаційна сторінка

ВИСНОВКИ

У результаті виконання випускної кваліфікаційної роботи було спроектовано та побудовано розумний будинок на базі ESP32. Використано наступні електричні компоненти та деталі: тактова кнопка, RGB-світлодіод, п’єзо-динамічний випромінювач, LCD-дисплей, сервомотор, датчик температури та вологості, датчик газу, ІЧ датчик перешкоди, система RFID, дві макетні плати та мікросхема ESP32. Опрацьовано роботу електронних компонентів системи та їх взаємодію.

Проведено огляд літератури у сфері автоматизації технологічних процесів та домашньої автоматизації. Досліджено методи побудови будівель концепції Smart House, визначено переваги та недоліки впровадження автоматизованих систем у реальному житті.

Запрограмовано розумний будинок використовуючи мову C++. Створено вебсервер, де за допомогою вебсторінки користувач отримує доступ до контролю будинком та має можливість спостерігати за показниками температури, вологості та рівню вуглекислого газу в повітрі. Для обміну даними використовується мова JavaScript. Технологія AJAX дозволяє отримувати данні з сервера та відображати їх на вебсторінці використовуючи GET-запит.

Створено мобільний додаток, що слугує для відслідковування показників датчиків та керування освітленням та дверима розумного будинку. Використано платформу Node.js та фреймворк React Native для клієнтської частини та платформу Node.js та фреймворк Express.js для серверної частини. Обмін інформацією між клієнтом та сервером відбувається через мову запитів даних GraphQL. Досліджено переваги використання технології WiFi для передачі даних та використання технології Bluetooth.

СПИСОК ОПРАЦЬОВАНОЇ ЛІТЕРАТУРИ

1. Sever Spanulescu. Esp32 Programming for the Internet of Things – 2018 p.
2. Getting Started with ESPHome – 2021 p.
3. Вольфганг Беєр. Open Source Home Automation – 2021 p.
4. ESP32 Datasheet (PDF) - ESPRESSIF SYSTEMS. [Електронний ресурс] – Доступ до ресурсу: https://pdf1.alldatasheet.com/datasheet-pdf/view/1148023/ESPRESSIF/ESP32.html
5. Espressif Systems. [Електронний ресурс] – Доступ до ресурсу: https://www.espressif.com/
6. Ajax Documentation. [Електронний ресурс] – Доступ до ресурсу: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
7. Font Awesome. [Електронний ресурс] – Доступ до ресурсу: https://fontawesome.com/
8. Platformio. [Електронний ресурс] – Доступ до ресурсу: https://platformio.org/
9. React Native documentation. [Електронний ресурс] – Доступ до ресурсу: https://reactnative.dev/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
#include <Arduino.h>
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include "ESPAsyncWebServer.h"
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <ESP32Servo.h>
#include <LiquidCrystal_I2C.h>
#include "EasyBuzzer.h"
#include <IRremote.h>

#define SS_PIN  5
#define RST_PIN 27
#define DHTTYPE DHT11
#define DHT_PIN 4 //Пін датчка темп. та волог.
#define PINA 33 //Пін датчика газу
#define SERVO_PIN 2 //Пін сервоприводу
#define BUZZER_PIN 32 //Пін п'єзодинаміка
#define IR_PIN 35 //Пін ІЧ сенсоруу
#define BTN_PIN 15 //Пін чорної кнопки
#define DIST 34 //Пін датчика перешкоди

const char *ssid = "GeyFell"; //Назва мережі Wi-Fi
const char *password = "multicultural41"; //пароль мережі

const char* PARAM_INPUT_1 = "output";
const char* PARAM_INPUT_2 = "state";
```

```
unsigned long uidDec, uidDecTemp;
long UID = 4141495794;
int counter = 0; int freq = 0; int light = 0; int ggwp = 0;
int flagIR = 0;
byte uID[4];
String S = "";
int IRarr [4];
bool flagBtn = false;

DHT dht(DHT_PIN, DHTTYPE);
Servo servo; //Ініціалізація сервоприводу
LiquidCrystal_I2C lcd(0x27, 16, 2); //Ініціалізація LCD-дисплею
MFRC522 rfid(SS_PIN, RST_PIN); //Ініціалізація MFRC522
AsyncWebServer server(80); //Призначення вебсерверу порт прослуховування 80

// HTML-сторінка
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>ESPWebServer</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="Refresh" content="2" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
```

```
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG
9Sr" crossorigin="anonymous">
 </head>
 <body>

  <style>
   html {
   font-family: Arial;
   display: inline-block;
   margin: 0px auto;
   text-align: center;
   }
   h2 { font-size: 3.0rem; }
   p { font-size: 3.0rem; }
   .units { font-size: 1.2rem; }
   .dht-labels{
    font-size: 1.5rem;
    vertical-align: middle;
    padding-bottom: 15px;
   }

   #h2{
    display: flex;
    flex-direction: column;
    align-items: center;
   }
   .Main{
    display: flex;
```

```
  justify-content: space-around;
}

.switch {
  position: relative;
  display: inline-block;
  vertical-align: middle;
  width: 60px;
  height: 34px;
}

/* Hide default HTML checkbox */
.switch input {
  opacity: 0;
  width: 0;
  height: 0;
}

.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ccc;
  -webkit-transition: .4s;
  transition: .4s;
}
```

```
.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}
.validate {
  width: 120px;
  height: 40px;
  vertical-align: middle;
  text-align: center;
  text-overflow: ellipsis;
}

input:checked + .slider {
  background-color: #0066cc;
}

input:focus + .slider {
  box-shadow: 0 0 1px #0066cc;
}

input:checked + .slider:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
```

```
  transform: translateX(26px);
 }

 /* Rounded sliders */
 .slider.round {
  border-radius: 34px;
 }

 .slider.round:before {
  border-radius: 50px;
 }

 .button {
    background-color: #0066cc;
    border: none;
    color: white;
    padding: 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 20px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 12px;
 }
</style>

<br>
<div id="h2"><h2>ESP32 HOUSE CONTROL</h2>
<br>
```

```
<div>
<p>
  <i class="fas fa-fire" style="color:red;"></i>
  <span class="dht-labels">Fire in the house!</span>
  <label class="switch">
        <input class="dht-labels" type="checkbox" align="middle">
  <span class="slider round">
  </span>
  </label>
  </p>
</div>
</div>
<br><br><br>
<div class="Main">
<div class="Col1">
<div>
 <p>
  <i class="fas fa-wind" style="color:#ffcc00;"></i>
  <span class="dht-labels">Air pollution:</span>
  <span id="air" style="font-size: 30px">%AIR%</span>
  <span class=" dht-labels">ppm</span>
</p>
</div>

<div>
<p>
  <i class="fas fa-thermometer-half" style="color:#ffcc00;"></i>
  <span class="dht-labels">Temperature:</span>
  <span id="temperature" style="font-size: 30px">%TEMPERATURE%</span>
  <span class=" dht-labels">C</span>
```

```
</p>
</div>

<div>
<p>
  <i class="fas fa-water" style="color:#ffcc00;"></i>
  <span class="dht-labels">Humidity:</span>
  <span id="humidity" style="font-size: 30px">%HUMIDITY%</span>
  <span class=" dht-labels">%</span>
</p>
</div></div>

<div class="Col2">
<div>
<p>
  <i class="fas fa-volume-up" style="color:#0066cc;"></i>
  <span class="dht-labels">Buzzer Frequency</span>
  <input value="500" type="number" class="validate dht-labels">
  <span class="units dht-labels">Hz</span>
</p>
</div>

<div>
<p>
  <i class="fas fa-lightbulb" style="color:#0066cc;"></i>
  <span class="dht-labels">Lighting</span>
  <label class="switch">
        <input class="dht-labels" type="checkbox" align="middle">
  <span class="slider round">
  </span>
```

```
    </label>
    </p>
  </div>

  <div>
  <p>
    <i class="fas fa-door-open" style="color:#0066cc;"></i>
    <span class="dht-labels">Door</span>
    <label class="switch">
          <input class="dht-labels" type="checkbox" align="middle">
    <span class="slider round">
    </span>
    </label>
    </p>
  </div>

  <p>
    <button class="button" href = \"/data\"><b>Send data</b></button>
  </p>

  </div>
  </div>
</body>

<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("air").innerHTML = this.responseText;
```

```
    }
  };
  xhttp.open("GET", "/air", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;
</script>
</html>
)rawliteral";
```

```
String readT(){
  return String(dht.readTemperature());
}

String readH(){
  return String(dht.readHumidity());
}

String readA(){
  return String(analogRead(PINA));
}

String outputState(int output){
  if(digitalRead(output)){
    return "checked";
  }
  else {
    return "";
  }
}

String processor(const String &var){
  //Serial.println(var);
  if (var == "TEMPERATURE")
  {
    return readT();
  }
  else if (var == "HUMIDITY")
  {
```

```
    return readH();
  }
  if (var == "AIR")
  {
    return readA();
  }
  return String();
}

void setup() {
  Serial.begin(115200);
  Serial.println("Start...");
  SPI.begin();
  rfid.PCD_Init();
  //IrReceiver.begin(IR_PIN);
  dht.begin();
  lcd.init();
  lcd.backlight();
  EasyBuzzer.setPin(BUZZER_PIN);
  EasyBuzzer.stopBeep();
  servo.setPeriodHertz(50); // standard 50 hz servo
  servo.attach(SERVO_PIN);
  if (servo.attached())servo.write(0);
  pinMode(PINA, INPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(14, OUTPUT);
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  digitalWrite(14, LOW);
```

```
  pinMode(BTN_PIN, INPUT);
  pinMode(DIST, INPUT);
  // ESP32PWM::allocateTimer(0);
  // ESP32PWM::allocateTimer(1);
  // ESP32PWM::allocateTimer(2);
  // ESP32PWM::allocateTimer(3);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Put your card");

  WiFi.begin(ssid, password);
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
        { request->send_P(200, "text/html", index_html, processor); });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request)
        { request->send_P(200, "text/plain", readT().c_str()); });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request)
        { request->send_P(200, "text/plain", readH().c_str()); });
  server.on("/air", HTTP_GET, [](AsyncWebServerRequest *request)
        { request->send_P(200, "text/plain", readA().c_str()); });
  // server.on("/servo", HTTP_GET, [](AsyncWebServerRequest *request)
  //       { request->send_P(200, "text/plain", readS().c_str()); });
  server.begin();
}

int amount = 1;

void loop() {

  //WiFi
  if (WiFi.status() == 3 && amount == 1)
```

```
{
  Serial.println("CONNECTED!");
  Serial.println(WiFi.localIP());
  amount = 0;
}
if (WiFi.status() == 4)
{
  Serial.println("Connection failed!");
  delay(5000);
}
else if (WiFi.status() == 5)
{
  Serial.println("Oops... Connection lost");
  delay(5000);
}
else if (WiFi.status() == 1)
{
  Serial.println("No SSID available!");
  delay(5000);
}

//RGB and Button
if(digitalRead(BTN_PIN) && flagBtn == 0){
  //digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
  digitalWrite(14, LOW);
  flagBtn=!flagBtn;
  delay(100);
}
else if(digitalRead(BTN_PIN) && flagBtn == 1){
```

```
  //digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  digitalWrite(14, HIGH);
  flagBtn=!flagBtn;
  delay(100);
}

//IR obstacle
if (digitalRead(DIST) == 0){
  EasyBuzzer.stopBeep();
  counter++;
  Serial.println(counter);
  delay(100);
  if (counter>3){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Someone in the");
    lcd.setCursor(0, 4);
    lcd.print("house!");
    servo.write(0);
    EasyBuzzer.singleBeep(
    700,
    3000
    );
    delay(3000);
    EasyBuzzer.stopBeep();
    counter=0;
  }
}
```

```
//RFID
if (!rfid.PICC_IsNewCardPresent()){
  return;}
if (!rfid.PICC_ReadCardSerial()){
  return;}

uidDec = 0;
for (byte i = 0; i < rfid.uid.size; i++)
{
  uidDecTemp = rfid.uid.uidByte[i];
  uidDec = uidDec * 256 + uidDecTemp;
}
// Serial.println("Card UID: ");
// Serial.println(uidDec);

if (uidDec == UID){
  //EasyBuzzer.stopBeep();
  Serial.println("Welcome home!");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Welcome home!");
  servo.write(90);
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(14, LOW);
  delay(5000);
  servo.write(0);
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(14, HIGH);
```

```
  }
  else{
    Serial.println("WRONG!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Who are you?!");
    servo.write(0);
    digitalWrite(12, HIGH);
    digitalWrite(13, LOW);
    digitalWrite(14, LOW);EasyBuzzer.singleBeep(
      500,
      5000
    );
    EasyBuzzer.stopBeep();
  }
  //EasyBuzzer.stopBeep();

  rfid.PICC_HaltA(); // halt PICC
  rfid.PCD_StopCrypto1(); // stop encryption on PCD
}
```