КІБЕРБЕЗПЕКА

«ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ БЕЗДРОТОВОЇ СИГНАЛІЗАЦІЇ В САДОВОМУ ТОВАРИСТВІ ШЛЯХОМ ДОСЛІДЖЕННЯ ТА РОЗРОБКИ МЕРЕЖІ LORAWAN»

Шифр „СІГНАЛІЗАЦІЯ LORAWAN”

2021

ЗМІСТ

ВСТУП

У зв’язку з тим, що в Україні збільшується число кримінальних справ, пов’язаних з пограбуванням, дедалі активніше ставиться питання впровадження систем охорони для спостереження та сигналізації у випадку вторгнення. Зважаючи на те, що в Україні все більше поширюється Інтернет Речей (Internet of Things, IoT) є актуальним використання безпровідних технологій для організації охоронного комплексу.

Системи охорони потребує забезпечити високий рівень надійності безпровідного зв’язку для віддаленого контролю, високий рівень захищеності мережі від злонавмистного втручання.

Мета роботи: розробка надійної та ефективної безпровідної системи охорони.

Наукова новизна полягає в тому, що створена система охорони є безпровідною віддаленою корпоративною мережею, що вигідно виділяє її серед аналогів.

Завдання:

- розробка тестового комплексу системи охорони;
- дослідження якості віддаленого безпровідного зв’язку;
- розробка клієнтського та базового трансіверів на базі мікроконтролерів, котрі мають LoRaWAN-ядро;
- розробка конструкції надійної мережі датчиків для об’єктів охорони.

Практична застосування: запропонована система охорони є більш вдалою для використання в поширених територіально об’єктах. На відміну від існуючих аналогів, побудована система є незалежною від операторів мобільного зв’язку та не потребує оплати кожного повідомлення. Впровадження безпровідної корпоративної мережі дозволяє більш оперативно реагувати у випадку вторгнення злочинців. Запропонована трьохдротова мережа датчиків є простим та надійним рішенням, а також дозволяє мати необмежену кількість датчиків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз охоронних систем

Для захисту різноманітних об’єктів майна від злочинних процесів та явищ використовують різні засоби захисту. Охоронна система – найважливіша частина контролю захисту майна від ситуацій, в котрих будуть нанесені збитки людям та матеріальним і нематеріальним цінностям, пов’язаних, перш за все, з діяльністю несанкціонованих осіб.

Використання системи охорони дозволяє автоматизувати процес захисту майна, а саме – будівель, прилягаючої території, окремих приміщень та інше, від негативних процесів за різними необхідними критеріями.

На сьогоднішній день найбільш популярним рішенням серед існуючих є використання GSM-систем охорони. Приклади таких систем наведені в [1] та [2]. Основний їх принцип роботи: датчики під’єднані до контролера з GSM-модулем. У випадку перевищення якогось з параметрів, що відстежуються, дані відсилаються до користувача.

Недоліки GSM-систем охорони:

- постійна робота встановленої SIM-карти: для обслуговування приладу оператором мобільного зв’язку потрібне постійне поповнення рахунку. У випадку відсутності поповнення рахунку безпека об’єкту охорони ставиться під сумнів;
- рівень сигналу не завжди однаковий, а в деяких місцях, наприклад, в приміській або сільській місцевості рівень сигналу, зазвичай, низький, а від цього залежить швидкість передачі екстреного повідомлення;
- чуттєвість до рівня перешкод в мережі зв’язку;
- залежність від оператора мобільного зв’язку. У випадку технічних робіт або у випадку повторення ситуації 2014 року на території Донецької

області, коли був відключений мобільний зв’язок, існуючі рішення не будуть працювати, що створює можливу загрозу майну;

- складність монтажу та налаштування. Встановлення обладнання без допомоги спеціалістів не є можливим;
- низька енергоефективність GSM-модулю. Наприклад, у популярного GSM-модулю SIM800L, згідно з [4], максимальний струм живлення при передачі даних сягає 2 А. Це означає, що прилад потрібно постійно заряджати. А також ставить під сумнів використання великої кількості існуючих батарей та акумуляторів.

1.2 Аналіз об'єкту охорони

На рисунку 1.1 показана територія садового товариства. Маркером 1 позначене місцезнаходження пункту охорони, на якому знаходяться охоронці. Вони здійснюють пропуск на територію садового товариства лише господарів дачних ділянок та обслуговуючий персонал товариства. Також щоночі охорона здійснює обхід території товариства. Маркери 2 та 3 на рис. 1.1 вказують на місцезнаходження майбутніх об’єктів охорони. Можливо, з часом об’єктів охорони буде більше, але нині зацікавлених небагато.

Територія садового товариства має огорожу. Проте в деяких місцях вона відсутня. Для зловмисників немає великої складності потрапити на територію товариства, чим вони й користуються.

Як видно з рис. 1.1, об’єкти охорони знаходяться на відстані приблизно 200-250 м від пункту охорони.

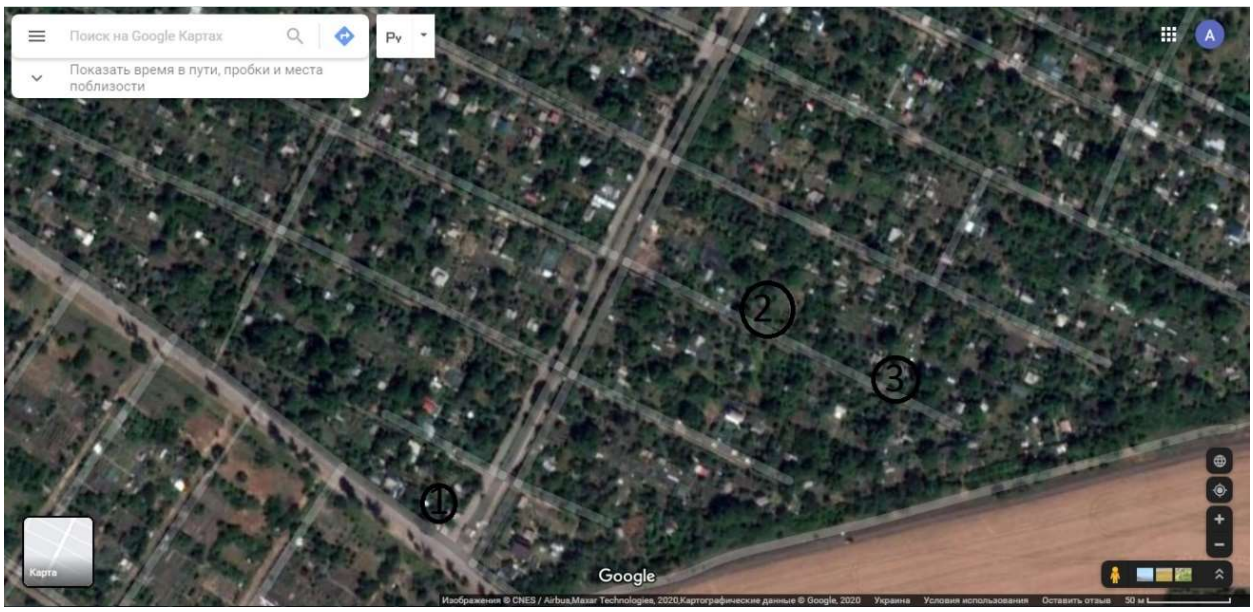

Рисунок 1.1 – Садове товариство. Скріншот з Google Maps

Так як об’єктом автоматизації є ціле садове товариство потрібно створити систему, що дозволить масштабування, буде працювати в різні погодні ситуації та при широкому діапазоні температур, так як в приміщеннях взимку відсутнє опалення.

Так як територія відносно велика, варіант провідної сигналізації відпадає, через високу ціну впровадження по всій території садового товариства, та низку надійність такої системи.

Було вирішено відмовитись від готових GSM-систем, так як необхідне постійне обслуговування та постійна плата операторам мобільного зв’язку, відсутність можливості організовано забезпечувати обслуговування незв’язаних між собою систем.

Було вирішено використовувати безпровідні технології та сучасні IoT рішення.

1.3 Аналіз LPWAN-технологій

Ключовою проблемою організації системи сигналізації стає безпровідна передача даних на великі відстані. Рішенням стали LPWAN-технології [5] (англ. енергоефективна мережа далекої дії). Прилад з дуже низьким

споживанням енергії передає дані на дуже великі відстані, при цьому наявні різні обмеження, наприклад, низька швидкість. Ця технологія працює в вільних від ліцензіювання частотах.

Переваги:

- відносно низька ціна пристроїв;
- низьке енергоспоживання;
- висока дальність передачі даних.

Недоліки:

- низька швидкість передачі даних;
- необхідність встановлювати антенну наскільки можливо вище від рівня землі, що не завжди є доцільним.

На даний момент існує безліч розробок LPWAN-мереж [5]. Однак, одна технологія не може вирішити всі проблеми. LPWAN-мережі вирішують лише деякі з них. Цю мережеву технологію використовують, коли необхідне охоплення великої території при низькій вартості розгортання, для приладів, у яких немає великих вимог до затримок, немає необхідності в високих швидкостях передачі даних та наявна необхідність в низькому енергоспоживанні. Зокрема, моніторинг систем чи умов є ідеальним випадком використання LPWAN-мережі.

Нині існують різноманітні LPWAN-технології. Порівняльне дослідження деяких з них наведено в [5].

Серед LPWAN-технологій широкого розповсюдження досягли LoRa та LoRaWAN[6].

LoRa використовує вільні від ліцензій частоти(індустріальні, наукові та медичні частоти (industrial, scientific and medical (ISM) bands)), нижчі за 1 ГГц, наприклад, 169 МГц, 433 МГц, 868 МГц(більшість країн Європи) та 915 МГц (Північна Америка). LoRa дозволяє передавати дані на великі відстані (більш ніж 10 км в сільській місцевості) з дуже низьким споживанням електроенергії.

1.4 Постановка завдань досліджень

Головна мета розробки – підвищення ефективності обслуговування системи охорони шляхом дослідження та розробки більш надійної та енергоефективної системи, ніж існуючі аналоги. Дана мета досягається шляхом виконання наступних завдань:

1. проектування системи охорони на базі мережі LoraWAN;
2. дослідження роботи безпровідних пристроїв віддаленого зв’язку;
3. проектування та розробка пристроїв системи;
4. проектування та розробка програмного забезпечення системи.

Розроблена система повинна відповідати таким вимогам:

1. надійність роботи;
2. енергоефективність приладів;
3. наявність можливості розширення системи;
4. використання батарейного типу живлення;
5. наявність виводу повідомлень стану тривоги у випадку перевищення норми параметрів.

2 ДОСЛІДНА ЧАСТИНА

2.1 Проектування структурної схеми дослідження системи охорони на базі LoraWAN

Для дослідження можливості використання LoraWAN була запропонована спрощена структура системи охорони, рисунок 2.1.

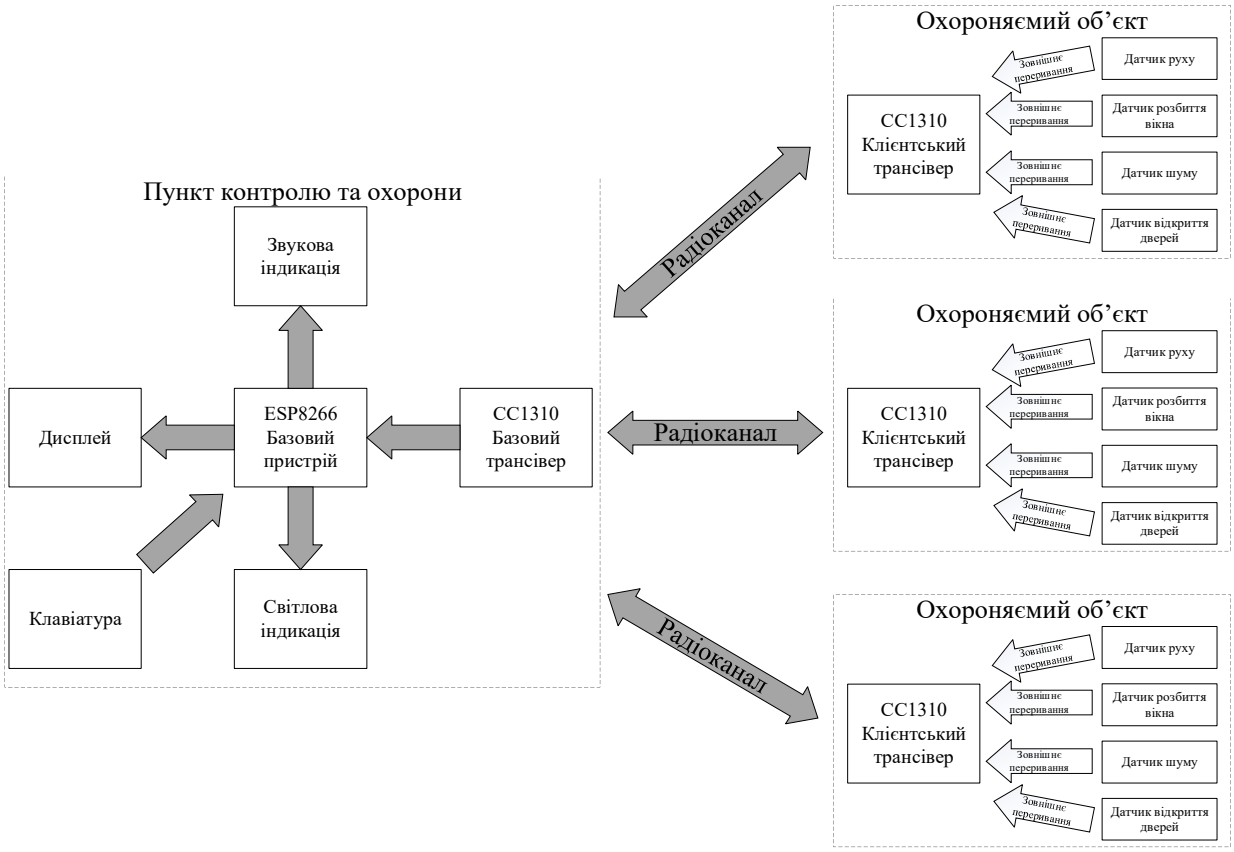

Рисунок 2.1- Структурна схема системи охорони

На першому етапі понад усе була поставлена задача спрощення організації дослідження. Тому використовуємо мережеву топологію «зірка». Також було вирішено відмовитись від використання mesh-мережі, так як її дуже важко реалізувати практично.

Центральний пристрій тестової мережі було вирішено встановити на пункті охорони, бо там постійно є світло та нагляд за обладнанням. До того ж переміщення отоплюється і не потрібно використовувати герметичні корпуси.

Базовий пристрій складається з мікроконтролеру ESP8266, до якого приєднані такі пристрої вводу-виводу: клавіатура, пристрої звукової та світлової індикації, дисплею. ESP8266 обмінюється даними з базовим трансівером за допомогою інтерфейсу UART.

Кожен об'єкт охорони, що тестується, має 4 типи датчиків: руху, розбиття вікна, шуму, відкриття дверей. Датчики кожного типу можна об'єднати логічними елементами, щоб зменшити кількість використовуваних портів вводу-виводу мікроконтролеру. Передбачається, шо клієнтські трансівери передають інформацію кожні 10 секунд, а у випадку надзвичайної ситуації – негайно.

2.2 Аналіз обраних основних периферійних приладів системи

Для задоволення вимог, поставлених до тестової системи було обрано мікроконтролер CC1310F128RGZ американського виробника напівпровідникових пристроїв Texas Instruments. Цей мікроконтролер побудований на ядрі ARM Cortex M3, ліцензований британською корпорацією ARM Holdings. Має в собі, перш за все, вбудоване низькоспоживаюче LoRa ядро, яке дозволяє використовувати стабільний безпровідний зв'язок на великі відстані. Основні параметри МК CC1310F128 вказані в [7].

Так як проектувати модуль LoRa з нуля досить складно та дорого, було вирішено використовувати готові модулі Ebyte E70-433T14S, котрі базуються на Texas Instruments CC1310. Це дозволяє зменшити час відладки апаратної складової проекту та зосередитись на вирішенні програмних питань.

Взагалі, розробка тестової мережі цілком орієнтується на перспективну систему, що була описана автором та його керівниками у роботі [8].

Для проектованої системи є гостра необхідність в наявності легкого способу виводу результатів дослідження мережі. Тому було вирішено встановити WiFi модуль.

Для використання на базовій станції було обрано ESP8266, так як він вже використовуйся у деяких розробках автора. Цей модуль дозволяє організувати якісну передачу даних за дуже мінімальну потребу в часі розробки. Приклади практичного використання наведені в [9, 10, 11 та 12].

2.3 Дослідження проблем забезпечення якості зв'язку роботи LoRaWAN-мережі

Серед існуючих проблем, котрі можуть виникнути при впровадженні LoRaWAN-мережі:

- можливе збільшення кількісті працюючих приладів в одній мережі;
- можливості масштабування мережі;
- коливання якості бездротового зв'язку.

Дослідження по максимально можливої кількості приладів, що можуть працювати в мережі LoRaWAN проведені в [13, 14].

Робота [15] зосереджена на проблемі масштабування мережі LoRaWAN.

Лише робота [16] проводить дослідження якості зв'язку в реальних умовах. Проте брати ці результати за основу не можна, так як вони проведені в іншій країні на частоті, що відрізняється від дозволених в Україні.

Було прийнято рішення дослідити якість передачі даних мікроконтролеру CC1310 при різних сценаріях роботи на частотах, дозволених в Україні.

Стенд для дослідження зображено на рис. 2.2, використовуються декілька таких пристроїв, один з котрих працює як Master, а інший як Slave.

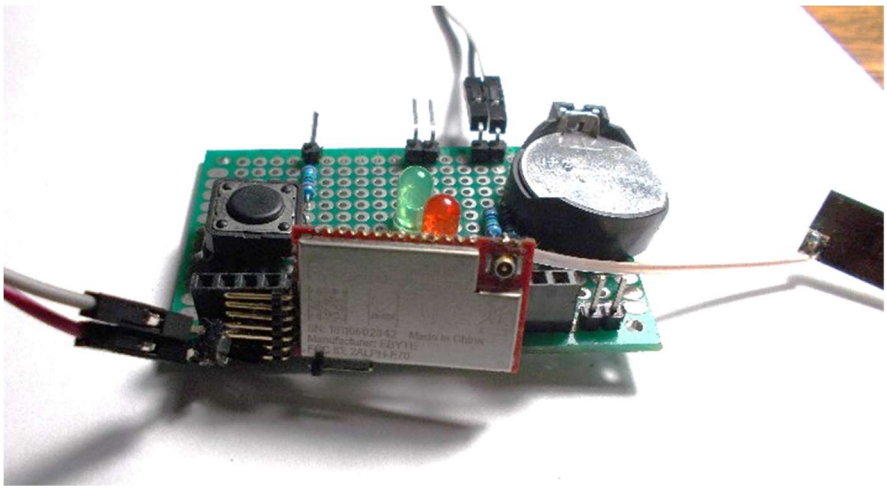

Рисунок 2.2 – Стенд для дослідження якості віддаленого бездротового зв’язку

На монтажній платі закріплено модуль LoraWAN з припаяною до нього мікрополосковою антеною, батарея CR2032, кнопка та декілька Led діодів. Програма керування наведена у додатку А.

В якості досліджень було перевірено три можливі різні ситуації: зв’язок в місті, зв’язок в приміській місцевості та зв’язок поряд з електромагнітними перешкодами.

Першим експериментом було вирішено провести дослідження зв’язку в міській місцевості. Якщо ідеальним випадком використанням LoRaWAN-пристрою є ситуація, коли прилади знаходяться в полі зору один одного, то використання в місті – дуже складний сценарій. На зв’язок впливають такі фактори, як висота встановлення пристрою-передатчика, відстань до пристрою-приймача, наявність перешкод на шляху проходження радіохвиль в вигляді будівель та металічних об’єктів, географічна складність рельєфу, наявність об’єктів з високим електромагнітним випромінюванням (трансформатори, електродвигуни та інше).

Пристрій-передатчик знаходився в студентському гуртожитку на висоті третього поверху. Пакети надсилались з інтервалом 5 секунд. Пристрій-приймач переносився навколо території навчального закладу. Після того як пакет надходив до приймача від надсилався без змін знову до передатчику.

Передатчик надсилав логи до персонального комп'ютеру через USB-UART конвертер, де вони зберігались.

В таблиці 2.1 наведені результати експерименту. Результати наведені в відсотках пакетів, які успішно повернулись до передатчику.

Таблиця 2.1 – Кількість пакетів, які успішно повернулись до центрального пристрою. Експеримент в міській місцевості

Розмір пакету	Швидкість передачі даних			
	650 біт/сек	2,5 кбіт/сек	5 кбіт/сек	50 кбіт/сек
20 байт	72%	56%	50%	30%
50 байт	63,5%	53%	46%	28,5%
128 байт	42%	35%	28,5%	21%
256 байт	30,5%	22%	20%	15%

Другим експериментом було проведено тестування зв'язку в приміській місцевості, а саме на території майбутнього впровадження системи. Алгоритм експерименту відповідає попередньому.

Для місцевості характерна повна відсутність багатоповерхових будівель. Наявні маленькі будинки. Майже відсутні джерела електромагнітного випромінювання .

Результати експерименту наведені в табл. 2.1.

Таблиця 2.2 - Кількість пакетів, які успішно повернулись до центрального пристрою. Експеримент в приміській місцевості

Розмір пакету	Швидкість передачі даних			
	650 біт/сек	2,5 кбіт/сек	5 кбіт/сек	50 кбіт/сек
20 байт	95%	85%	83,5%	79%
50 байт	92%	82,5%	79,5%	75%
128 байт	91,5%	83,5%	81%	72%
256 байт	82,5%	80%	78%	69,5%

Третім експериментом було проведено тестування зв’язку поруч з об’єктом з сильним електромагнітним випромінюванням, а саме - поряд з трансформаторною підстанцією.

Ілюстрація експерименту показана на рис. 2.3.

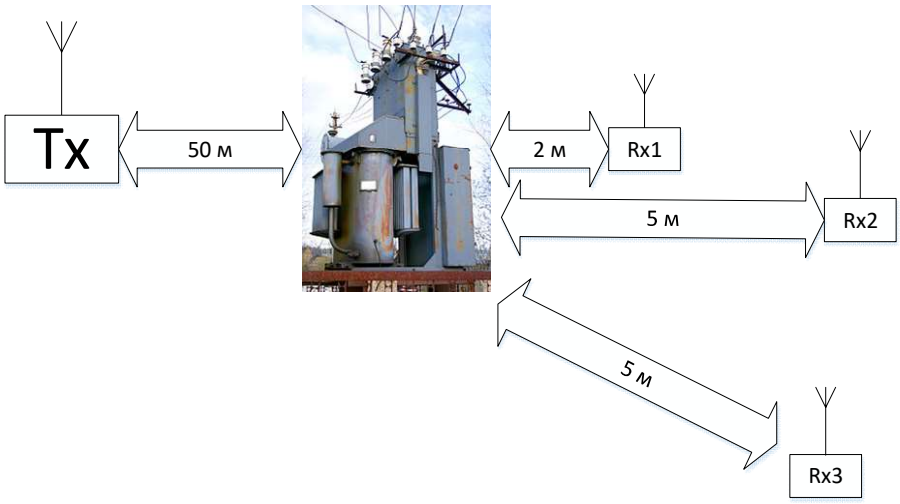

Рисунок 2.3 – Ілюстрація третього експерименту

Пристрій-передатчик позначений як Tx на рис. 2.3. Він передає 100 пакетів розміром 20 байт. Інтервал між пакетами - 1 секунда. В першому випадку приймач знаходиться на відстані 2 метри від трансформаторної підстанції. Передатчик, підстанція та приймач знаходяться на одній лінії. В результаті приймач не отримав жодного пакету.

В другому випадку всі елементи теж знаходяться на одній лінії, проте відстань від приймача до підстанції збільшилась до 5 м. В цьому випадку передатчик отримав назад 83% пакетів.

В третьому випадку приймач було переміщено так, щоб підстанція не була на одній лінії з пристроями. В цьому випадку всі 100 пакетів були повернуті пристрою-передатчику.

2.4 Реєстрація вимірюваних даних за допомогою WiFi модулю

Вивід вимірюваних даних на об’єкті дослідження було вирішено робити через http-сторінку. Вона дозволить зручно спостерігати параметри системи та налаштовувати віддалені пристрої навіть з планшета.

Основна сторінка http-серверу показана на рисунку 2.9. Там показано, що в центрі є таблиця с активними пристроями. В ній вказані унікальний ID пристрою, номер вулиці та номер будинку, де встановлений прилад, а також статус пристрою (1 – все гаразд, 0 – тривога). Нижче таблиці є кнопки взаємодії зі сторінкою. (Програма наведена у додатку Б).

Для додавання нового пристрою зроблена відповідна форма. В формі наявні 3 поля: унікальний ID пристрою, номер вулиці та номер будинку, де буде встановлений прилад. Після натиску на кнопку «Зберегти» нові дані будуть збережені в базі даних на microSD-картці.

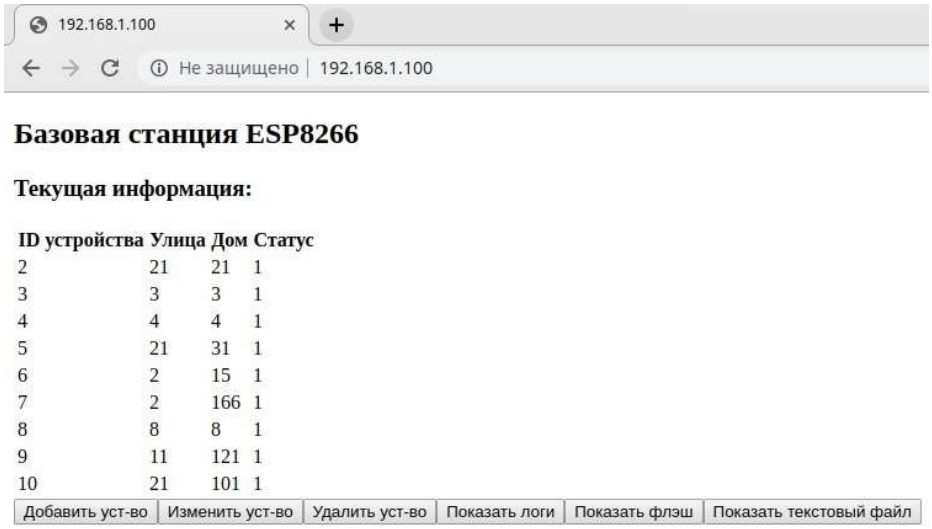

Рисунок 2.9 – Основна http-сторінка

Таким чином, було проведене дослідження роботи мережі LoraWAN на реальному об’єкті, яке дозволило виявити обмеження при плануванні надійності мережі.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка базового пристрою

Базовим пристроєм для системи тестування мережі LoraWAN є ESP8266. Електрична схема зображена на рисунку 3.1. Основний функціонал базового пристрою - отримання інформації, а також у разі необхідності система може ввімкнути сигнал тривоги та вивести інформацію на дисплей. Цю інформацію використовує сторож. Дослідницька інформація виводиться за допомогою http-сторінки.

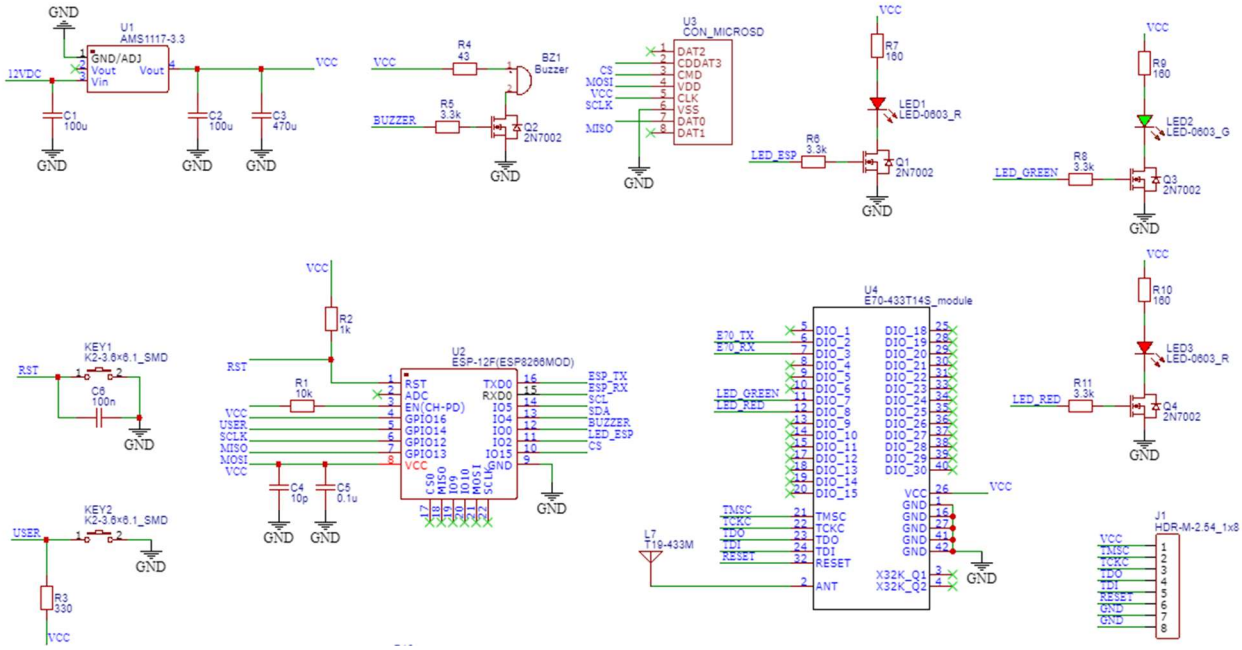

Рисунок 3.1 – Електрична схема базового пристрою

Алгоритм роботи модуля LoraWAN зображено на рисунку 3.2. Після ініціалізації пристрій переходить в режим очікування повідомлення. Мікроконтролер відправляє та отримує повідомлення у вигляді пакетів. Ці пакети мають довжину, котру можна змінювати. В існуючій версії прошивки довжина складає 30 байт. Потім пакет відправляється до ESP8266.

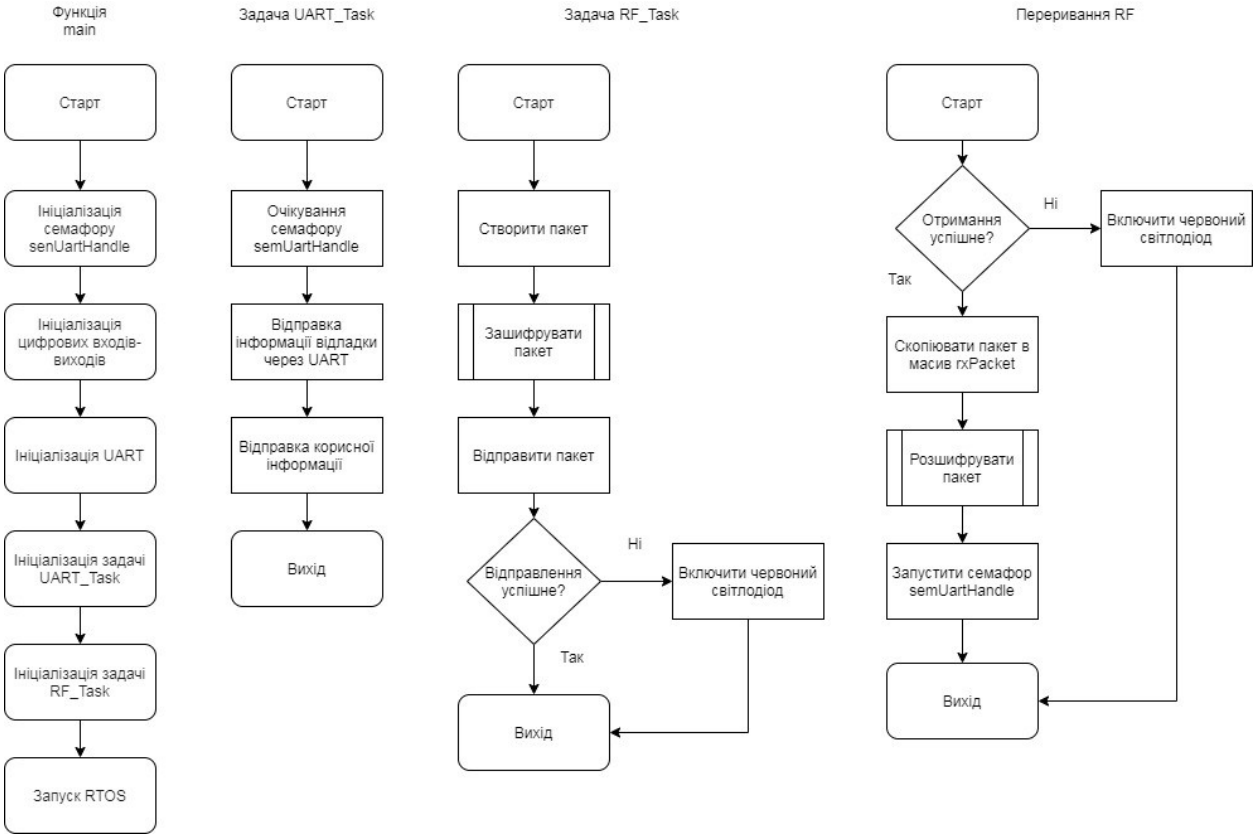

Рисунок 3.3 – Блок-схема алгоритму роботи модуля LoraWAN

Обмін пакетами між базовою станцією та станціями клієнтів здійснюється [8] в синхронному режимі. Щосекунди базова станція відправляє інформаційно-синхронізуючий сигнал Маяка. Якщо клієнт збирається провести сеанс зв'язку то він синхронізується по сигналу Маяка та генерую блок відповіді з випадковим номером. Виділено дві часові зони: зона аварійних повідомлень і зона інформаційних повідомлень (рисунок 3.4).

Рисунок 3.4 – Часова діаграма обміну повідомленнями

Зона аварійних повідомлень виділена тільки для повідомлень спрацьовування сигналізації, тому тривалість її становить 0,25 секунди. Час, що залишився, виділений для інформаційних сигналів. Часові зони розділені на блоки. Номер блоку повідомлення вибирається випадковим образом, що забезпечує одночасну передачу інформації від декількох вилучених об'єктів. Підтвердження приймання містить наступний Маяк.

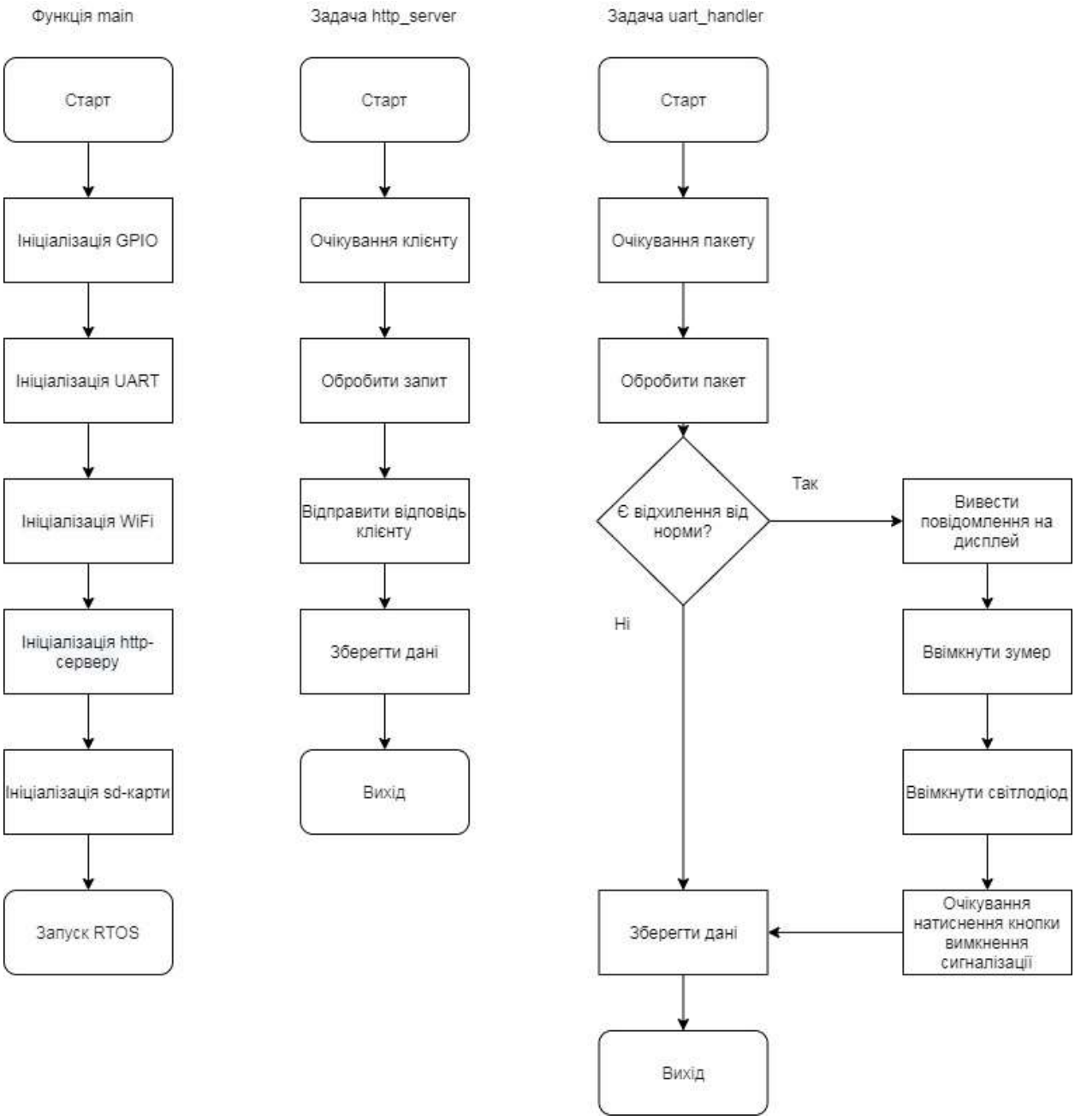

Рисунок 3.4 – Блок-схема алгоритму роботи базового пристрою

3.2 Розробка та дослідження алгоритму шифрування пакетів

Для прототипу було вирішено використовувати простий алгоритм шифрування даних, наведений на рис. 3.5.

```
for (int32_t i = 0; i < PACKET_SIZE; i++)
    {   crypted_packet[i] ^= key[i];  }
for (int32_t i = 0; i < PACKET_SIZE - 1; i++)
    {  crypted_packet [i] ^= crypted_packet [i+1];  }
```

Рисунок 3.5 – Алгоритм шифрування пакету

В першому циклі здійснюється операція АБО-НІ (XOR) кожного і-елементу масиву з кожним і-елементом масиву ключа. Ключ задається константним та є однаковим для базового та клієнтського трансіверів.

В другому циклі виконується операція АБО-НІ кожного і-елементу масиву з і+1-елементом (або наступним).

Отриманий пакет необхідно розшифрувати. Для вирішення цієї проблеми був розроблений відповідний алгоритм, показаний на рис. 3.6.

```
for (int32_t i = PACKET_SIZE - 2; i >= 0; i--)
    {   uncrypt[i] ^= uncrypt[i+1];  }
for (int32_t i = 0; i < PACKET_SIZE; i++)
     {   uncrypt[i] ^= key[i];  }
```

Рисунок 3.6 – Алгоритм розшифрування пакету

Як видно з рис. 3.6, розроблений алгоритм є оберненим до того, що наведений на рис. 3.5.

Таким чином, був розроблений алгоритм та програмний код шифрування пакетів, якій відрізняється від стандартного.

3.3 Розробка клієнтських систем сигналізації

Систему керування охоронюваним об'єктом показано на рис. 3.7. Основний блок системи розміщений [8] у герметичному циліндричному корпусі, куди за допомогою гермовводів підключені дводротова лінія 220V і трьохдротова інформаційно-живильна лінія.

Інформаційно-живильна лінія проходить через увесь охоронюваний об'єкт. У ключових точках контролю встановлюються модулі інфрачервоного датчика руху. На двері встановлюються датчики відкриття. Контроль цілісності лінії здійснюється за допомогою кінцевих модулів датчиків обриву лінії.

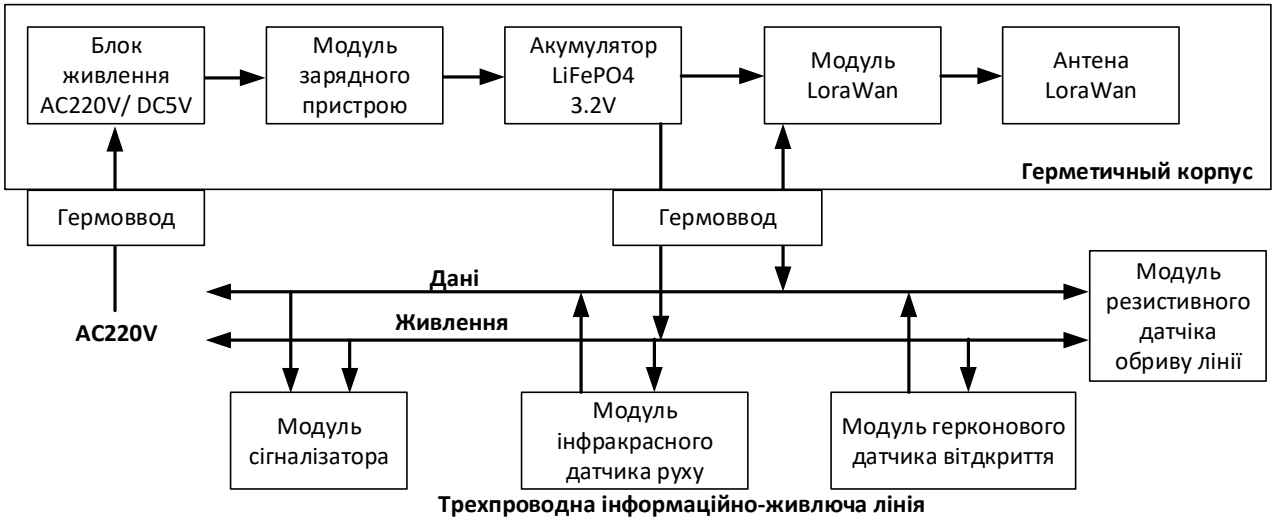

Рисунок 3.7 - Система керування охоронюваним об'єктом

Система керування охоронюваним об'єктом розрахована на роботу як при зовнішньому живленні, так і при тривалому відключенні від нього. Для цього усередині герметичного корпусу встановлений ємний акумулятор LiFePO4. Зв'язок з верхнім рівнем здійснюється за допомогою модуля LoraWAN, оснащений вбудованої вкороченою антеною. У випадку виникнення проблем зі зв'язком застосовується антена без укорочення.

Особливістю використання CC1310 в складі системи сигналізації є вимога автономності по живленню. Виходячи з параметрів модуля LoraWAN,

максимальне напруження живлення не повинно перевищувати 3.8В. Кращим варіантом для системи охорони є застосування LiFePO4 акумуляторів.

Так, акумулятор 14500 HFC14500 3,2V 500mAh LiFePO4 має напругу номінальну: 3.2V; повний заряд: 3.65V; повний розряд 2V; Максимально безперервний розрядний струм: до 13А; заряд: 1С; робоча температура -25С ... + 75С; 50 mOm; ресурс при половинному розряді більше 5000 циклів.

LiFePO4 акумулятори зазвичай не оснащені контролером заряду. Тому, з урахуванням того, що необхідний тільки один елемент на напругу 3.2В, був використаний модуль заряду на базі мікросхеми TP5000. Вхідна напруга контролера заряду становить 5 В. Тому для забезпечення електроживлення контролера заряду був обраний безкорпусне імпульсне джерело живлення SM-PLB05A з параметрами: потужність 5 Вт, вихідна напруга 5В, вихідний струм 1А, вхідна напруга 85 ... 264VAC, робоча температура -30 ... 70 ° С.

Габаритні розміри модулів E70-433T14S, модулів живлення та акумуляторної батареї дозволяють їх лінійно розмістити всередині стандартної водопровідної пластикової труби діаметром 32мм. На рис. 3.13 показана структурна схема і зовнішній вигляд елементів живлення перед поміщенням в корпус сигналізації. У вхідному ланцюзі джерела живлення були додатково встановлені засоби захисту - індуктивність на 47мкГн від імпульсних сплесків, варистор 07D471K від перенапруги і термістор 5D-7 для обмеження струму.

Додаткові засоби захисту дозволяють запобігти виходу з ладу джерела живлення при близьких грозових розрядах.

Рисунок 3.8 – Розміщення елементів живлення сигналізації

На рисунку 3.9 показаний корпус системи сигналізації, виконаний за допомогою пайки з дюймової пластикової труби і заглушки.

Рисунок 3.9 – Корпус системи сигналізації

Таким чином, клієнтська частина системи сигналізації виготовлюється у повністю герметичному вигляді і завдяки використання LiFePO4 акумулятора може працювати навіть при відсутності опалення і при мінусових температурах. Вдало вибране технічне рішення дозволяє зменшити потрібні кошти як на виготовлення так і на подальший супровід системи сигналізації.

ВИСНОВКИ

В даній роботі була спроектована та досліджена система охорони, що базується на LoRaWAN-пристроях. Створені прототипи основних компонентів охоронної системи.

Була досліджена якість зв’язку LoRaWAN-пристроїв при різних можливих сценаріях роботи. Запропоновано періодично контролювати під’єднання клієнтських модулів з метою своєчасного виявлення спроб придушення каналу зв’язку.

Розроблена система є більш економічно доцільною, надійною та енергоефективною відносно існуючих аналогічних систем, особливо в плані корпоративного використання. Ця система найбільш актуальна для розгортання у вигляді корпоративної мережі моніторингу. Запропонована система є незалежною від мобільного зв’язку.

Розроблена трьохдротова мережа датчиків клієнтської сигналізації є простою та надійною, легко масштабується та може бути адаптована до будь-якого переміщення.

Розроблені прототипи основних компонентів системи. Спроектовано та розроблено програмне забезпечення системи. Використані сучасні операційні системи реального часу (TI RTOS та FreeRTOS), котрі дозволяють модернізувати програмний код з мінімальними зусиллями.

ПЕРЕЛІК ПОСИЛАНЬ

1. Khan S. R. et al. Design and implementation of low cost home security system using GSM network //International Journal of Scientific & Engineering Research. – 2012. – Т. 3. – №. 3. – С. 1.
2. Bangali J., Shaligram A. Design and Implementation of Security Systems for Smart Home based on GSM technology //International Journal of Smart Home. – 2013. – Т. 7. – №. 6. – С. 201-208.
3. Mekki K. et al. A comparative study of LPWAN technologies for large-scale IoT deployment //ICT express. – 2019. – Т. 5. – №. 1. – С. 1-7.
4. Datasheet GSM GPRS SIM800L // – URL: https://img.filipeflop.com/files/download/Datasheet_SIM800L.pdf – Дата доступу: 28.01.21 – Datasheet GSM GPRS SIM800L.
5. Mekki K. et al. A comparative study of LPWAN technologies for large-scale IoT deployment //ICT express. – 2019. – Т. 5. – №. 1. – С. 1-7.
6. de Carvalho Silva J. et al. LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities //2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech). – IEEE, 2017. – С. 1-6.
7. Instruments T. CC1310 SimpleLink Ultra-Low-Power Sub-1 GHz Wireless MCU //Texas Instruments: Dallas, TX, USA. – 2016.
8. Застосування протоколу бездротової передачі даних у системах керування. / Донченко Є.І., Жартовський О.В., Мещеряков А.О., Рябин І.О. // Вісник Донбаської державної машинобудівної академії. – Краматорськ: ДДМА, 2020. – № 1 (47). ISSN 1993-8322
9. Thaker T. ESP8266 based implementation of wireless sensor network with Linux based web-server //2016 Symposium on Colossal Data Analysis and Networking (CDAN). – IEEE, 2016. – С. 1-5.
10. Kodali R. K., Mahesh K. S. Low cost ambient monitoring using ESP8266 //2016 2nd International Conference on Contemporary Computing and Informatics (IC3I). – IEEE, 2016. – С. 779-782.
11. Kodali R. K., Soratkal S. R. MQTT based home automation system using ESP8266 //2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). – IEEE, 2016. – С. 1-5.
12. Saha S., Majumdar A. Data centre temperature monitoring with ESP8266 based Wireless Sensor Network and cloud based dashboard with real time alert system //2017 Devices for Integrated Circuit (DevIC). – IEEE, 2017. – С. 307-310.

13. Varsier N., Schwoerer J. Capacity limits of LoRaWAN technology for smart metering applications //2017 IEEE international conference on communications (ICC). – IEEE, 2017. – С. 1-6.

14. Adelantado F. et al. Understanding the limits of LoRaWAN //IEEE Communications magazine. – 2017. – Т. 55. – №. 9. – С. 34-40.

15. Mikhaylov K., Petäjäjärvi J., Janhunen J. On LoRaWAN scalability: Empirical evaluation of susceptibility to inter-network interference //2017 European Conference on Networks and Communications (EuCNC). – IEEE, 2017. – С. 1-6.

16. Wixted A. J. et al. Evaluation of LoRa and LoRaWAN for wireless sensor networks //2016 IEEE SENSORS. – IEEE, 2016. – С. 1-3.

17. Datasheet PIR Sensor D203S// – URL: https://www.compel.ru/item-pdf/9413e50b7243fc6e905d241f18622574/pn/pir~d203s.pdf – Дата доступу: 05.02.21 – Datasheet PIR Sensor D203S.

ДОДАТОК А. Програмний код модуля Ebyte E70-433T14S

```
/***** Includes *****/
/* Standard C Libraries */
#include <stdlib.h>
/* TI Drivers */
#include <ti/drivers/rf/RF.h>
#include <ti/drivers/PIN.h>
/* Driverlib Header files */
#include DeviceFamily_constructPath(driverlib/rf_prop_mailbox.h)
/* Board Header files */
#include "Board.h"
/* Application Header files */
#include "RFQueue.h"
#include "smartrf_settings/smartrf_settings.h"
/***** Defines *****/
/* Packet RX/TX Configuration */
/* Max length byte the radio will accept */
#define PAYLOAD_LENGTH        20
/* Set Transmit (echo) delay to 100ms */
#define TX_DELAY              (uint32_t) (4000000*0.1f) //0 //
/* NOTE: Only two data entries supported at the moment */
#define NUM_DATA_ENTRIES      2
/* The Data Entries data field will contain:
 * 1 Header byte (RF_cmdPropRx.rxConf.bIncludeHdr = 0x1)
 * Max 30 payload bytes
 * 1 status byte (RF_cmdPropRx.rxConf.bAppendStatus = 0x1) */
#define NUM_APPENDED_BYTES    2
/* Log radio events in the callback */
//#define LOG_RADIO_EVENTS
/***** Prototypes *****/
static void echoCallback(RF_Handle h, RF_CmdHandle ch, RF_EventMask e);
/***** Variable declarations *****/
```

```
static RF_Object rfObject;
static RF_Handle rfHandle;
/* Pin driver handle */
static PIN_Handle ledPinHandle;
static PIN_State ledPinState;
/* Buffer which contains all Data Entries for receiving data.
 * Pragmas are needed to make sure this buffer is aligned to a 4 byte boundary
 * (requirement from the RF core) */
#if defined(__TI_COMPILER_VERSION__)
#pragma DATA_ALIGN(rxDataEntryBuffer, 4)
static uint8_t
rxDataEntryBuffer[RF_QUEUE_DATA_ENTRY_BUFFER_SIZE(NUM_DATA_ENTRIES,
                                    PAYLOAD_LENGTH,          NUM_APPENDED_BYTES)];
#elif defined(__IAR_SYSTEMS_ICC__)
#pragma data_alignment = 4
static uint8_t
rxDataEntryBuffer[RF_QUEUE_DATA_ENTRY_BUFFER_SIZE(NUM_DATA_ENTRIES,
                                    PAYLOAD_LENGTH,          NUM_APPENDED_BYTES)];
#elif defined(__GNUC__)
static uint8_t
rxDataEntryBuffer[RF_QUEUE_DATA_ENTRY_BUFFER_SIZE(NUM_DATA_ENTRIES,
                        PAYLOAD_LENGTH,    NUM_APPENDED_BYTES)]
                                    __attribute__((aligned(4)));
#else
#error This compiler is not supported
#endif //defined(__TI_COMPILER_VERSION__)
/* Receive Statistics */
static rfc_propRxOutput_t rxStatistics;
/* Receive dataQueue for RF Core to fill in data */
static dataQueue_t dataQueue;
static rfc_dataEntryGeneral_t* currentDataEntry;
static uint8_t packetLength;
static uint8_t* packetDataPointer;
```

```
static uint8_t txPacket[PAYLOAD_LENGTH];
#ifdef LOG_RADIO_EVENTS
static volatile RF_EventMask eventLog[32];
static volatile uint8_t evIndex = 0;
#endif // LOG_RADIO_EVENTS
/* Application LED pin configuration table: All LEDs board LEDs are off. */
PIN_Config pinTable[] =
{
#if defined(Board_CC1352R1_LAUNCHXL)
 Board_DIO30_RFSW | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
#endif
#if defined(Board_CC1350_LAUNCHXL)
 Board_DIO30_SWPWR | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
#endif
 Board_PIN_LED1 | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
 Board_PIN_LED2 | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
 IOID_8        | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
 PIN_TERMINATE
};
/***** Function definitions *****/
void *mainThread(void *arg0)
{
   RF_Params rfParams;
   RF_Params_init(&rfParams);
   /* Open LED pins */
   ledPinHandle = PIN_open(&ledPinState, pinTable);
   if (ledPinHandle == NULL)
   {
      while(1);
   }
```

```
if( RFQueue_defineQueue(&dataQueue,       rxDataEntryBuffer,
                    sizeof(rxDataEntryBuffer),      NUM_DATA_ENTRIES,
                    PAYLOAD_LENGTH + NUM_APPENDED_BYTES))
{  /* Failed to allocate space for all data entries */
   PIN_setOutputValue(ledPinHandle, Board_PIN_LED1, 1);
   PIN_setOutputValue(ledPinHandle, Board_PIN_LED2, 1);
   while(1);
}
/* Modify CMD_PROP_TX and CMD_PROP_RX commands for application needs */
/* Set the Data Entity queue for received data */
RF_cmdPropRx.pQueue = &dataQueue;
/* Discard ignored packets from Rx queue */
RF_cmdPropRx.rxConf.bAutoFlushIgnored = 1;
/* Discard packets with CRC error from Rx queue */
RF_cmdPropRx.rxConf.bAutoFlushCrcErr = 1;
/* Implement packet length filtering to avoid PROP_ERROR_RXBUF */
RF_cmdPropRx.maxPktLen = PAYLOAD_LENGTH;
/* End RX operation when a packet is received correctly and move on to the
 * next command in the chain */
RF_cmdPropRx.pktConf.bRepeatOk = 0;
RF_cmdPropRx.pktConf.bRepeatNok = 1;
RF_cmdPropRx.startTrigger.triggerType = TRIG_NOW;
RF_cmdPropRx.pNextOp = (rfc_radioOp_t *)&RF_cmdPropTx;
/* Only run the TX command if RX is successful */
RF_cmdPropRx.condition.rule = COND_STOP_ON_FALSE;
RF_cmdPropRx.pOutput = (uint8_t *)&rxStatistics;
RF_cmdPropTx.pktLen = PAYLOAD_LENGTH;
RF_cmdPropTx.pPkt = txPacket;
RF_cmdPropTx.startTrigger.triggerType = TRIG_REL_PREVEND;
RF_cmdPropTx.startTime = TX_DELAY;
/* Request access to the radio */
rfHandle = RF_open(&rfObject, &RF_prop,
                (RF_RadioSetup*)&RF_cmdPropRadioDivSetup, &rfParams);
```

```
/* Set the frequency */
RF_postCmd(rfHandle, (RF_Op*)&RF_cmdFs, RF_PriorityNormal, NULL, 0);
while(1)
{
   /* Wait for a packet
    * - When the first of the two chained commands (RX) completes, the
    * RF_EventCmdDone and RF_EventRxEntryDone events are raised on a
    * successful packet reception, and then the next command in the chain
    * (TX) is run
    * - If the RF core runs into an issue after receiving the packet
    * incorrectly onlt the RF_EventCmdDone event is raised; this is an
    * error condition
    */
   RF_EventMask terminationReason =
         RF_runCmd(rfHandle, (RF_Op*)&RF_cmdPropRx, RF_PriorityNormal,
                  echoCallback, (RF_EventRxEntryDone |
                  RF_EventLastCmdDone));
   switch(terminationReason)
   {
      case RF_EventLastCmdDone:
         // A stand-alone radio operation command or the last radio
         // operation command in a chain finished.
         break;
      case RF_EventCmdCancelled:
         // Command cancelled before it was started; it can be caused
         // by RF_cancelCmd() or RF_flushCmd().
         break;
      case RF_EventCmdAborted:
         // Abrupt command termination caused by RF_cancelCmd() or
         // RF_flushCmd().
         break;
      case RF_EventCmdStopped:
         // Graceful command termination caused by RF_cancelCmd() or
```

```
        // RF_flushCmd().
        break;
    default:
        // Uncaught error event
        while(1);
}
uint32_t cmdStatus = ((volatile RF_Op*)&RF_cmdPropRx)->status;
switch(cmdStatus)
{
    case PROP_DONE_OK:
        // Packet received with CRC OK
        break;
    case PROP_DONE_RXERR:
        // Packet received with CRC error
        break;
    case PROP_DONE_RXTIMEOUT:
        // Observed end trigger while in sync search
        break;
    case PROP_DONE_BREAK:
        // Observed end trigger while receiving packet when the command is
        // configured with endType set to 1
        break;
    case PROP_DONE_ENDED:
        // Received packet after having observed the end trigger; if the
        // command is configured with endType set to 0, the end trigger
        // will not terminate an ongoing reception
        break;
    case PROP_DONE_STOPPED:
        // received CMD_STOP after command started and, if sync found,
        // packet is received
        break;
    case PROP_DONE_ABORT:
        // Received CMD_ABORT after command started
```

```
            break;
        case PROP_ERROR_RXBUF:
            // No RX buffer large enough for the received data available at
            // the start of a packet
            break;
        case PROP_ERROR_RXFULL:
            // Out of RX buffer space during reception in a partial read
            break;
        case PROP_ERROR_PAR:
            // Observed illegal parameter
            break;
        case PROP_ERROR_NO_SETUP:
            // Command sent without setting up the radio in a supported
            // mode using CMD_PROP_RADIO_SETUP or CMD_RADIO_SETUP
            break;
        case PROP_ERROR_NO_FS:
            // Command sent without the synthesizer being programmed
            break;
        case PROP_ERROR_RXOVF:
            // RX overflow observed during operation
            break;
        default:
            // Uncaught error event - these could come from the
            // pool of states defined in rf_mailbox.h
            while(1);
    }
  }
}

static void echoCallback(RF_Handle h, RF_CmdHandle ch, RF_EventMask e)
{
#ifdef LOG_RADIO_EVENTS
  eventLog[evIndex++ & 0x1F] = e;
```

```
#endif// LOG_RADIO_EVENTS
  if (e & RF_EventRxEntryDone)
  {
    /* Successful RX */
    /* Toggle LED2, clear LED1 to indicate RX */
    //PIN_setOutputValue(ledPinHandle, Board_PIN_LED1, 0);
    PIN_setOutputValue(ledPinHandle, Board_PIN_LED1,
                    !PIN_getOutputValue(Board_PIN_LED1));
    /* Get current unhandled data entry */
    currentDataEntry = RFQueue_getDataEntry();
    /* Handle the packet data, located at &currentDataEntry->data:
     * - Length is the first byte with the current configuration
     * - Data starts from the second byte */
    packetLength      = *(uint8_t *)(&(currentDataEntry->data));
    packetDataPointer = (uint8_t *)(&(currentDataEntry->data));
    /* Copy the payload + status byte to the rxPacket variable, and then
     * over to the txPacket         */
    memcpy(txPacket, packetDataPointer, PAYLOAD_LENGTH);
    RFQueue_nextEntry();
  }
  else if (e & RF_EventLastCmdDone)
  {  /* Successful Echo (TX)*/
    /* Toggle LED2, clear LED1 to indicate RX */
    //PIN_setOutputValue(ledPinHandle, Board_PIN_LED1, 0);
    PIN_setOutputValue(ledPinHandle, IOID_8, !PIN_getOutputValue(IOID_8));
  }
  else // any uncaught event
  {
    /* Error Condition: set LED1, clear LED2 */
    PIN_setOutputValue(ledPinHandle, Board_PIN_LED1, 1);
    PIN_setOutputValue(ledPinHandle, IOID_8, 0);
  }
}
```

ДОДАТОК Б. Програмний код базового пристрою ESP8266

```
#include <SD.h>
#include <ESP8266WiFi.h>
#include <SPI.h>
#define DEBUG 1
#ifndef STASSID
  #define STASSID "ASUS_E8"
  #define STAPSK  "12345678"
#endif

#define LED 4

File root;
const char* ssid = STASSID;
const char* password = STAPSK;
const static String nameDataBase = "DB.txt";
String html_table = "";
const String html_head = ""
"<!DOCTYPE html>"
"<html>"
"<head>"
  "<meta http-equiv='content-type' content='text/html' charset='utf-8'>"
"</head>"
"<body>";
const String html_main_up = ""
"<h2>Базовая станция ESP8266</h2>"
"<h3>Текущая информация:</h3>";

const String html_table_start = ""
"<div>"
  "<table>"
```

```
  "<tr>"
    "<th>ID устройства</th>"
    "<th>Улица</th>"
    "<th>Дом</th> "
    "<th>Статус</th>"
  "</tr>";
  const String html_table_end = ""
  "</table>"
"</div>";
const String html_button_menu = ""
"<form action='#' method='get'>"
  "<input type='submit' class='submit' name='add'    value='Добавить уст-во'>"
  "<input type='submit' class='submit' name='change' value='Изменить уст-во'>"
  "<input type='submit' class='submit' name='remove' value='Удалить уст-во'>"
  "<input type='submit' class='submit' name='view'   value='Показать логи'>"
  "<input type='submit' class='submit' name='flash'  value='Показать флэш'>"
  "<input type='submit' class='submit' name='print_txt'    value='Показать текстовый
файл'>"
"</form>";
const String html_form_add = ""
"<form action='#' method='get'>"
  "<div>"
    "<label for='ID'>ID устройства</label>"
    "<input name='ID' id='ID' value='001'>"
  "</div>"
  "<div>"
    "<label for='street'>Номер улицы : </label>"
    "<input name='street' id='street' value='0'>"
  "</div>"
  "<div>"
    "<label for='house'>Номер дома</label>"
    "<input name='house' id='house' value='10'>"
  "</div>"
```

```
  "<div>"
    "<button>Сохранить</button>"
  "</div>"
"</form>";
const String html_form_change = ""
"<form action='#' method='get'>"
  "<div>"
    "<label for='oldID'>ID устройства</label>"
    "<input name='oldID' id='oldID' value='001'>"
  "</div>"
  "<div>"
    "<label for='street'>Номер улицы : </label>"
    "<input name='street' id='street' value='0'>"
  "</div>"
  "<div>"
    "<label for='house'>Номер дома</label>"
    "<input name='house' id='house' value='10'>"
  "</div>"
  "<div>"
    "<button>Изменить</button>"
  "</div>"
"</form>";
const String html_form_remove = ""
"<form action='#' method='get'>"
  "<div>"
    "<label for='removeID'>ID устройства</label>"
    "<input name='removeID' id='removeID' value='001'>"
  "</div>"
  "<div>"
    "<button>Удалить</button>"
  "</div>"
"</form>";
const String html_form_open_txt_file = ""
```

```
"<form action='#' method='get'>"
  "<div>"
    "<label for='txtName'>Название файла</label>"
    "<input name='txtName' id='txtName' value='DB.txt'>"
  "</div>"
  "<div>"
    "<button>Открыть</button>"
  "</div>"
"</form>";
const String html_end = ""
"</body>"
"</html>";
// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

void setup()
{
  Serial.begin(115200);
#ifdef DEBUG
  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print(F("Connecting to "));
  Serial.println(ssid);
#endif
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
#ifdef DEBUG
    Serial.print(F("."));
#endif
```

```
  }
#ifdef DEBUG
  Serial.println();
  Serial.println(F("WiFi connected"));
#endif
  // Start the server
  server.begin();
#ifdef DEBUG
  Serial.println(F("Server started"));
  // Print the IP address
  Serial.println(WiFi.localIP());
#endif
  ////////////////////////////////////////////
  /*Flash*/
#ifdef DEBUG
  Serial.print("Initializing SD card...");
#endif
  if (!SD.begin(4))
  {
#ifdef DEBUG
    Serial.println("initialization failed!");
#endif
    return;
  }
#ifdef DEBUG
  Serial.println("initialization done.");
#endif
  root = SD.open("/");
#ifdef DEBUG
  Serial.println("done!");
#endif
  html_table = html_table_start + GetDataBase() + html_table_end;
}
```

```
void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  int status = 0;
  String temp1 = "";

  if (!client) {
    return;
  }
#ifdef DEBUG
  Serial.println(F("new client"));
#endif
  client.setTimeout(5000); // default is 1000
  // Read the first line of the request
  String req = client.readStringUntil('\r');
#ifdef DEBUG
  Serial.println(F("request: "));
  Serial.println(req);
#endif
  // Match the request
  int val = 0;
  if (req.indexOf(F("/gpio/0")) != -1) {
    val = 0;
  } else if (req.indexOf(F("/gpio/1")) != -1) {
    val = 1;
  }
    else if (req.indexOf(F("/?add=%D0%94%D0%BE%D0%B1%D0%B0%D0%B2%D0%B8%D1%82%D1%8C+%D1%83%D1%81%D1%82-%D0%B2%D0%BE")) != -1) {
    status = 1;
  }
```

```
else if (req.indexOf(F("/?flash=%D0%9F%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D1%8C+%D1%84%D0%BB%D1%8D%D1%88")) != -1) {
status = 2;
}
else if (req.indexOf(F("/?ID=")) != -1) {
status = 3;
temp1 = req.substring(req.indexOf(F("?")) + 1, req.indexOf(F("HTTP/1.1")));//, ); // between sign '?' and ' ' (spacebar) 'cause request are like : GET /?street=0&house=10 HTTP/1.1
}

///?change=%D0%98%D0%B7%D0%BC%D0%B5%D0%BD%D0%B8%D1%82%D1%8C+%D1%83%D1%81%D1%82-%D0%B2%D0%BE
else if (req.indexOf(F("/?change=%D0%98%D0%B7%D0%BC%D0%B5%D0%BD%D0%B8%D1%82%D1%8C+%D1%83%D1%81%D1%82-%D0%B2%D0%BE")) != -1) {
status = 4;
}
else if (req.indexOf(F("/?oldID=")) != -1) {
status = 5;
temp1 = req.substring(req.indexOf(F("?")) + 1, req.indexOf(F("HTTP/1.1")));//, ); // between sign '?' and ' ' (spacebar) 'cause request are like : GET /?street=0&house=10 HTTP/1.1
}

///?print_txt=%D0%9F%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D1%8C+%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9+%D1%84%D0%B0%D0%B9%D0%BB
else if (req.indexOf(F("/?print_txt=%D0%9F%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D1%8C+%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9+%D1%84%D0%B0%D0%B9%D0%BB")) != -1) {
status = 6;
}
else if (req.indexOf(F("/?txtName=")) != -1){
status = 7;
temp1 = req.substring(req.indexOf(F("=")) + 1, req.indexOf(F("HTTP/1.1")));
}
```

```
    else if (req.indexOf(F("/?remove=%D0%A3%D0%B4%D0%B0%D0%BB%D0%B8%D1%82%D1%8C+%D1%83%D1%81%D1%82-%D0%B2%D0%BE")) != -1) {
      status = 8;
    }
    else if (req.indexOf(F("/?removeID=")) != -1) {
      status = 9;
      temp1 = req.substring(req.indexOf(F("=")) + 1,  req.indexOf(F("HTTP/1.1")));
    }
      else {
  #ifdef DEBUG
      Serial.println(F("invalid request"));
  #endif
    }
    // read/ignore the rest of the request
    // do not client.flush(): it is for output only, see below
    while (client.available()) {
      // byte by byte is not very efficient
      client.read();
    }

    // Send the response to the client
    // it is OK for multiple small client.print/write,
    // because nagle algorithm will group them into one single packet
    if (status == 0){
      client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
      html_table = html_table_start + GetDataBase() + html_table_end;
      client.print(html_head + html_main_up + html_table + html_button_menu + html_end);
    }
    else if (status == 1){
      client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
      html_table = html_table_start + GetDataBase() + html_table_end;
      client.print(html_head + html_main_up + html_table + html_button_menu + html_form_add  + html_end);
```

```
}
else if (status == 2){
  client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
  client.print(html_head + "<h3>Текуща файлова система: </h3>" + printDirectory(root, 0) + html_end);
}
else if (status == 3){
  client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
  String ID     = temp1.substring(temp1.indexOf(F("=")) + 1,  temp1.indexOf(F("&")));
  String street = temp1.substring(temp1.indexOf(F("&")) + 8, temp1.lastIndexOf(F("&")));
  String house  = temp1.substring(temp1.lastIndexOf(F("=")) + 1);
  client.print(html_head + "<p>"+ temp1 +"</p>" + "<p>"+ ID +"</p>" + "<p>"+ street +"</p>" + "<p>"+ house +"</p>" + html_end);
  WriteToDataBase(ID,  street, house);
}
else if (status == 4){
  client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
  client.print(html_head + html_main_up + html_table + html_button_menu + html_form_change  + html_end);
}
else if (status == 5){
  String ID     = temp1.substring(temp1.indexOf(F("=")) + 1,  temp1.indexOf(F("&")));
  String street = temp1.substring(temp1.indexOf(F("&")) + 8, temp1.lastIndexOf(F("&")));
  String house  = temp1.substring(temp1.lastIndexOf(F("=")) + 1);
  client.print(html_head + "<p>"+ temp1 +"</p>" + "<p>"+ ID +"</p>" + "<p>"+ street +"</p>" + "<p>"+ house +"</p>" + html_end);
  ChangeDeviceSettings(ID, street, house);
}
else if (status == 6)
{
  //html_form_open_txt_file
  client.print(html_head + html_main_up + html_table + html_button_menu + html_form_open_txt_file  + html_end);
}
```

```
else if (status == 7)
{
  //html_form_open_txt_file
  client.print(html_head + GetTxtFile(temp1) + html_end);
}
else if (status == 8)
{
  client.print(html_head + html_main_up + html_table + html_button_menu +
html_form_remove + html_end);
}
else if (status == 9)
{
  client.print(html_head + "<p>"+ temp1 +"</p>" + html_end);


  RemoveDevice(temp1);
}
else
{   client.print(F("HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"));
  client.print(html_head + "<h1>error</h1>"  + html_end);
}
status = 0;
// The client will actually be *flushed* then disconnected
// when the function returns and 'client' object is destroyed (out-of-scope)
// flush = ensure written data are received by the other side
#ifdef DEBUG
Serial.println(F("Disconnecting from client"));
#endif
}
String printDirectory(File dir, int numTabs)
{
String temp = "";
while (true)
{
```

```
    File entry =  dir.openNextFile();
    if (! entry)
    {  // no more files
      break;
    }
    for (uint8_t i = 0; i < numTabs; i++)
    {
      temp += "\t";
    }
    temp += "<p>";
    temp += entry.name();
    if (entry.isDirectory())
    {
      temp += "/";
      temp += printDirectory(entry, numTabs + 1);
    }
    else {
      // files have sizes, directories do not
      temp += "\t\t";
      temp += entry.size();
    }
    temp += "</p>";
    entry.close();
  }
  return temp;
}
String ReadFromFile(String path)
{
  String temp = F("");
  File myFile = SD.open(path);
  if (myFile)
  {
    while (myFile.available())
```

```
    {
      //Serial.println(temp);              // for test
      temp += char(myFile.read());
    }
#ifdef DEBUG
    Serial.println("readAll:\n " + temp);
#endif
    myFile.close();
  }  else
  {  // if the file didn't open, print an error:
#ifdef DEBUG
    Serial.println("error opening " + path);
#endif
  }
  return temp;
}

void WriteToFile(String path, String text)
{
  File  myFile = SD.open(path, FILE_WRITE);
  if (myFile)
  {
#ifdef DEBUG
     Serial.print("Writing to " + path);
#endif
     myFile.print(text);
     // close the file:
     myFile.close();
#ifdef DEBUG
     Serial.println("done.");
#endif
  }
  else
```

```
  {
#ifdef DEBUG
    // if the file didn't open, print an error:
    Serial.println("error opening " + path);
#endif
  }
}

void WriteToDataBase(String ID, String street, String house)
{
  String oldFile = ReadFromFile(nameDataBase);
  String newLine = ID + "\t" + street + "\t" + house + "\t\n";
  WriteToFile(nameDataBase, oldFile + newLine);
}

String GetDataBase()
{
#ifdef DEBUG
  Serial.println(F("Test 1"));
#endif
  String file = ReadFromFile(nameDataBase);
#ifdef DEBUG
  Serial.println("File:\n " + file);
#endif
  int i = 0;
  String temp_buff_file  = F("");
  String temp_buff_line  = F("");
  String temp_buff_param = F("");
#ifdef DEBUG
  Serial.println(F("Test 2"));
#endif
  while (file[i] != '\0')
  {
```

```
      if (file[i] == '\t')
      {
        temp_buff_param = "<td>" + temp_buff_param + "</td>";
        temp_buff_line += temp_buff_param;
        temp_buff_param = F("");
      }
      else if (file[i] == '\n')
      {
        if (temp_buff_line != "")
          temp_buff_line = "<tr>" + temp_buff_line + "<td>1</td>" + "</tr>";

        temp_buff_file += temp_buff_line;
        temp_buff_line = "";
      }
      else temp_buff_param += String(file[i]);

      //Serial.println(F("p:") + temp_buff_param + F("l:") + temp_buff_line + F("f:") +
temp_buff_file);

      i++;
    }
  #ifdef DEBUG
    Serial.println(F("Test 3"));
    Serial.println(temp_buff_file);
  #endif

    return temp_buff_file;
  }
  void ChangeDeviceSettings(String ID, String street, String house)
  {
    String old_file = ReadFromFile(nameDataBase);
    int32_t i = 0;
    uint8_t counter = 0;
```

```
  bool isNewLine = false;
#ifdef DEBUG
  Serial.println("Old_file:\n" + old_file);
#endif
  String temp_buff_file  = F("");
  String temp_buff_line  = F("");
  String temp_buff_param = F("");
  while (old_file[i] != '\0')
  {
    if (old_file[i] == '\t')
    {
      if (counter == 0 && temp_buff_param == ID)
      {
        temp_buff_line = ID + "\t" + street + "\t" + house + "\t\n";
        temp_buff_file += temp_buff_line;

        temp_buff_line = "";
        counter = 0;
        isNewLine = true;
      }
      else temp_buff_param += "\t";
      if (!isNewLine)
        temp_buff_line += temp_buff_param;
      temp_buff_param = "";
      counter++;
    }
    else if (old_file[i] == '\n')
    {
      if ((!isNewLine))
        temp_buff_line += "\n";
#ifdef DEBUG
      Serial.println("Change_func temp_buff_line:\n" + temp_buff_line);
#endif
```

```
      temp_buff_file += temp_buff_line;
      isNewLine = false;
      temp_buff_line = "";
      counter = 0;
    }
    else
    {
      if (!isNewLine)
        temp_buff_param += String(old_file[i]);
    }
    i++;
  }

#ifdef DEBUG
  Serial.println(temp_buff_file);
#endif
  WriteToFile(nameDataBase, temp_buff_file);
}
void RemoveDevice(String ID)
{
  String old_file = ReadFromFile(nameDataBase);
  int32_t i = 0;
  uint8_t counter = 0;
  bool isRemoveLine = false;
#ifdef DEBUG
  Serial.println("Removing from database:\n");
  Serial.println("Old_file:\n" + old_file);
#endif
  String temp_buff_file  = F("");
  String temp_buff_line  = F("");
  String temp_buff_param = F("");
  while (old_file[i] != '\0')
  {
```

```
      if (old_file[i] == '\t')
      {
        if (counter == 0 && temp_buff_param + " " == ID)  // remove in future
        {
          isRemoveLine = true;
        }
        temp_buff_param += "\t";
        temp_buff_line += temp_buff_param;
        temp_buff_param = "";
        counter++;
      }
      else if (old_file[i] == '\n')
      {
        temp_buff_line += "\n";
        //Serial.println("Remove_func temp_buff_line:\n" + temp_buff_line + "IsRemoveLine:\n" + isRemoveLine);
        if (temp_buff_line != "" && isRemoveLine == false)
          temp_buff_file += temp_buff_line;
        isRemoveLine = false;
        temp_buff_line = "";
        counter = 0;
      }
      else
      {
        temp_buff_param += String(old_file[i]);
      }
      i++;
    }
  #ifdef DEBUG
    Serial.println(temp_buff_file);
  #endif
    WriteToFile(nameDataBase, temp_buff_file);
  }
```

```
String GetTxtFile(String path)
{
  String file = ReadFromFile(nameDataBase);
  String line = "";
  String new_file = "";
  int32_t i = 0;

  while (file[i] != '\0')
  {
    if (file[i] == '\t')
      line += "    "; // html doesn't use '\t'

    else if (file[i] == '\n')
    {
      if (line != "")
      {
        line = "<p>" + line + "</p>" ;

        new_file += line;
        line = "";
      }
    }
    else line += file[i];

   i++;
  }

  return new_file;
}
```