МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Інститут післядипломної освіти

Кваліфікаційна робота бакалавра

на тему: **Розробка апаратного та програмного забезпечення пристрою «Паркомат»**

Виконав студент групи **КН-5**
спеціальності 122 Комп'ютерні науки
Попов Іван Олександрович

Керівник **д.т.н., проф. Казакова Надія Феліксівна**

Консультант

Рецензент **Попов В.Л.**
Директор КП Броварської Міської Ради Броварського Району Київської Області "Центр Інформаційних Технологій Міста"

Одеса 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARM – Advanced Risc Machines.
ARP – Address Resolution Protocol.
ASK – Amplitude- Shift Keying
CAN – Controller Area Network.
CGI – Common Gatway Interface.
DLL – Dynamic Link Library.
DMA – Direct Memory Access
EMC – Electo Magnetic Compatibility.
FAT – File Allocation Table
ICMP – Internet Control Messsage Protocol.
IDE – Integrated Development Environment.
IP – Internet Protocol.
LTDC – LCD TFT Controller.
MAC – Media Access Control address.
NFC – Near Field Connection.
PCD – Proximity Coupling Device.
PICC – Proximity Chip Card.
POS – Point Of Sale.
QR – Quick Response Code
RAM – Random Access Memory.
RTOS – Real Time Operation System.
SD – Secure Digital memory card.
TFT – Thin Film Transistor
UART – Universal Asynchronous Receiver-Transmitter.
UDP – User Datagram Protocol.
UID – Unique Identifier.
USB – Universal Serial Bus.
WAV – Waveform file

ВСТУП

На сьогодні в світі існують сотні, якщо не тисячі, аналогічних систем автоматичного паркінгу и всі вони, безумовно, забезпечують базовий функціонал, очікуваний замовниками від такого класу виробів, тому вигадати щось нове або лише додати якийсь додатковий сервіс досить важко. Взагалі, конкуренція між виробниками автоматизованих систем доступу точиться навколо ціни, якості та додаткових сервісів та послуг, таких як великі кольорові дисплеї на стійках, наявність котрих не впливає на основний робочій функціонал виробу, але підвищте естетичну привабливість даного приладу для користувачів. Не зважаючи на широкий вибір подібних систем автоматичного паркінгу, керівництво компанії-виробника було прийнято рішення розробити та збудувати власну систему автоматичного паркінгу, щоб перевірити деякі нестандартні ідеї, яки з'явились у групи інженерів під час обговорення технічних деталей майбутнього проекту.

Традиційно переважна більшість систем автоматичного паркінгу будуються за топологією «**online**». Цей термін означає що робота всіх пристроїв в системі керується з одного центру: перевірка наявності автотранспортного засобу перед стійкою паркомату, пошук в базі даних його реєстраційного державного номера або UID-коду картки, яку водій транспорту приклав до зчитувача, відправка керуючої команди на відкриття шлагбауму, потім відправка керуючої команди на його закриття, коли автомобіль залишить зону дії паркомату – усі ці окремі дії виконуються кінцевими виконуючими механізмами *виключно* за командами від єдиного центру. Вочевидь, що до центрального керуючого комп'ютеру вимоги більш жорсткі, ніж до звичайного офісного комп'ютеру. Тому часто в якості керуючого комп'ютера використовують так звані «промислові комп'ютери», які мають ширший діапазон робочих температур та більш стійкі до впливу деструктивних факторів навколишнього середовища: волога, пил, високі або низькі температури, тиск, вібрації, тощо. Крім

цього, прикладне програмне забезпечення працює під більш надійними операційними системами, ніж Windows. Відповідно, надійна робота цілої системи парковки залежить лише від одного єдиного центрального комп'ютера, працездатність якого є критичною для всієї системи, яка на великих паркінгах може нараховувати десятки в'їзних та виїзних терміналів, та POS-терміналів для сплати. Приймаючі рішення про розробку своєї власної системи автоматичного паркінгу група інженерів розробила топологію, яка не містить єдиного виділеного сервера, який керує усіма виконавчими механізмами. В даної концепції передбачено тільки віддалений сервер баз даних, постійно оновлюючий робочі бази даних, фізично розміщенні в мікросхемах пам'яті в стійках в'їзду та виїзду. До його обов'язків також входить сбір накопиченої всіма стійками інформації по оброблених транзакціях (в'їзд та виїзд автомобілю). Таким чином реалізована розподілена база даних, окремі елементи якої зберігаються в кожній стійки в'їзду або виїзду, а виділений сервер періодично синхронізує свої бази даних з діючими базами в стійках.

1. СИСТЕМИ АВТОМАТИЧНОГО ПАРКІНГУ АВТОТРАНСПОРТУ

1.1. Аналіз базових складових системи автоматичного паркінгу

Система автоматичного паркінгу (рис.1), яка описана в даній роботі, розроблена на замовлення великої торгівельної компанії, власника мережі супермаркетів на півдні України, але може бути встановлена в інших місцях, де потрібна автоматизація автомобільних стоянок: торгівельні центри, аеропорти, вокзали и т.д.

Рисунок 1 – Система автоматичного паркінгу.

Система автоматичного паркінгу це програмно-апаратний комплекс, виконуючий курування, контроль, облік, прийом платежу за стоянку від водіїв транспортних засобів, які в'їжджають на стоянку або виїжджають з стоянки. Структурна схема системи автоматичного паркінгу зображена на рис.1 та містить наступні пристрої:

– Стійки в’їзних паркоматів (Entry Parking ticket machine), які керують виконавчим механізмом - шлагбаумом (Barrier), фізично розташовані на всіх в’їздах до парковки.

– Стійки виїзних паркоматів (Exit Parking ticket machine), які керують виконавчим механізмом - шлагбаумом (Barrier), фізично розташовані на всіх виїздах з парковки.

– POS-термінал (Point Of Sale), автомат для сплати послуг стоянки.

– Сервер баз даних. Виконує керуючу програму, яка крім іншого дозволяє оператору ручне управління шлагбаумами у разі екстреної необхідності. Також зберігає бази даних автомобільних номерів, зафіксованих спеціальною камерою, зберігає номери NFC-карт, керує «білими» та «чорними» списками, тобто списками автомобільних номерів та NFC-карт, яким дозволена або заборонена безкоштовна стоянка, що перевищує безкоштовний ліміт 60 хвилин.

– Стійка в’їзду (Entry Parking ticket machine). Зберігає всі номери та дати талонів з QR кодом, які вона роздрукувала, зберігає всі UID коди NFC-карток, які водії прикладають до зчитувача в’їзної стійки або всі номери автомобілів, які зафіксувала спеціальна камера з розпізнавання номерів.

– Стійка виїзду (Exit Parking ticket machine). Зберігає всі NFC-картки та номери талонів з QR кодом, скановані вбудованими сканерами, під час виїзду транспортного засобу з стоянки. Після сканування NFC-картки або QR талону стійка виїзду звертається до всіх стійок в’їзду з запитом по сканованій інформації, щоб з’ясувати точний час в’їзду даного транспортного засобу та за потреби вимагати від водія додаткової оплати перевищеного безплатного часу.

Описані етапи взаємодії стійок проходять без участі серверу, тому вихід з ладу однієї стійки приведе до втрати лише даних, накопичених тільки однією окремою стійкою, усі інші будуть працювати в нормальному режимі. В той же час при топології «**online**» вихід з ладу або відключення сервера (або його перезавантаження) призведе до колапсу всієї системи. Тестова експлуатація даної системи автоматичного паркінгу з серпня 2021 року по червень 2022 року (на момент написання цього тексту) на стоянці одного з підприємств м.Одеси

підтвердила припущення інженерів про підвищену надійність обраної топології, після чого керівництво підприємства прийняло рішення про промислове виробництво даної системи автоматичного паркінгу.

2 ОПИС ТА РОЗРОБКА СТІЙКІ ПАРКОМАТУ.

Рисунок 2 – Загальний вигляд стійки паркомату, встановленої на паркінгу, м. Одеса, 2021

У дипломній роботі описана конструкція та програмне забезпечення стійки паркомату (Parking Ticket Machine), яка е складовою та невід'ємною частиною системи автоматичного паркінгу. Загальний вигляд стійки паркомату зображений на Рисунок 2, структурна схема В'ЇЗНОЇ стійки паркомату зображена на рис. 3А, ВИЇЗНОЇ стійки – на рис. 3В.

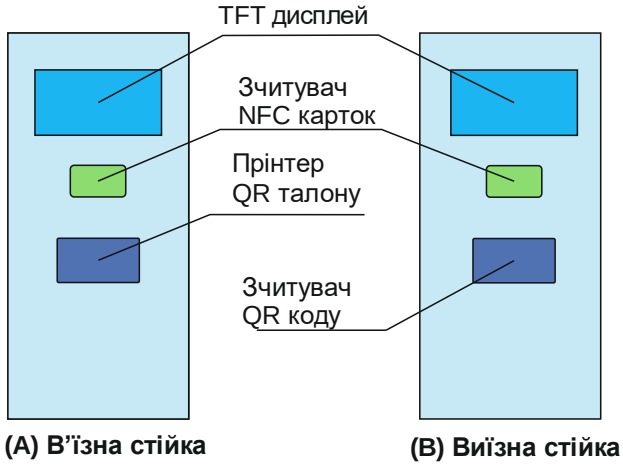

Рисунок 3 – Структурна схема в'їзної (А) та виїзної (В) стійки.

Як видно з структурних схем, різниця між в'їзною та виїзною стійками полягає в тому, що в'їзна стійка має принтер для друку талонів с QR кодом, а виїзна стійка замість принтеру має сканер QR кодів. Програмне забезпечення

спільне для обох варіантів, конкретний режим роботи – чи це стійка виїзду чи в'їзду, - конфігурується з вбудованої веб-сторінки за допомогою любого інтернет-браузеру, або можна використати для конфігурування спеціальне ПО, розроблене на підприємстві. Там же задаються інші параметри стійки: адреси MAC, IP та Gateway в мережі Ethernet, яскравість дисплею, число датчиків присутності автомобіля, час утримання шлагбауму в піднятому положенні, режим роботи вбудованого термостату підігріву електроніки в зимовий час, та інші інженерні параметри. Детальніше конфігураційна веб-сторінка буде розглянута у відповідному розділі. Хоча стійки паркоматів складаються з кількох функціонально закінчених вузлів, в даній роботі будуть розглянуті тільки ті модулі, які є об'єктом інженерної розробки при виготовленні даного виробу підприємством-виробником парковочного комплексу. Вузли та компоненти, виготовлені сторонніми виробниками та придбані як готовий виріб, будуть розглянуті частково, лише для розуміння їхнього загального функціоналу та призначення в даному виробі (рис.4).

Рисунок 4 – Стійки паркоматів в цеху під час виробництва, м. Одеса, 2021.

2.1. Архітектура та апаратні модулі стійки паркомату

2.1.1 Модулі сторонніх виробників

Наступні параграфи містять інформацію щодо функціонально та конструктивно завершених модулів, які були використані в стійках паркомату.

2.1.1.1 Джерело живлення

Джерело живлення постійного струму потужністю 200W серії **LRS-200-24**, забезпечує 24V 8A. Вироблено стороннім виробником та придбане як конструктивно та функціонально закінчений виріб. Продукція відомої в світі компанії Mean Well Enterprise Co., Ltd., Taiwan, добре зарекомендувала себе високою якістю своєї продукції. Джерело живлення є одним з важливіших компонентів всього пристрою, тому що саме від джерела живлення залежить працездатність всієї стійки паркомату. Зовнішній вигляд джерела живлення зображений на фото рис.5, а його технічні параметри наведені в [1].

Рисунок 5 – Блок живлення Mean Well LRS-200-24.

2.1.1.2 Принтер (Kiosk printer)

Принтер для друку чеків на термочутливому папері. Розроблений для роботи в вуличних умовах та в неопалюваних приміщеннях. Забезпечує друк чеку, обрізання паперу, видачу чеку з утриманням та, при необхідності, затягування чеку назад в спеціальний лоток. Виготовлений компанією CUSTOM S.p.A., Fontevivo/Parma (Italy) та придбаний як конструктивно та функціонально завершений виріб. Конкретна модель принтера Custom **VKP80ii-SX** обрана через наявність вбудованого коду для формування QR-кодів. Має інтерфейси USB та RS-232, останній використовується для зв’язку з процесорною

платою. Зовнішній вигляд принтеру зображений на фото рис.6, технічні параметри та умови експлуатації наведені в [2], а система команд в [3].

Рисунок 6 – Принтер для друку чеків CUSTOM VKP80ii-SX.

2.1.1.3 Сканер QR коду (QR and Barcode scanner)

Сканер EM20-80 компанії Newland Europe BV (Netherlands) зчитує з чеку та розпізнає інформацію, закодовану в форматі QR. Виготовлений іншою компанією та придбаний як конструктивно та функціонально завершений виріб. має інтерфейси USB и UART, останній використовується для зв’язку з процесорною платою.

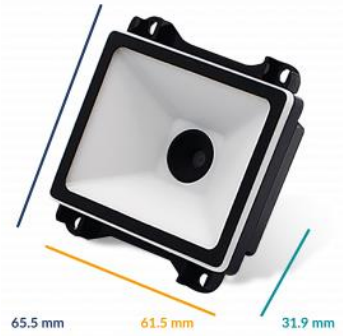

Рис. 7 – Сканер QR коду EM20-80.

Здатен зчитувати та декодувати наступні коди: CODE39, CODE128, EAN-8, UPC-A/EAN-13, ITF-6, ITF-14, GS1, Code11, ISBN, ISSN, PDF417, QR, MicroQR, Aztec, Data Matrix, USPS, KIX, Passport OCR, Aim ID Prefix. Зовнішній вигляд сканеру зображений на фото рис.7, а система команд, інтерфейси, протокол зв’язку та його детальний опис наведений в [4].

2.1.1.4 Індуктивний детектор (Inductive Loop Vehicle Detector)

Індуктивний детектор присутності автомобілю - сенсор присутності металевих об’єктів. Чутливим елементом якого є пара кілець з ізольованих дротів, приховані в під поверхнею дорожнього покриття перед стійкою паркомату. Завдяки двом незалежним виткам своєрідної антени з’являється можливість розпізнати напрямок руху транспортного засобу. Виготовляється багатьма виробникам, один з яких і був обраний як постачальник: CARLO GAVAZZI Automation Components, LDD2PA2DU24. Зовнішній вигляд є на фото рис.8, детальний опис виробу наведений в [5]

Рисунок 8 – Індуктивний детектор присутності транспортного засобу LDD2PA2DU24.

2.1.1.5 Картки MIFARE (MIFARE cards)

Однією з головних вимог замовника була можливість в’їзду до паркінгу за допомогою безконтактних карток NFC класу MIFARE© Classic 1K, MIFARE© Classic 4K, MIFARE© Ultrlight, MIFARE© Ultrlight C. Детальний опис карток, їхні електричні та експлуатаційні параметри детально описані в [6], [7], [8], [9]. Зовнішній вид карток-заготовок, призначених для нанесення власного зображення замовником-емітентом, наведений на рис. 9.

Рисунок 9 – Картки MIFARE Classic 1K

Картки MIFARE© передають та приймають дані дистанційно, дистанція залежить від геометрії та розміру антен та особливості зчитувача. Живлення для роботи мікросхеми, прихованої в середині картки, отримує від передавача, так зване «паразитне живлення».

2.2 Модулі власного виробництва

Наступна інформація стосується модулів власного виробництва, які були розроблені виключно для цього проекту. До них входить плата процесору, плата NFC зчитувача та плата реле.

Головне завдання інженера-конструктора це втримання компромісу серед багатьох вимог. Головні вимоги: технічне завдання, отримане від замовника, собівартість виробу, доступність (наявність на ринку) ключових компонентів та швидкість їх придбання, кваліфікація наявних інженерів, тобто вміння та досвід інженерів створювати подібні пристрої за визначений термін та з потрібною якістю, наявність виробничих потужностей та кваліфікація працівників в цехах, де планується будувати розроблені інженерами вироби.

2.2.1 Плата безконтактного зчитувача

Цей параграф присвячений детальному огляду плати NFC зчитувача.

Розробка плати NFC зчитувача зводиться до витримування вимог виробника мікросхеми безконтактного зчитування CLRC663 та розрахунок топології антени, виробленої на поверхні друкованої плати.

2.2.1.1 **Основні принципи обміну даними з карткою**

Стандарт ISO/IEC 14443 визначає скорочення, іменуючи учасників обміну даними між двома пристроями. Так, активна сторона - зчитувач (завжди ініціатор обміну даними), називається PCD (Proximity Couple Device – Безконтактний Парний Пристрій). В свою чергу картки мають назву, згідно стандарту, PICC (Proximity Chip Card – Безконтактний Чип на Картці). Стандарт ISO/IEC 14443 визначає частотний діапазон, метод модуляції та протокол обміну даними з безконтактними пасивними пристроями, тобто картками. Картки MIFARE© підтримують протокол ISO14443-3A [10]. Обмін даними здійснюється на частоті 13.56MHz. Модуляція передавача PCD – 100% ASK, модифікований Код Міллера. Модуляція приймача, тобто пасивної картки (PICC), здійснюється навантажувальним резистором, піднесуча частота 847.5kHz, OOK код Манчестера, напівдуплексний, зчитувач завжди ініціює обмін даними, рис.10, рис.11. З огляду на пасивність картки стає зрозумілим, що передача енергії від зчитувача до картки є необхідністю, без якої обмін даними між зчитувачем на карткою стає неможливим. Далі будуть представлені лише фундаментальні принципи MIFARE© стандарту та описані базові принципи передачі енергії. Також будуть показані принципи обміну даними між зчитувачем та карткою.

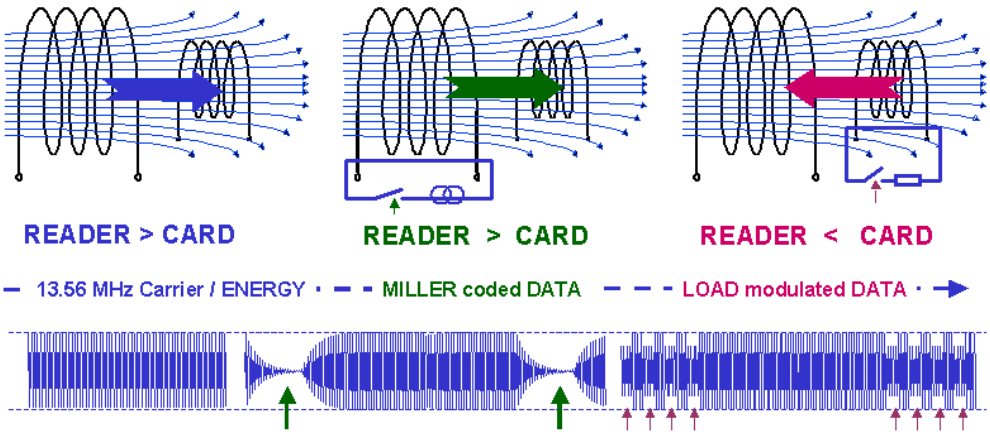

Рисунок 10 – Обмін даними між зчитувачем (PCD) та карткою (PICC)

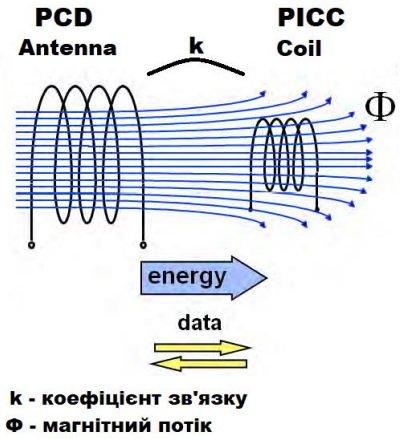

Рисунок 11 – Магнітний зв’язок між зчитувачем (PCD) та карткою (PICC)

Передача енергії від зчитувача PCD до пасивної картки PICC (пасивною вважається картка, яка не має свого власного джерела енергії) базується на принципі трансформатору. З боку зчитувача (тобто передавача) виготовляється антена з декількох обертів мідного провідника на поверхні друкованої плати. Така ж сама антена виготовляється з боку картки. Еквівалентну схему можна знайти на рис. 12:

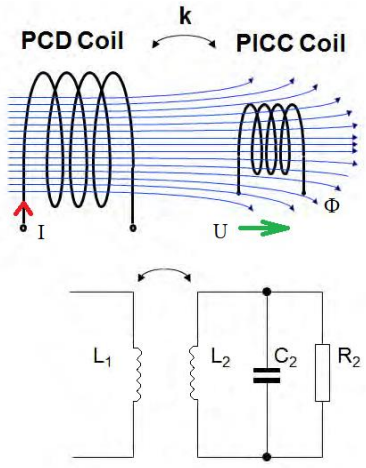

Рисунок 12 – Схема еквівалентного трансформатору між зчитувачем (PCD) та карткою (PICC)

На рис.12 верхня частина схеми відображає антени зчитувача та картки під час передачі енергії. Електричний струм **I** в кільцях антени PCD генерує магнітний потік **Φ**, частина цього потоку проходить скрізь антену картки PICC та генерує у її витках напругу **U**. Після її випрямлення та досягнення заданого значення картка починає працювати. Індукована напруга може змінюватися в залежності від дистанції між антенами PCD та PICC. Таким чином максимальна дистанція лімітується індукованою енергією. Нижня частина рис.12 відображає еквівалентну електричну схему моделі трансформатору.

2.2.1.2 **Розрахунок розміру антени**

Розрахунок розміру антени стосується лише антени зчитувача PCD, тому що розміри антен карток PICC вже завдані виробником карток. Довжина сторони антени, або її діаметр, якщо форма антени є коло, може бути від 6 до 12см. Експериментально отриманий графік рис.13 вказує на зв’язок між розміром антени та робочою дистанцією [11].

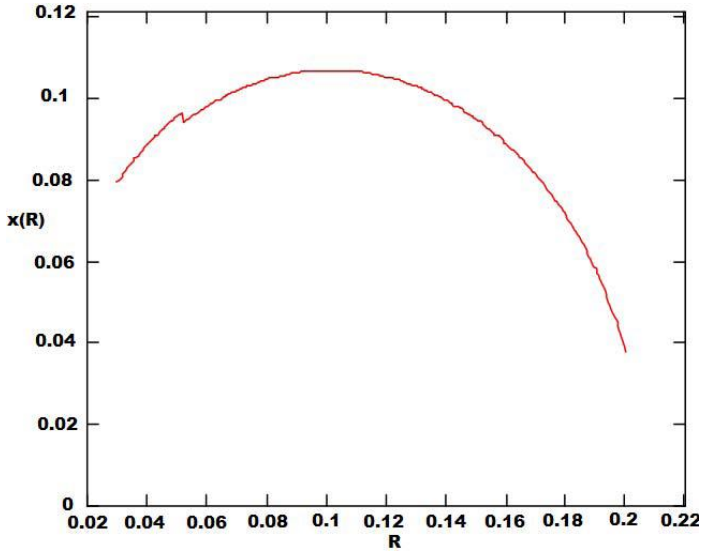

Рисунок 13 – Залежність робочої відстані від радіусу антени.

Рис.13 дає приблизну відстань надійної роботи з карткою, в залежності від розміру антени. З цього графіку ми можемо бачити що найкраще зчитування або запис (**x(R)**) можна досягнути при розмірі кругової антени з радіусом 10см (**R=10**).

2.2.1.3 **Схема узгодження**

Найважливішим етапом розробки схеми та друкованої плати антени є вибір компонентів узгодження антени та вхідного/вихідного каскадів інтерфейсної мікросхеми. Для роботи з картками була вибрана спеціальна мікросхема безконтактного зчитування CLRC663 (виробник Philips/NXP), яка підтримує більшість сучасних протоколів, включно з потрібним протоколом MIFARE. Для написання цієї глави використані матеріали, наведені в [12], [13]. Детальніше з мікросхемою CLRC663 можна ознайомитись в [14].

Узгодження антен з імпедансом 50Ω дуже важливе для передачі в картку максимальної потужності. Повна схема зв’язку мікросхеми та антени для читання та запису в картку містить компоненти EMC (Electromagnetic Compatibility), елементи узгодження та, власне, саму антену. Антена може бути замінена

еквівалентною схемою з послідовним включенням елементів, як зображено на рис.14.

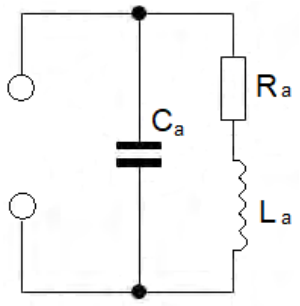

Рисунок 14 – Схема послідовного еквівалента антени.

Параметри компонентів можуть бути легко знайдені аналізатором імпедансу або мережевим аналізатором. Однак, в першому наближенні індуктивність (**L_a**), резистивний опір (**R_a**) та ємність (**C_a**) антени можуть бути обчислені за допомогою (1), (2), (3):

$$L_a[nH] = 2l_1[cm]\left(\ln\left(\frac{l_1}{D_1} \right) - K \right) N_1^{1.8} \tag{1}$$

l_1 = довжина провідника кільця антени.

D_1 = Діаметр провідника або ширина доріжки на друкованій платі.

K = коефіцієнт: 1.07 для кругових антен, 1.47 для прямокутних антен.

N = кількість витків антени

ln = функція натурального логарифму

$$R_a = R_{dc} \tag{2}$$

R_{dc} = резистивний опір антени.

$$C_a = \frac{1}{(2\pi f_{ra})^2 L_a} \tag{3}$$

f_{ra} = власна частота резонансу антени

Добротність антени може бути отримана з (4)

$$Q_a = \frac{2\pi f_{ra} L_a}{R_a} \tag{4}$$

Якщо отримана добротність $\mathbf{Q_a}$ вища за бажане значення 30, то потрібно встановити демпферний резистор $\mathbf{R_Q}$, який знизить добротність антени до потрібних 30 (в межах ±10%). Номінал резистора можна отримати з (5)

$$R_Q = 0.5\left(\frac{2\pi f_{ra} L_a}{30}\right) - R_a \tag{5}$$

Паралельний еквівалент антени, разом з демпферним резистором, зображений на рис.15. $\mathbf{L_{pa}}$, $\mathbf{C_{pa}}$, $\mathbf{R_{pa}}$ можуть бути обчислені за допомогою (6), (7), (8).

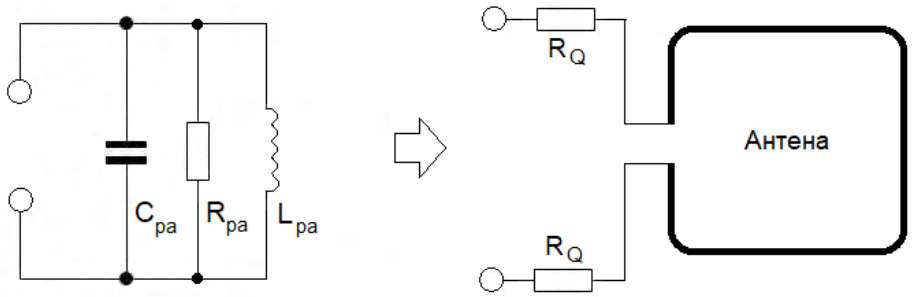

Рисунок 15 – Схема паралельного еквівалента антени.

$$L_{pa} \cong L_a \tag{6}$$

$$C_{pa} \cong C_a \tag{7}$$

$$R_{pa} \cong \frac{\left(2\pi f_{ra} L_a\right)^2}{R_a + 2R_Q} \tag{8}$$

Принципова схема запису/зчитування містить мікросхему безконтактного зчитування CLRC663, елементи узгодження антени з мікросхемою та

елементи фільтру електромагнітної сумісності. Разом з еквівалентом антени це зображено на рис.16.

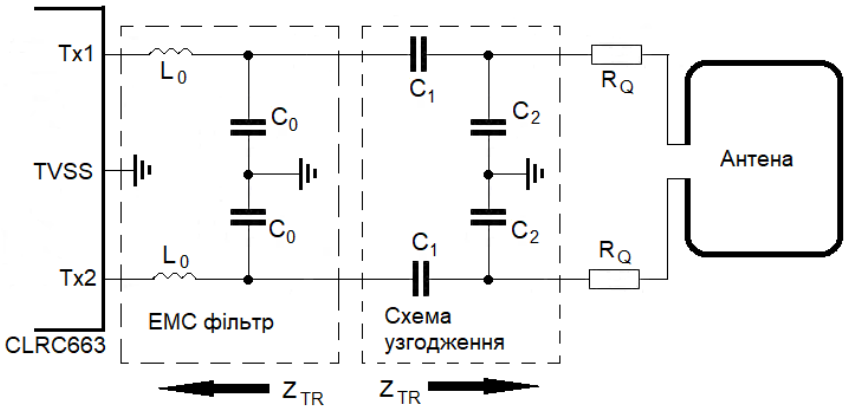

Рисунок 16 – Схема запису та читання.

Схема EMC фільтру використана як фільтр сигналу та як трансформатор імпедансу. Головні завдання перетворювача імпедансу це зменшення часу нарощування сигналу після фази модуляції та збільшення пропускної здатності при прийомі сигналу. Змінна $\mathbf{F_{r0}}$ була обрана на рівні 21MHz, номінал C_0 можна обчислити з (9). Значення 1μH обране для L_0, та 22pF обране для C_0 в колах EMC фільтру.

$$C_0 = \frac{1}{(2\pi f_{r0})^2 L_0} \tag{9}$$

EMC фільтр та схема узгодження мусять перетворити імпеданс антени до відповідного опору $\mathbf{Z_{match}(f)}$ на робочій частоті 13.56MHz. Виміряне значення $\mathbf{Z_{match}(f)}$ може бути змодельоване еквівалентною схемою навантаження для кожного піну **Tx** мікросхеми безконтактного зчитувача CLRC663 $\mathbf{R_{match}/2}$. При розділенні схеми після EMC фільтру, щоб обчислити номінали решти схеми, необхідною передумовою є відоме значення $\mathbf{R_{match}/2}$.

$$Z_{tr} = R_{tr} + jX_{tr} \tag{10}$$

$$Z_{tr}^{*} = R_{tr} + jX_{tr} \tag{11}$$

$$\omega = 2\pi f \tag{12}$$

$$R_{tr} = \frac{R_{match}}{\left(1-\omega^2 L_0 C_0\right)^2 + \left(\omega \frac{R_{match}}{2} C_0\right)^2} \tag{13}$$

$$X_{tr} = 2\omega \frac{L_0\left(1-\omega^2 L_0 C_0\right) - \frac{C_0 R_{match}^2}{4}}{\left(1-\omega^2 L_0 C_0\right)^2 + \left(\omega \frac{R_{match}}{2}\right)^2} \tag{14}$$

$$C_1 \approx \frac{1}{\omega\left(\sqrt{\frac{R_{tr} R_{pa}}{4}} + \frac{X_{tr}}{2}\right)} \tag{15}$$

$$C_2 \approx \frac{1}{\omega^2 \frac{L_{pa}}{2}} - \frac{1}{\omega\sqrt{\frac{R_{tr} R_{pa}}{4}}} - 2C_{pa} \tag{16}$$

Нарешті, в схемі узгодження експериментально були випробувані багато номіналів конденсаторів. Найкращі результати були отримані з номіналом 50pF для конденсатора $\mathbf{C_1}$. Конденсатор $\mathbf{C_2}$ під час експерименту та налаштування антени був розташований на друкованій платі біля антени та складався з двох послідовно включених конденсаторів: перший зі сталою ємністю 150pF, другий змінний, ємністю 4-20pF. Змінний конденсатор використовувався для точного налаштування антени під час підбору номіналів, які потім, під час автоматичного монтажу схем на підприємстві, були замінені на сталі номінали. Під час експериментів було виготовлено більше 5 екземплярів антен на друкованих платах з різними геометричними розмірами, поки не було знайдено оптимальні параметри геометрії антени та номіналів компонентів, котрі, потім, нарешті і були використані в фінальному зразку плати зчитувача.

2.2.1.4 **Кількість витків антени**

Зміна кількості витків в антені не змінює коефіцієнту зв'язку між антенами, тому що індуктивність не впливає на зв'язок, але кількість витків впливає на індуктивність антени. На рис.17 можна побачити експериментально отримані дані, які зв'язують число витків антени з радіусом одного витку:

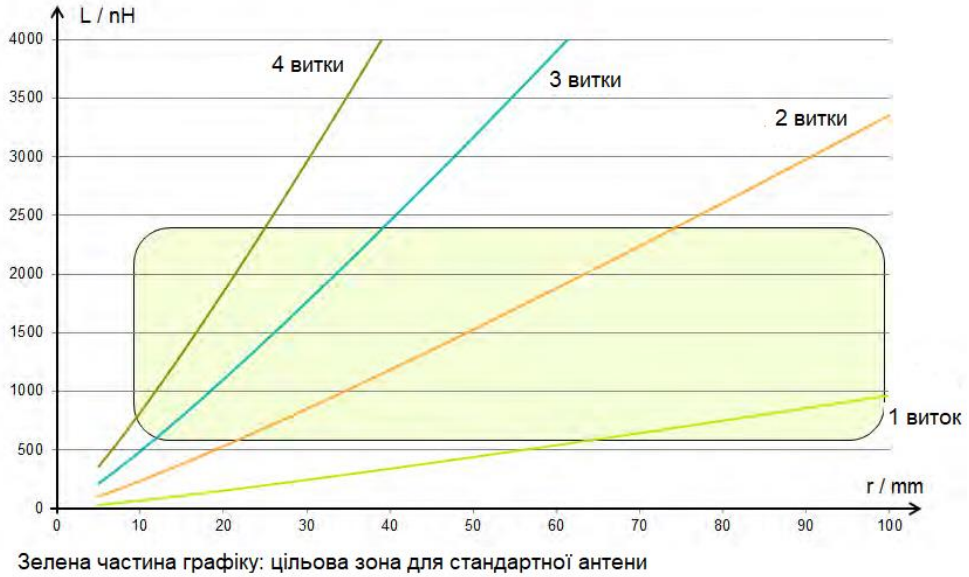

Рисунок 17 – Приклад залежності індуктивності від радіусу.

2.2.1.5 **Кодування даних від зчитувача до картки**

Для передачі даних між зчитувачем та карткою використовується напівдуплексна передача (half duplex structure). Обмін завжди ініціює зчитувач PCD, «зчитувач говорить першим». Передавання даних від зчитувача до картки здійснюється згідно з ISO14443 Type A використовуючи 100% ASK модуляцію. Рис.18 ілюструє типову форму сигналу.

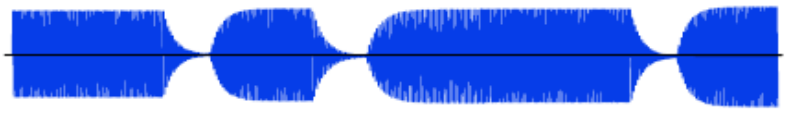

Рисунок 18 – Типовий сигнал передавання даних між зчитувачем (PCD) та карткою (PICC).

Добротність індуктивності антени Q змінює форму переданого сигналу який показаний на рис.18. Ця форма може бути використана для вимірювання налаштування антени. Детальніше з цім можна ознайомитись в [15].

З огляду на пасивне живлення картки PICC, зчитувач PCD мусить забезпечити картку достатньою енергією на весь час обміну даними між PCD та PICC. Отже MIFARE© (ISO14443) використовує оптимізоване кодування щоб забезпечити постійний рівень енергії незалежно від даних, які передаються від PCD до PICC. Це модифікований код Міллера, показаний на рис.19.

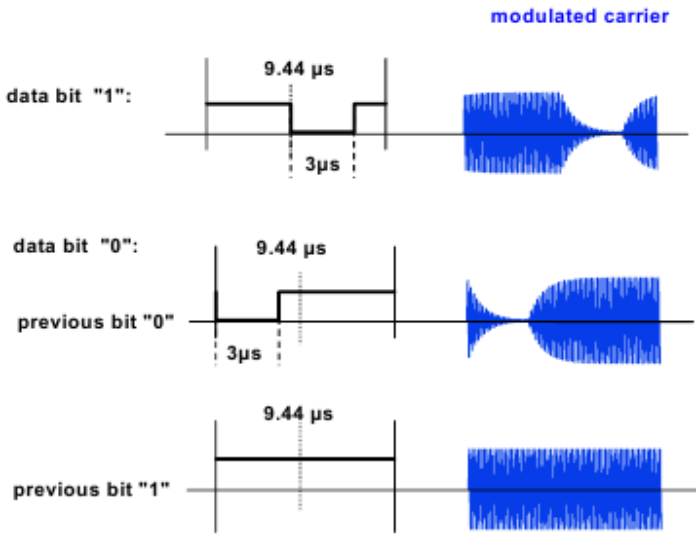

Рисунок 18 – Передача даних від зчитувача (PCD) на картку (PICC), код Міллера.

Зображені на рис.19 часові інтервали для швидкості 105.9kHz, отже довжина бітового інтервалу 9.44uS. Імпульс коду Міллера має довжину 3uS. Логічна ‘1’ виражена імпульсом в середині бітового інтервалу.

Для кодування логічного ‘0’ існує два варіанти, це залежить від попереднього біту:

Якщо попередній біт був ‘0’, наступний ‘0’ відображається імпульсом 3uS у перші частині бітового інтервалу.

Якщо попередній біт був ‘1’, наступний ‘0’ відображається без імпульсу взагалі на протязі всього бітового інтервалу.

2.2.1.6 **Передавання даних від картки до зчитувача**

Передавання даних у зворотному напрямку, тобто від карки PICC до зчитувача PCD, використовує принцип модуляції навантаженням, приклад зображений на рис.20.

2.2.1.7 **Принцип модуляції під несучої**

Картка PICC розроблена як резонансний контур та використовує енергію, яка передана зчитувачем PCD. *Споживання наведеної енергії має зворотній ефект як падіння напруги на боці зчитувача PCD. Цей ефект використовується для передачі даних від картки назад до зчитувача, формуючі навантаження в мікросхемі картки*.

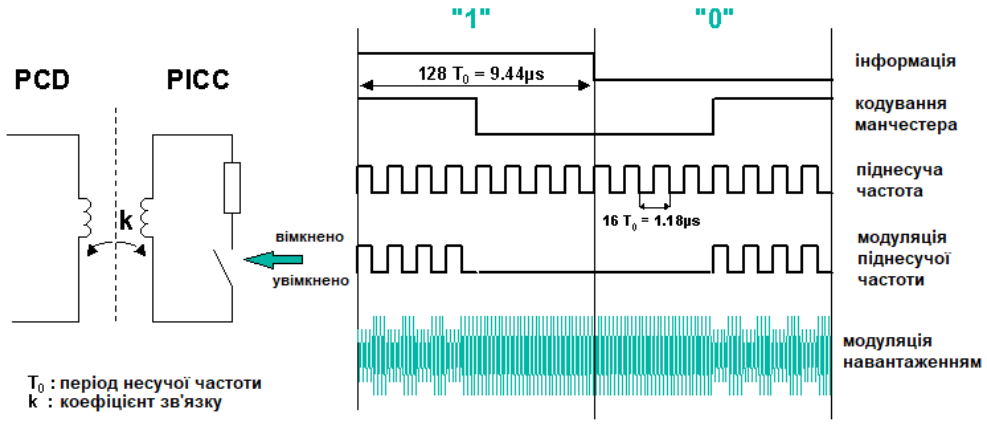

Рисунок 20 – Принцип модуляції під несучої частоти навантаженням

Антена зчитувача PCD налаштована на резонансну частоту f_R = 13.56 MHz. Довжина періоду T_0 цієї частоти дорівнює $T_0 = 1/f_R \approx 74$nS. Насправді резонансна схема антени PCD генерує амплітуду в декілька разів вищу за рівень напруги живлення. Через досить малий зв’язок між антенами PCD та PICC, картка може сформувати сигнал на 60dB нижче ніж напруга на антені

зчитувача. Щоб прийняти такий слабкий сигнал потрібно ретельно спроектувати вхідні каскади зчитувача.

Картка PICC передає дані на зчитувач з швидкістю 105.9kbit/s використовуючи кодування Манчестера. В коді Манчестера кожен біт представлений або підйомом або падінням рівня напруги в середині бітового інтервалу. На рис.20 зображено принцип модуляції для технології MIFARE© , права частина малюнка:

Логічна «1» передається переходом від високого рівня до низького в середині бітового інтервалу.

Логічний «0» передається переходом від низького рівня до високого в середині бітового інтервалу.

Ці данні, кодовані кодом Манчестера, модулюють піднесучу частоту $f_{SUB} = \frac{f_R}{16} = 847.5kHz$.

Нарешті, ця модульована піднесуча маніпулює (вмикає та вимикає) навантаженням в картці PICC, фактично, маніпулює звичайним резистором, що в результаті призводить до модуляції навантаженням, яке показане на останньому графіку на рис.19, яка знову приймається та декодується зчитувачем PCD.

2.2.1.8 Принципова схема плати зчитувача

Результатом обчислювань, наведених в попередньому параграфі, є друкована плата. Принципова схема для швидкого ознайомлення наведена на рис. 21, детальні схеми з максимальною якістю та шари друкованої плати в форматі PDF прикладені до файлового додатку.

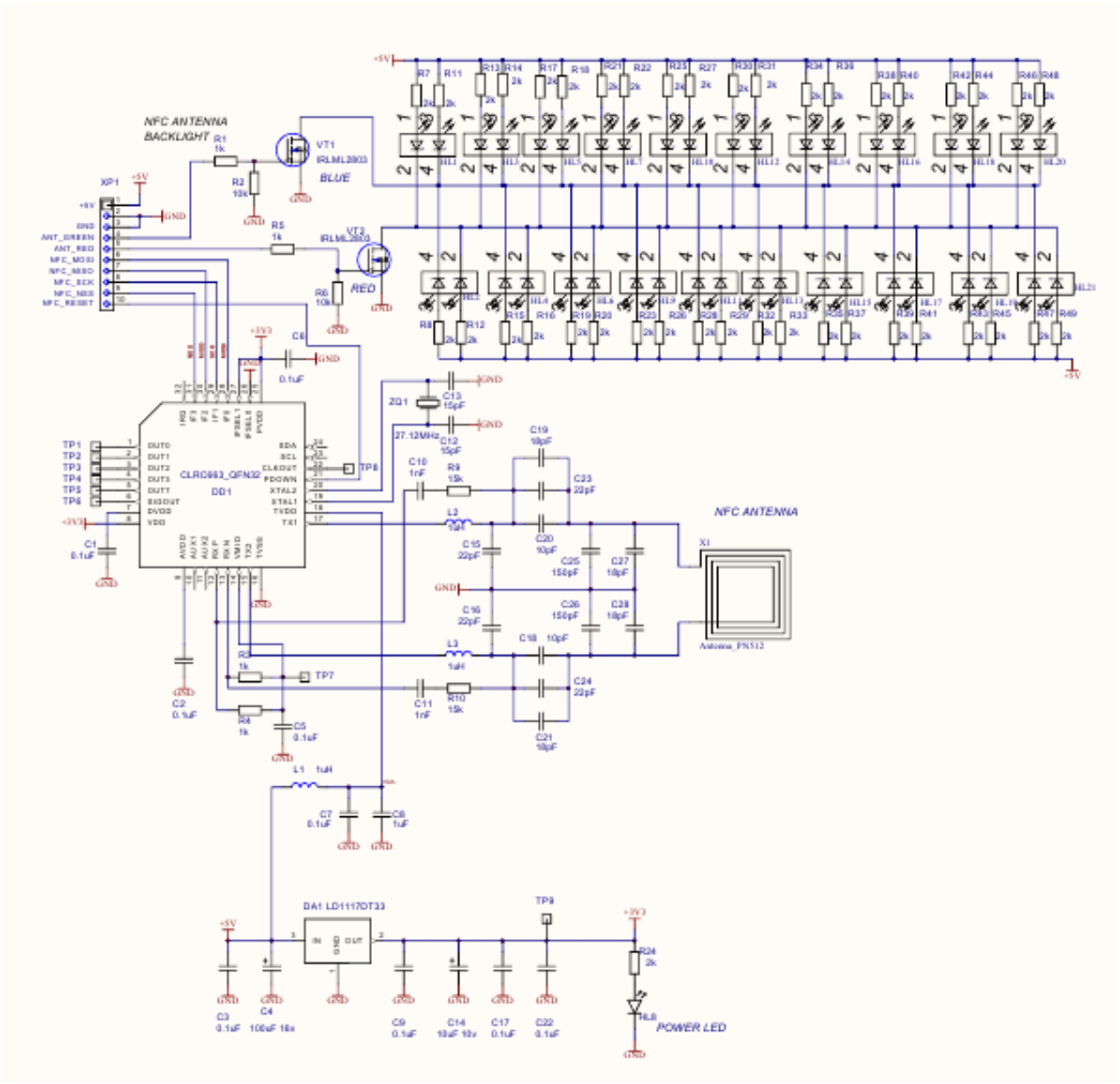

Рисунок 21 – Принципова електрична схема плати безконтактного зв’язку з картками NFC.

На електричній схемі, окрім власне мікросхеми безконтактного зчитувача CLRC663 та елементів, необхідних для її роботи, ще є світлодіоди червоного та зеленого кольору, та транзистори для їхньої комутації, які формують підсвічування зони картки для зручного користування паркоматом у нічний час та формування естетичного вигляду. Стабілізатор напруги знижує підведені до плати 5V до рівня 3.3V, потрібного мікросхемі. Зверніть увагу, що вихідний каскад мікросхеми має окремий вхід живлення, на який можна подати

до 5 вольт, що підвищує вихідну потужність, але ядро мікросхеми потребує лише 3.3V.

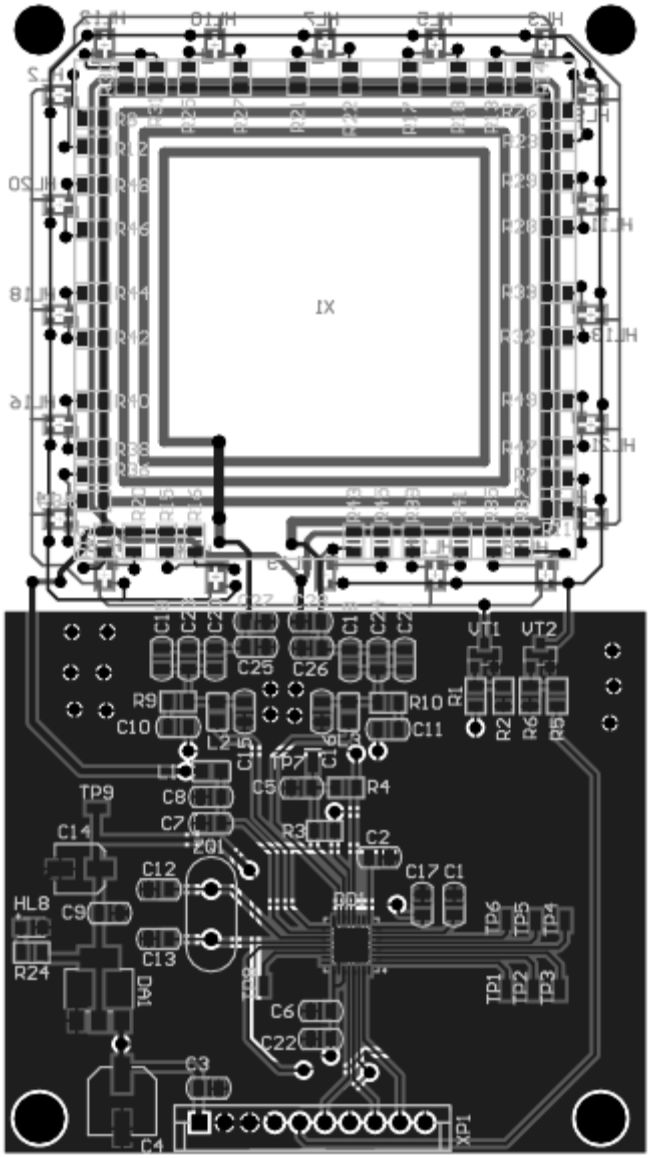

Рисунок 22 – Шари друкованої плати зчитувача карток NFC, суміщені разом.

На рис. 22 зображені суміщені разом шари друкованої плати, які вже підготовлені до відправлення виробнику друкованих плат. На малюнку можна розрізнити антену безконтактного зчитувача та місце для встановлення мікросхеми CLRC663. Зверніть також увагу що пасивні компоненти фільтру EMC та схеми узгодження розташовані на платі досить симетрично. Тому що будь яка асиметрія це різна довжина доріжок та різна паразитна ємність, а це означає змінення характеристик всього ланцюжка що на частотах 13.56MHz має сильний вплив, який призводить до зниження робочої відстані між зчитувачем та

карткою. На рис. 23 зображені обидві сторони вже готової друкованої плати. Програмні налаштування високочастотної частини мікросхеми CLRC663 та використані в цьому проекті протоколи будуть описані в розділі програмного забезпечення.

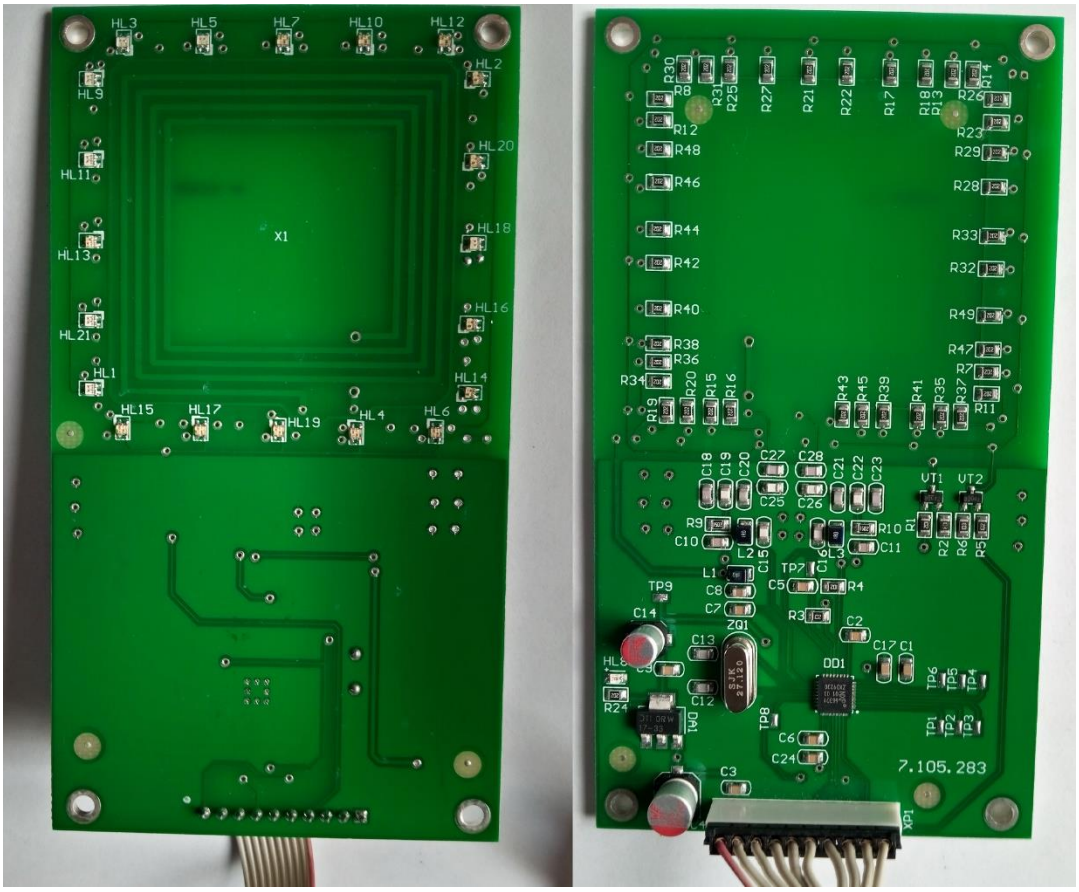

Рисунок 23 – Виготовлені друковані плати зчитувача карток NFC з обох сторін.

2.3.2 Процесорна плата

У цьому розділі буде описана плата, яка керує всією роботою стійки паркомату, відображає інформацію на екрані, керує периферійними пристроями, такі як принтер або сканер, та здійснює обмін даними з іншими стійками та сервером. Це плата процесору. Встановлений на платі процесор, точніше мікроконтролер класу **ARM Cortex M4** STM32F429, є центральною частиною всього пристрою. Принципова схема плати для швидкого ознайомлення

наведена на рис.24, а у високій якості в форматі PDF схема наведена в додатках. Ніяких особливостей схема не містить, всі компоненти включені згідно рекомендації їх виробників, тому лише треба перелічити всі наявні інтерфейси, які доступні програмісту: Ethernet, 2 штуки RS-232, UART, та спеціальний інтерфейс з кольоровою TFT панеллю з роздільною здатністю 800x480 пікселів. Документ на мікроконтролер STM32F429 наведений в [16], [17] документ на дисплейну панель – в [18].

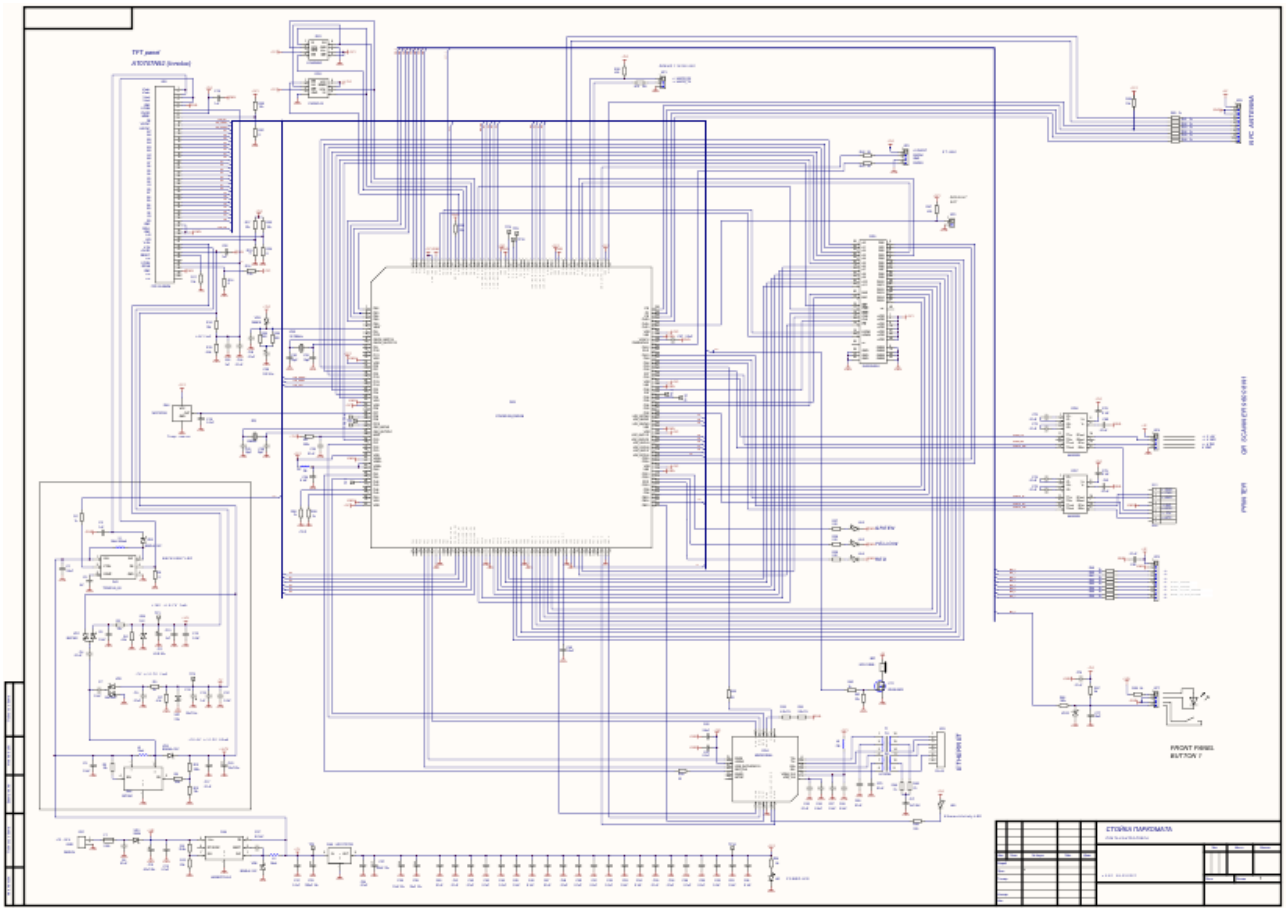

Рисунок 24 – Схема процесорної плати.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Вибір інструменту (IDE) для програми мікроконтролера

На сьогодні існує багато спеціалізованих інструментів для створення програмного забезпечення для мікроконтролерів. Загальна назва цих програмних комплексів - IDE (Integrated Development Tool), в перекладі
Інтегрований Інструмент Розробника. Вони можуть бути комерційними, тобто платними, так і некомерційними, тобто безплатними, або умовно безплатними. Як серед комерційних так і некомерційних продуктів можна знайти інструмент, який дозволить виконати поставлену задачу. Але перевага комерційних продуктів перед некомерційними полягає в системному підході до створення своєї продукції, наявності докладної інформації з прикладами та, часто, швидка відповідь від персоналу компанії. Нажаль, останній пункт не завжди справедливий навіть для комерційних IDE. Компанія-розробник системи автоматичного паркінгу має ліцензію на одну таку IDE, не найсучаснішу, але, крім самої IDE, яка дозволяє писати код для майже всього спектру ARM мікроконтролерів, та містить вбудовану мультизадачну систему реального часу (RTOS – Real Time Operation System). Крім того, за окрему плату до неї можна інтегрувати бібліотеки, які мають вигляд попередньо скомпільованих файлів в форматі DLL, які дають змогу працювати з основними протоколами, які сьогодні є найбільш популярні. Це підтримка файлових систем FAT-12, FAT-16 та FAT-32, робота з SD картками до 16GB, повна підтримка стеку протоколів TCP/IP включно з формуванням Веб-сторінок ресурсами самого мікроконтролеру. Також бібліотека містить драйвери для роботи з комунікаційними протоколами шин CAN та USB. Наявність на підприємстві офіційної версії та її повна придатність для виконання поставленої задачі і були причиною вибору системи RL-ARM від німецької компанії Keil. Зараз ця компанія належить британській компанії ARM, розробнику та власнику відомої архітектури

ARM© але новий власник зберіг стару назву програмного комплексу: RL-ARM Keil µVision.

3.2. Програма для керуючого мікроконтролера

Програма для мікроконтролера, використаного в стійці паркомату, загалом нараховує 68 файлів, включно з бібліотечними файлами та файлами драйверів як вбудованих в мікроконтролер, такі як модулі підтримки TFT дисплею та SDRAM, так і зовнішнього обладнання: мікросхем постійної пам’яті, мікросхеми Ethernet, та інші.

3.3 Загальна структура програми

Загальна задача роботи програми мікроконтролеру – це керування стійкою паркомату та зв’язок з сервером. Структура програми є безкінечним циклом, виконання окремих гілок якого залежить від інформації яка постійно надходить з сенсорів та запитів по Ethernet. загальна блок-схема зображена на рис.25.

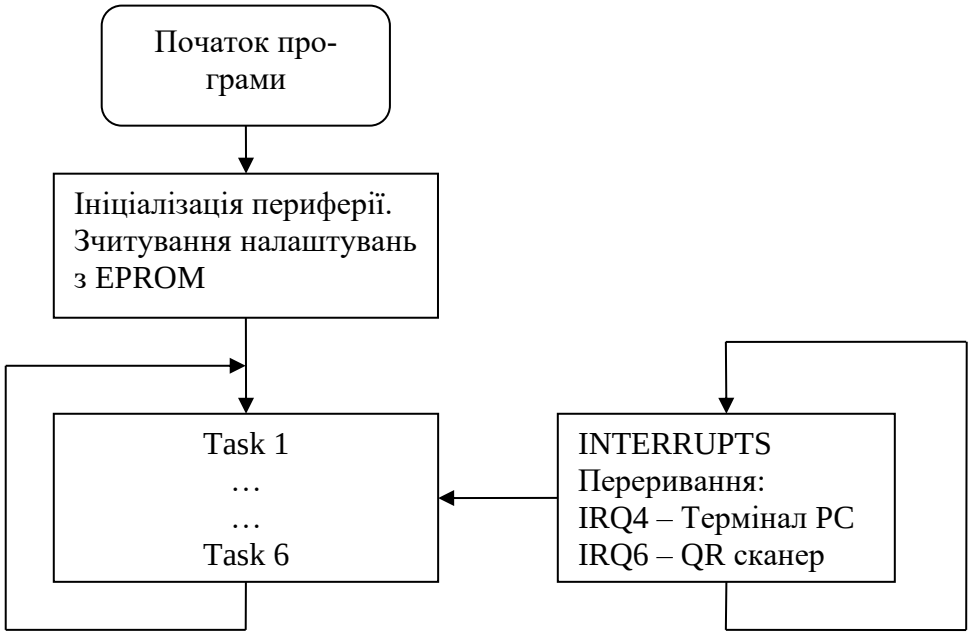

Рисунок 25 – Загальна блок-схема.

Якщо ліва частина блок-схеми потребує лише деталізації по її блоках, то відокремленість правої частини має бути роз'ясненою. Справа в тім що деякі процеси в мікроконтролерах виконуються незалежно від основного циклу програми. Така ж сама організація невідкладних системних процесів є й в абсолютно усіх інших комп'ютерних системах, але на відміну від, наприклад, персонального комп'ютеру, в мікроконтролерах ці процеси мають буди описані власне програмістом основного програмного коду, в той час як в персональних комп'ютерах за це відповідають програмісти операційної системи, а програміст прикладної програми не має не тільки повноважень, але навіть і технічної можливості на втручання в процеси операційної системи – за цим ретельно стежить спеціальний модуль процесору Supervisor, щоб задачі користувача (User Mode) не втручались в пам'ять операційної системи, яка має пріоритет, вищий за рівень користувача. Але програміст мікроконтролерів вимушений писати обробник задач переривання, щоб вчасно зреагувати на запити, яки надходять асинхронно та не можуть чекати обробки з головного циклу програми.

Нижче будуть наведені основні задачі, а потім до кожної з них буде надана докладніша деталізація. Програма для обох стійок: в'їзду та виїзду - одна, різні лише деякі налаштування, тому далі вони розділятися в тексті не будуть. Отже, стійка мусить виконувати:

- *Виводити всю необхідну інформацію на кольоровий дисплей TFT з роздільною здатністю 800x480.*
- *Ініціювати з'єднання з віддаленими стійками та відповідати на запити як віддалених стійок так і на запити серверу або POS-терміналу по інтерфейсу Ethernet, протокол UDP.*
- *Мати вбудований WEB-конфігуратор.*
- *Вести окремі бази даних: всіх дозволених номерів NFC карток, всіх заборонених номерів NFC карток, всіх роздрукованих на власній стійці талонів, всіх сканованих на власній стійці талонів.*

- *Керувати шлагбаумом, формуючі всі необхідні часові інтервали та відслідковуючи всі можливі ситуації через вбудовані в шлагбаум сенсори: неможливість закрити або відкрити шлагбаум, процес відкриття або закриття був перерваний, тощо.*
- *Відкриття шлагбауму при натисканні водієм на кнопку, встановлену на панелі та друк талону встановленої форми.*
- *Відкрити шлагбаум коли водій підніс картку NFC до зчитувача, якщо UID цієї картки був знайдений в базі даних серед дозволених номерів.*
- *Відкрити шлагбаум якщо водій підніс до сканеру талон, який містить спеціальну інформацію в форматі QR та контрольна сума CRC16 в кінці даних повністю співпала.*

Всі ці важливі задачі і будуть розглянуті в наступній, останній частині роботи.

3.4 Відображення інформації на TFT дисплеї

Мікроконтролер STM32F429 має деякі засоби для прискорення виводу інформації на кольорові дисплеї декількох типів. Для цього завдання було обрано кольорову панель з роздільною здатністю 800x480. Мікроконтролер має вбудований інтерфейс LTDC для дисплейних панелей цього типу, та модуль прискорення обробки та виводу під назвою Chrom-ART. Користуючись цими інструментами можна працювати з дисплеєм хоч і на досить низькому рівні, але вся рутинна робота буде виконуватись в автоматичному режимі без втручання програмного коду з головного циклу програми. Програмісту потрібно лише заносити необхідну інформацію в завчасно виділену пам’ять, а вбудовані в мікроконтролер апаратні сервіси будуть виконувати роботу по переміщенню даних з пам’яті в дисплей в фоновому режимі. Як приклад буде наведений принцип формування однієї літери на дисплеї.

3.5 Формування літери на TFT дисплеї

Для формування якогось зображення на дисплеї це зображення має вже знаходитись в пам’яті мікроконтролера, тобто завчасно бути там сформованим. Це стосується як літер так і малюнків. Взагалі, літери нічим не відрізняються від звичайного зображення, таким чином підхід до виводу і літер і зображення єдиний. За допомогою спеціальної програми-редактора були сформовані файли, що містять масив даних, з якого складається бітовий вигляд літери. Ось як процес створення або редагування виглядає для однієї літери ‘E’ (рис.26):

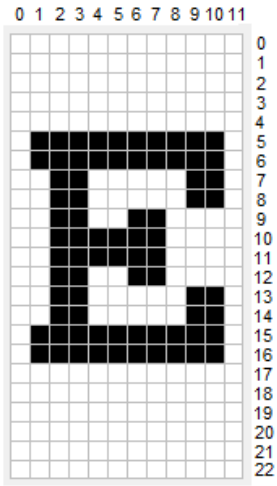

Рисунок 26 – Літера «E» під час створення.

Як можна бачити, літера побудована з 12 колонок, та з 23 рядків. Тобто в кожному рядку зберігається 12 пікселів, тобто кожен з рядків сформований з 12 біт. Найменшим дискретом даних для пам’яті мікроконтролера є байт, який складається з 8 біт. Таким чином на один рядок ми вимушені зарезервувати 16 біт (2 байти), з яких лише 12 біт несуть корисну інформацію. На всю літеру ми витрачаємо 2x23 = 46 байт. Програма-редактор формує вихідний файл у вигляді текстового хедерного файлу «.h» мови програмування C. В наступному вкладанні можна побачити як літера «E», що складається з масиву байт, представлена в хедерному файлі (рис.27):

Рисунок 27 – Літера «E» в вигляді масиву даних в хедерному файлі

Перший рядок з модифікаторами та типом **`static const uint8_t`** вказує на назву масиву, в яку внесено шрифт, його розмір та CP1251 код латинського символу «E» – 0x45. Розмір фіксований, 46 байт. Після масиву формується структура **`static const tImage`**, відповідальна за цей символ «E». В структурі вказані кількість активних біт в рядку – 12, кількість рядків – 23 та використана розмірність одиниці даних – 8 біт. Слід зауважити, що кодуючи пікселі одним бітом ми суттєво зменшуємо використану пам’ять, що вкрай важливо для мікроконтролерів, які всю інформацію зберігають у внутрішній пам’яті, яка, в цілому, і формує ціну на мікроконтролер. Цей метод не дозволяє робити градієнт кольорів літери. Кодуючи кожен піксель не бітом, а наприклад, байтом, ми змогли б задати колір для кожного окремого пікселя, але це призвело б до суттєвої витрати пам’яті – на кожну літеру пішло б в 8 разів більше пам’яті. На цю літеру «E», наприклад, замість 46 байт ми б витратили 368 байт. Взагалі, розмір всіх шрифтів з всіма розмірами та мовами може в такому разі досягнути мегабайта, що не припустимо для класу мікрокотнролерів.

Другим етапом після формування масиву зображень є декодування літери та розміщення масиву байт в відеобуфер – в спеціальну виділену ділянку пам’яті, з якої дані будуть автоматично перенесені в TFT панель контролером прямого доступу до пам’яті, DMA (Direct Memory Access). Під кожен піксель зображення виділено 2 байти (16 біт), що означає кольорову гаму в 2^{16} = 65536 кольорів, а загальна пам’ять відеобуферу складає 800*480*2 = 768000 байт. Таким чином кожен біт символу треба перетворити на 2 байти. Якщо біт в зображенні літери дорівнює одиниці, то він має бути відображений. Якщо біт дорівнює нулю він має бути фонового кольору, щоб його не було помітно. Біт, що відображається, може мати будь який колір зі згаданих 65536, звісно, окрім фонового. Для цього в функцію декодування додаються додаткові параметри: колір фону, колір символу, позиція на екрані в форматі X,Y, з якої почнеться відображення символу, назва та розмір шрифту, та, власне код символу, який треба вивести на екран. Загальна функція виводу тексту на екран TFT яка містить параметри, що передаються в функцію, має наступне визначення (рис.28):

```
147 // *********************************************************************
148 //      Format string printf()
149 //  Needs #include <stdarg.h>
150 //
151 //  PARAMETERS:
152 // 1) Position X
153 // 2) Position Y
154 // 3) Font number
155 // 4) Symbol color from table HTML 4
156 // 4) Background color from table HTML 4
157 // 5) Pointer to the string
158 //
159 //  RETURN:
160 // T_RETCHAR - struct consist last position X и Y
161 // *********************************************************************
162 T_RETCHAR printf_tft( unsigned long X,
163                       unsigned long Y,
164                       unsigned long Style,
165                       unsigned long ColorFont,
166                       unsigned long ColorBkgrnd,
167                       const char *format, ...)
168 {
169 #define TMPSTR_SIZE       80
170
171   unsigned long SizeStr;
172   va_list ap;
173   T_RETCHAR tRetChar={0};
174   char str[TMPSTR_SIZE];
175
176   va_start(ap, format);
177   SizeStr = vsnprintf(str, TMPSTR_SIZE, format, ap);
178   va_end (ap);
179   tRetChar = str_out_tft(X, Y, Style, ColorFont, ColorBkgrnd, SizeStr, str);
180   return tRetChar;
181 }
182
```

Рисунок 28 – Загальна функція виводу тексту на TFT екран.

Єдина функція, яка потребує роз'яснення, це функція **`str_out_tft()`**, опис інших функцій можна знайти в супроводжувальній документації. Власне, ця функція нічого суттєвого не робить, але вона викликає функцію **`print_char()`**, яка розпаковує бітові масиви в пам'ять, перетворюючі кожен біт в 16, призначає цьому пікселю відповідний колір, назва якого передається в параметрах функції та вже після цього виводить в визначене місце дисплею, координати якого передані в параметрах. Визначення функції можна побачити нижче (рис.29):

```
226
227 // ***************************************************************************
228 //            Output symbol to the TFT
229 //
230 //  PARAMETERS:
231 // 1) Symbol code CP1251 (U32)
232 // 2) Font Code (U32)
233 // 3) Start pos. X (U16)
234 // 4) Start pos. Y (U16)
235 // 5) Font color name from table (U32)
236 // 6) Background color name from table (U32)
237 //
238 // RETURN:
239 // T_RETCHAR - struct consist last position X and Y of the symbol
240 // ***************************************************************************
241 T_RETCHAR print_char( unsigned long cchar,
242                       unsigned long Style,
243                       unsigned short X,
244                       unsigned short Y,
245                       unsigned long F_color,
246                       unsigned long B_color)
247 {
```

Рисунок 29 – Визначення функції виведення окремого символу на TFT екран.

В самому початку функції (див. рядок 279 на рис. 30) шукаємо отриманий код символу серед існуючих та зберігаємо в змінних його розмір в бітах по горизонталі та число рядків вертикали. Якщо символ з таким кодом не знайдено, призначаємо невідомому коду символ «SPACE» (рядок 293) щоб не трапився вихід за масив, що є аварійним станом та обов'язково призведе до збою в роботі.

```
279   // Search a symbol code in array of symbols
280   for(i=0; i< ptFont->length; i++)
281   {
282     if(cchar == ptFont->chars[i].code)
283     {
284       pFont_image = (unsigned char*)ptFont->chars[i].image->pFont;
285       Font_Xsize = ptFont->chars[i].image->Xsize;
286       Font_Ysize = ptFont->chars[i].image->Ysize;
287       // stop search cycle with positive result if a symb. code is found
288       res = 1;
289       break;
290     }
291   }
292
293   // Set symbol 'SPACE' if code is not found
294   if(res == 0)
295   {
296     pFont_image = (unsigned char*)ptFont->chars[' '].image->pFont;
297     Font_Xsize = ptFont->chars[' '].image->Xsize;
298     Font_Ysize = ptFont->chars[' '].image->Ysize;
299   }
300
301   // store size of symbol in to return structure
302   tRet.Font_X = Font_Xsize;
303   tRet.Font_Y = Font_Ysize;
304
```

Рисунок 30 – Перевірка наявності символу та отримання його розмірів.

Для подальшої зручності у обчисленнях отриману кількість пікселів по горизонталі (рядок 285 рис.30) треба перетворити на ціле число байтів. Це виконує наступні 3 рядки коду (рядок 305, рис.31). Рядок 311 дуже важливий: в документації на мікроконтролер наголошується, що робота прискорюється якщо робити вибірку даних не з вбудованої FLASH пам’яті, а з оперативній пам’яті, в яку дані треба завчасно перенести з FLASH. Експериментально перевірено, що виведення кожного символу на дисплей прискорюється приблизно на 1mS, що значно позначається на сприйнятті та не дратує користувача повільністю появи інформації на екрані. Отже, в рядку 311 копіюємо в RAM символ та починаємо з ним працювати.

```
304
305   // Determine number of bytes in each line
306   bytes_in_line = Font_Xsize / 8;
307   tmp = Font_Xsize % 8;                                  // get reminder
308   if(tmp > 0)     bytes_in_line += 1;                    // if reminder is not 0, increment value
309
310   // Load font into temporary buffer in RAM. It faster about 1 mS than working from MCU FLASH
311   memcpy(&fontbuf[0], &pFont_image[0], bytes_in_line*Font_Ysize);
312
313   for(y=0; y < Font_Ysize; y++)
314   {
315     cnt = 0;                                             // counter scanned bits in line
316     // Check every bytes in every line of the font
317     for(b=0; b < bytes_in_line; b++)
318     {
319       u8tmp = fontbuf[y * bytes_in_line + b];
320       // Send every bit to the display from a line
321       // Check all bits in a byte
322       for(i=0; i < 8; i++)
323       {
324         if(BIT8(u8tmp, 7))
325           color = F_color;                               // set color of the font
326         else
327           color = B_color;                               // set color of the background
328         u8tmp <<= 1;
329         addrr = CurrentFrameBuffer + (X*2) + (Y*2*LCD_PIXEL_WIDTH) + (y*2*LCD_PIXEL_WIDTH) + ((i+(b*8))*2);
330         *(__IO uint16_t*)addrr = (unsigned short)color;
331         if(++cnt >= Font_Xsize) break;                   // Stop loop if all bits are checked and sent
332       }
333     }
334   }
335   return tRet;
336 }
```

Рисунок 31 – Перетворювання бітової послідовності на масив байтів та перенесення їх до відеобуферу.

Перетворення бітової послідовності символу в масив байтів виконується в циклі (рядок 313, рис. 31). Кожен біт в кожному рядку Y розгортається в чотирибайтову змінну (рядок 324), їй присвоюється переданий в параметрах колір (рядок 325 якщо треба показати піксель або рядок 327 якщо піксель треба сховати). В рядку 329 обчислюємо адресу зміщення від початку відеобуферу, куди буде скопійована змінна, яка містить колір пікселю:

`addr = CurrentFrameBuffer + (X*2) + (Y*2*800)+(y*2*480)+(i+(b*8)*2),`

`addr` - зміщення, яке шукаємо.

`CurrentFrameBuffer` – адреса початку відеобуферу (0xD0000000).

`X` - горизонтальна позиція, з якої починається вивід символу.

`Y` - вертикальна позиція, з якої починається вивід символу.

`y` - номер рядку символу, який обробляється в даний момент.

`b` - номер байту у поточному рядку.

`i` - номер біту у поточному байті.

`800, 480` - роздільна здатність TFT панелі по X та Y.

Треба мати на увазі, що пам’ять, зміст якої відображається на TFT панелі, має лінійну організацію, тобто безперервну послідовність байтів, в той час як зображення на дисплеї розділено на рядки та колонки. За співвідношенням даних в пам’яті та їх відображенням повинен стежити програміст, він повинен враховувати цю організаційну різницю в своїх програмах. Помноження на 2 в різних місцях формули означає що на кожен піксель призначається 2 байти в обраному режимі робот TFT панелі. Можливі також інші режими, наприклад, 24 біти (3 байти) на один піксель RGB, тобто на кожен колір припадає 8 біт, але такий режим значно уповільнює роботу з дисплеєм.

Отримавши зміщення лишається скопіювати змінну до відеобуферу з урахуванням отриманого зміщення, (див. рядок 330, рис. 31).

```
*(__IO uint16_t)addr = (unsigned short)color;
```

Змінна **addr** приводиться до беззнакового вказівника 16 біт з модифікатором **__IO**, що вказує компілятору не оптимізувати роботу з цією змінною, тому що вона призначена для портів вводу/виведення. Фактично, це аналог відомого модифікатору **volatile**.

Після збереження змінної в відеобуфері виконується перехід до наступного рядка символу або цикл переривається (331) якщо всі біти рядку цього символу вже виведені на TFT панель.

3.6 Інтерфейс ETHERNET в мікроконтролерах

Одним з ключових елементів бібліотеки RL-ARM є компонент RL-TCPnet [18]. Компонент RL-TCPnet написаний спеціально для малоресурсних ARM мікроконтролерів, має високу ступінь оптимізації та займає малий об’єм пам’яті. Набір протоколів TCP/IP розроблений як для підтримки локальних так і глобальних мереж. Всім відома семирівнева модель ISO 7-Layer Model. Модель TCP/IP розділяється на чотири рівні, які співвідносяться до семирівневої моделі ISO як це зображено на Рис 32. **Рівень мережевого доступу** складається з:

– Фізичного з’єднання з мережею.

– Формування пакетів прикладним рівнем програмного забезпечення для подальшого транспортування даних мережею.

– Керування пакетами даних, які транспортуються мережею.

В мікроконтролерних системах цей рівень співвідноситься з модулем MAC та мікросхемою PHY, та керується низькорівневими програмними драйверами. TCP/IP стек керує транспортним рівнем та рівнем мережевої маршрутизації.

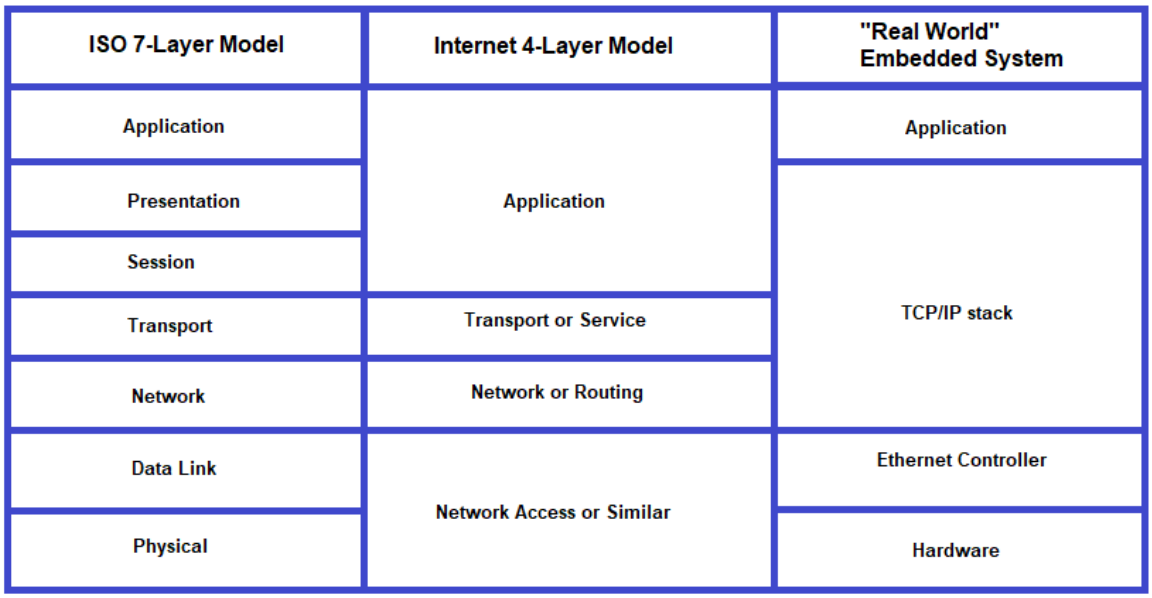

Рисунок 32 – Семирівнева модель ISO та мікроконтролерна реальність.

Мережевий рівень керує передачею пакетів даних між мережевими вузлами використовуючи Internet Protocol (IP). Транспортний рівень забезпечує з’єднання між прикладним рівнем та різними вузлами мережі. За це відповідають протоколи TCP (Transmission Control Protocol) та UDP (User Datagram Protocol). Прикладний рівень забезпечує доступ програмного забезпечення до комунікаційного обладнання. Прикладом цього є протоколи HTTP, SMTP та Telnet. Є можливість запровадити свій власний прикладний протокол та спілкуватись за його допомогою з різними вузлами використовуючи UDP та TCP пакети.

Для передачі даних користувача найчастіше використовуються три протоколи: IP (Internet Protocol), TCP (Transmission Control Protocol) та UDP (User Datagram Protocol). Типовим прикладом є ще ARP (Address Resolution Protocol) та ICMP (Internet Control Message Protocol). Задля зменшення об’єму пам’яті деякі мікроконтролерні реалізації TCP/IP містять лише невеликий набір протоколів з всього об’єму TCP/IP протоколів. Такі реалізації передбачають, що працювати мікроконтролерний пристрій буде з повноцінною реалізацією TCP/IP протоколів, наприклад, з персональним комп’ютером. Бібліотека RL-TCPnet підтримує повністю всі протоколи інтернет-вузлів.

3.7 Мережеві налаштування

Всі необхідні налаштування можна зробити з вкладники IDE як це показано на рис.33.:

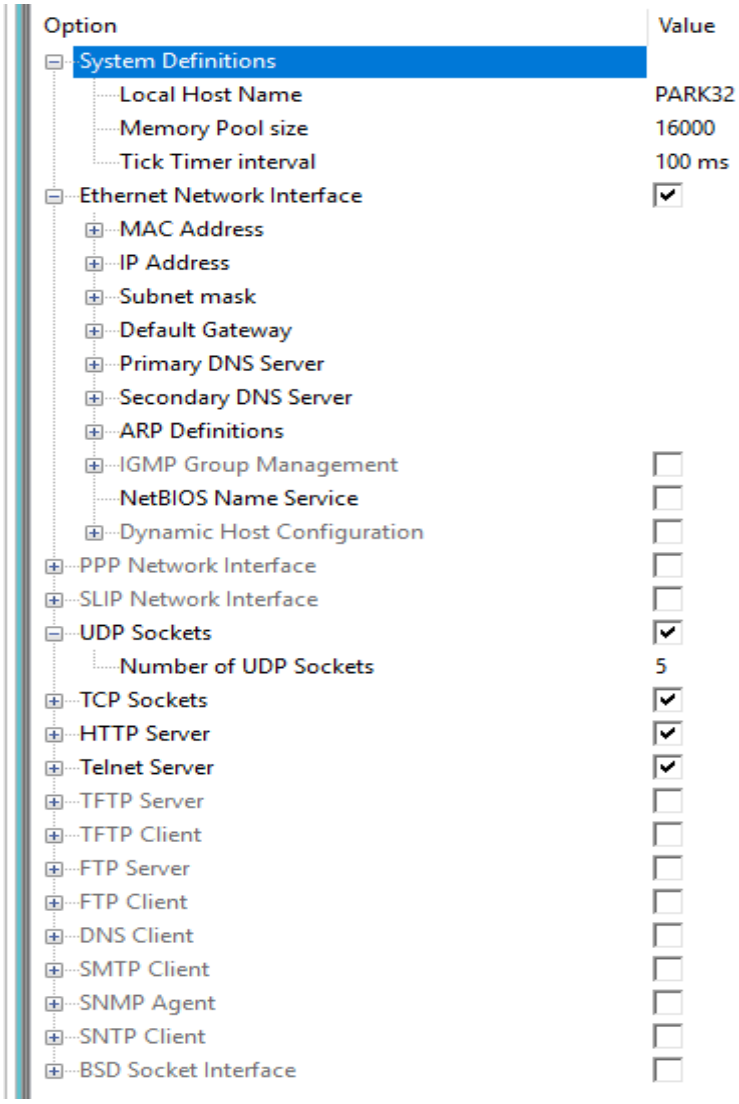

Рисунок 33 – Мережеві налаштування.

Тут можна задати MAC та IP адреси, ім’я хоста, розмір пам’яті для тимчасового збереження вхідних пакетів, максимальний розмір пакетів, які будуть оброблені, та інші налаштування.

3.8. Приклад роботи з протоколом UDP

В наступних двох пунктах буде наведено наглядні приклади роботи мікроконтролеру з мережею Ethernet, зокрема будуть показані механізми надходження та механізм відправки пакетів протоколу UDP.

3.9 Надходження пакету UDP

На рис.34 наведена функція по прийманню пакету UDP. Це типова call-back функція, викликається виключно операційною системою кожного разу, як мікроконтролер отримує адресований для нього пакет UDP. Це й є сутність так званих call-back функцій – їх викликає ОС при настанні деяких подій.

```
881
882  // *********************************************************************
883  //   This function is called when UDP data has been received
884  // *********************************************************************
885  U16 udp_callback (U8 socket, U8 *remip, U16 port, U8 *buf, U16 len)
886 {
887    memcpy(&tNetExt.remip[0], &remip[0], 4);
888    memset(&udp_rx_buf[0], 0, UDP_RX_SIZE);
889    memcpy(&udp_rx_buf[0], buf, len);
890    printf("Source IP: %d.%d.%d.%d  %d\r\n", remip[0], remip[1], remip[2], remip[3], port);
891    set_flag(UDP_NEW_PKT);
892    if((len == 2) && (bindex == (buf[0]<<8 | buf[1])))
893    {
894      wait_ack = __FALSE;
895    }
896    return (0);
897 }
898
```

Рисунок 34 – Call-back функція прийому даних по протоколу UDP.

Функція **`udp_callback()`** має 5 параметрів, через які операційна система передає важливу інформацію щодо відправника та реципієнта передана в параметрах функції:

1) сокет (комбінація IP адреси та порту, на який було надіслано пакет).
2) віддалена IP адреса, з якої надійшов пакет.
3) порт, з якого було надіслано пакет віддаленим IP.
4) вказівник на буфер, в якому зберігаються надіслані дані.
5) розмір отриманого пакету в цьому буфері.

В тілі Call-back функції треба якнайшвидше скопіювати надіслані дані щоб очистити буфер для наступних, які в цей момент, можливо, якраз надходять. Крім того Call-back функції мають дуже важливу неявну особливість, яка потребує уважного ставлення - з таких функцій не можна викликати інші функції, якщо вони можуть довго виконуватись, наприклад, відсилати якісь дані та потім чекати відповіді. Такі довгі за часом процеси будуть перериватись операційною системою та програма ніколи не дочекається відповіді, що, вочевидь,

порушує очікувану програмістом поведінку програми. Тому в рядок 891 (рис.34) введена глобальна змінна **UDP_NEW_PACKET**, яка активується якщо дійсно отримано новий пакет даних по UDP. Потім, в головному циклі потрібно в режимі полінгу постійно перевіряти зміст цієї змінної **UDP_NEW_PACKET**, та якщо він перестав бути рівним нулю, треба виконати функцію обробки прийнятого UDP пакету, після чого треба примусово обнулити змінну **UDP_NEW_PACKET**. Зрозуміло що перевіряти змінну **UDP_NEW_PACKET** треба частіше ніж пакети UDP надходять до мікроконтролера, інакше деякі з них будуть загублені.

3.10 Відправка пакету UDP

Щоб відправити дані за допомогою протоколу UDP достатньо викликати спеціальну бібліотечну функцію **udp_send()**. Для зручності вона обгорнута в свою власну функцію, яка виконує деяку попередню роботу, яку необхідно виконати перед надсиланням пакету UDP. Функція для надсилання даних наведена на Рис 35:

```
899
900 // ***************************************************************************
901 //           Send UDP packet
902 //
903 //  RETURN:
904 // FUNC_IS_ONGOING      - function is not finished
905 // FUNC_END_ERROR       - Error was occured while send UDP
906 // FUNC_END_OK          - Send UDP finished successfully
907 // ***************************************************************************
908 int udp_data_send(unsigned char *Data, unsigned short len)
909 {
910   unsigned char *pBuf, RemIP[4];
911
912   BOOL result=FALSE;
913
914   pBuf = udp_get_buf(len);
915   memcpy(pBuf, Data, len);
916   RemIP[0] = tNetExt.remip[0];
917   RemIP[1] = tNetExt.remip[1];
918   RemIP[2] = tNetExt.remip[2];
919   RemIP[3] = tNetExt.remip[3];
920   result = udp_send(udp_soc, &RemIP[0], tNetCfg.remUDPport, pBuf, len);
921   if(result == FALSE)
922   {
923     printf("Sending UDP packet was failed!\r\n");
924     return FUNC_END_ERROR;
925   }
926   // Reset timer wich sends PING requests to the server every 3-5 сек.
927   iPingTimeout = 0;
928   return FUNC_END_OK;
929 }
930
```

Рисунок 35 – Функція відправки даних по протоколу UDP.

Деякі рядки функції потребують роз'яснення. Рядок 914 (рис.35) викликає бібліотечну функцію **`udp_get_buf()`** – це ініціалізація спеціального буферу, лише з котрого можливо передати дані. У наступному рядку 915 копіюємо до цього спеціального буферу дані, що треба надіслати. Також зберігаємо в масиві IP адресу реципієнта даних (див. рядок 916-919, рис 35). Нарешті в рядку 920 викликається бібліотечна функція для надсилання даних та очікуємо результат її виконання, в залежності від якого в рядку 923 повертаємо негативний результат завершення функції та, якщо до плати підключений термінал, формуємо попередження про невдачу пересилання даних: "**`Sending UDP packet was failed`**". Це попередження з'явиться на екрані терміналу, в якості якого зазвичай використовують персональний комп'ютер з будь якою термінальною програмою, що суттєво допомагає налагоджувати програму.

3.11 WEB-конфігуратор

Користувач парковочного комплексу має змогу налаштувати деякі параметри стійки паркомату через веб-сторінку, яка зберігається в пам'яті мікроконтролера. Підключення до стійки здійснюється будь-яким веб-браузером через Ethernet з'єднання за фіксованою адресою 192.168.1.11. Зразу ж з'явиться головна сторінка, зображена на рис.36.

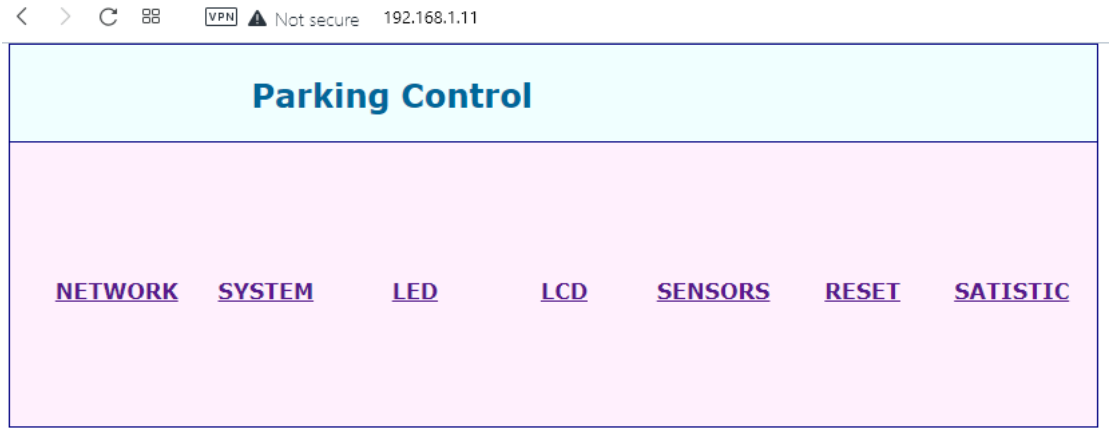

Рисунок 36 – Головна сторінка веб-конфігуратору.

Налаштування мережевих параметрів здійснюється в пункті **NETWORK** (див. рис.37):

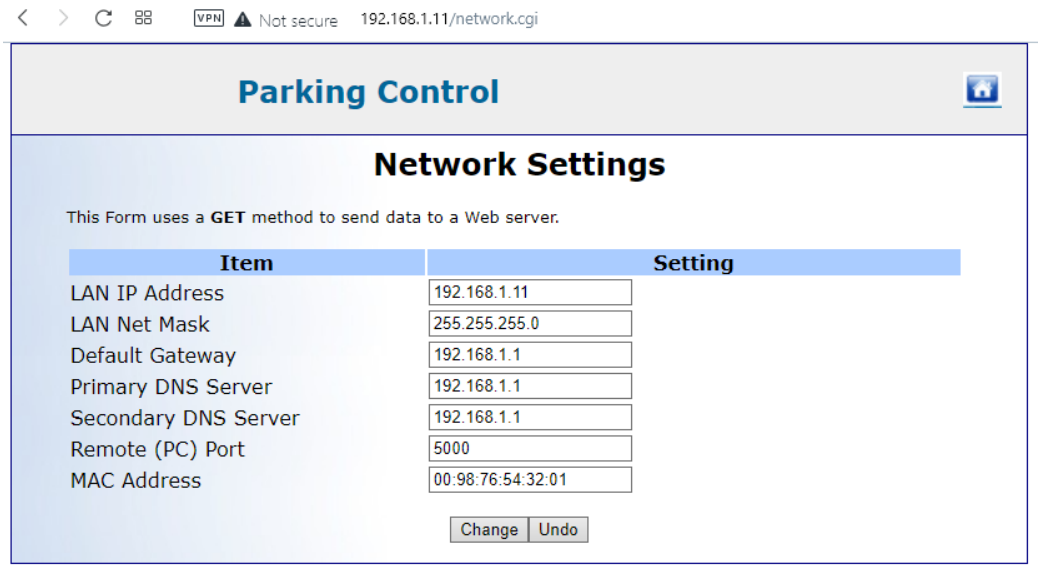

Рисунок 37 – Сторінка мережевих налаштувань.

Роз'яснень ця сторінка не потребує. Наступна сторінка має назву «**SYSTEM**», та дає змогу встановити або змінити паролі доступу до веб-конфігуратору. Особливої уваги вона не заслуговує, її зовнішній вигляд наведений на рис. 38.

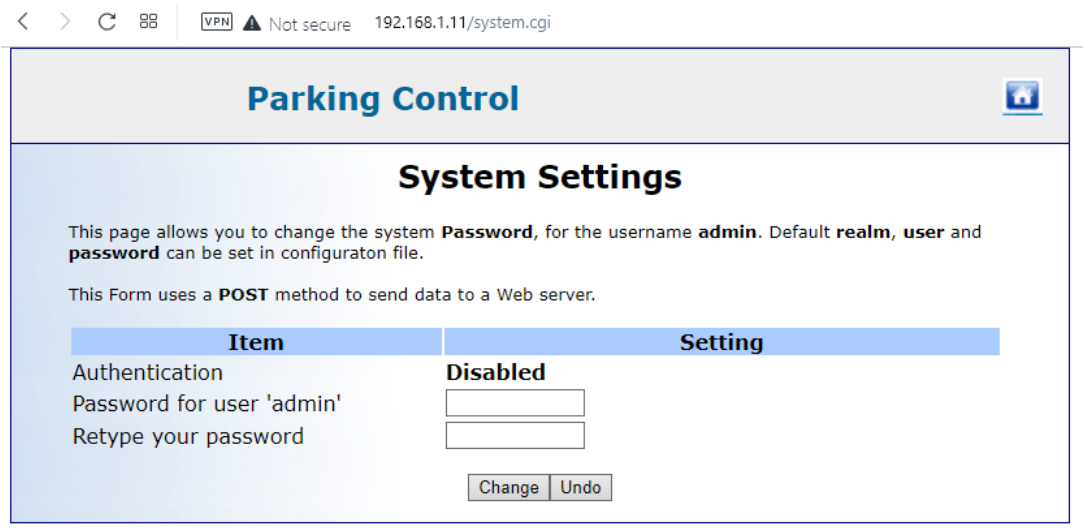

Рисунок 38 – Сторінка паролю.

Найбільш цікава сторінка це, власне, сторінка налаштування інженерних параметрів, побачити її можна на рис.39:

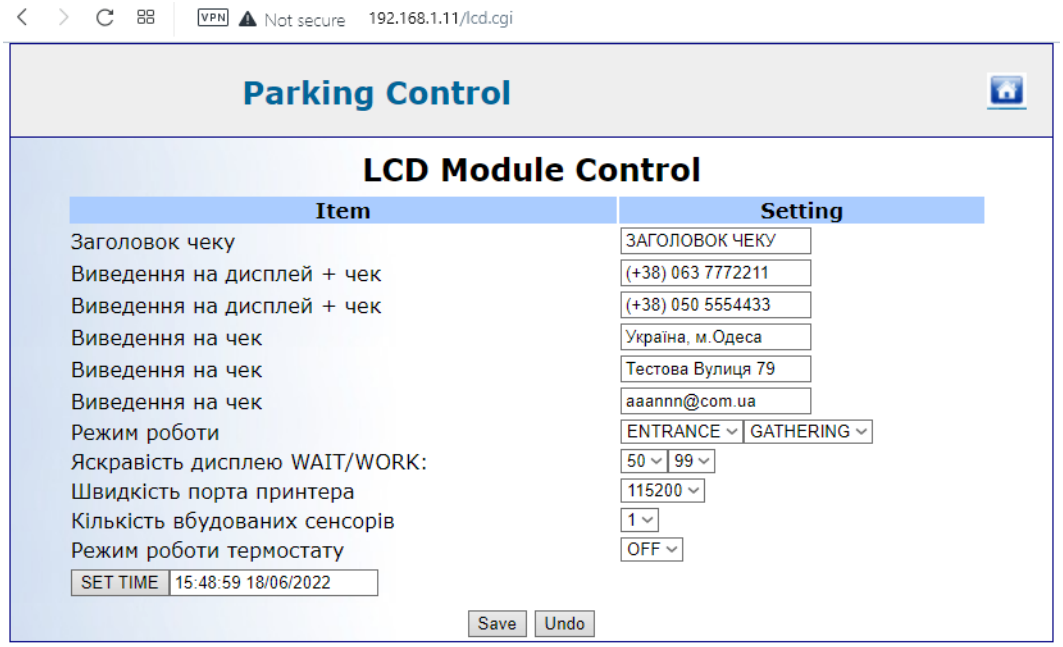

Рисунок 39. Сторінка інженерних налаштувань.

Шість перших рядків призначені для виведення тексту на паперовий чек та на дисплей. Наприклад, в рядку під назвою «**Заголовок чеку**» треба ввести текст, який буде зображено на першому рядку паперового чеку, який роздрукує принтер під час в'їзду на стоянку транспортного засобу. Тому ці пункти не заслуговують окремої уваги. Більш важливими є наступні пункти. Режими роботи стійки паркомату визначаються саме тут: якщо вибрати «**ENTRANCE**» то стійка буде працювати як в'їзна, а якщо вибрати «**EXIT**» то стійка стане виїзною стійкою. Рядок під назвою «**Яскравість дисплею / WAIT WORK**» визначає з якою яскравістю буде працювати дисплей в режимі очікування WAIT, це коли сенсори стійки не бачать транспортного засобу перед стійкою, то не треба занадто яскравого відображення інформації та в режимі WORK, коли сенсори визначили наявність засобу перед стійкою, тож яскравість має бути підвищеною. Наступний рядок визначає швидкість комунікаційного порту принтера. Рядок з назвою «**Кількість вбудованих сенсорів**» якраз і визначає кількість тих самих сенсорів, яки були описані раніше. «**Режим роботи термостату**» визначає на якій температурі вмикати та вимикати термостат обігріву. Цей

термостат встановлений в стійку щоб у зимовий період нагрівати прилади, чутливі до низьких температур. Останній рядок призначений синхронізувати час вбудованого в мікроконтролер годинника реального часу (RTC) з більш точним годинником комп'ютера.

Заслуговує невеликої уваги ще одна сторінка веб-конфігуратору – **Statistics**, рис.40:

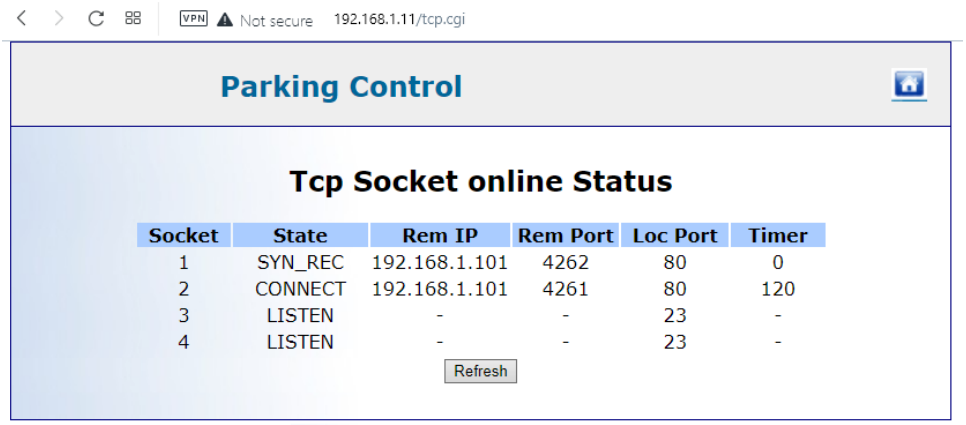

Рисунок 40 – Сторінка статистики з'єднань.

На цій сторінці можна побачити всі поточні з'єднання, які має стійка. Це дає можливість проконтролювати всі підключення та ресурси, до яких ці підключення намагаються дістатись. Це корисно як під час налагодження так і під час роботи, коли можуть мати місце хакерські атаки.

3.12 Програмна реалізація Веб-інтерфейсу

Одна з ключових відмінностей підтримки TCP/IP бібліотекою RL-TCPnet є підтримка веб серверу HTTP. З веб сервером можна працювати за допомогою браузерів на будь який платформі: PC, Mac, смартфони та будь які інші пристрої, які підтримують Інтернет. HTTP сервер підтримує CGI (Common Gateway Intreface) який дозволяє вводити та виводити дані в мікроконтролерні системи на мові «C».

Контент Веб-серверу може бути будь якого типу, що може бути відображений програмою браузером. Це може бути текст в форматі HTML, який

містить зображення в форматах PNG, GIF та GPEG, звукові файли в WAV або MP3 форматах, або активний контент як JavaScript. Обмеження тільки в доступному об’ємі пам’яті мікроконтролеру для зберігання контенту.

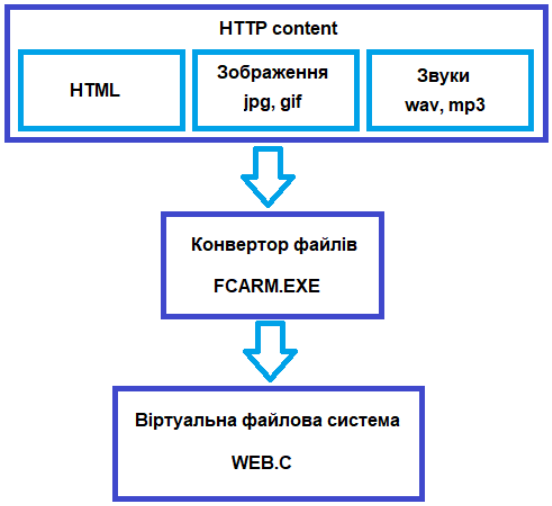

Рисунок 41 – Послідовність перетворення HTTP контенту у виконуваний файл.

Для того щоб інтегрувати HTML сторінки в мікроконтролерну програму необхідно додати всі файли (HTML файли, GIF та інші) до загального проекту. Всі ці файли мусять бути додані в текстовому форматі. Ці файли будуть оброблятися віртуальною операційною системою в тій послідовності, з якою будуть запити цих файлів від браузеру. Перетворення усіх файлів Веб серверу в текстовий файл масивів «С» здійснюється спеціальною утилітою FCARM.EXE, яка постачається разом з IDE. Назва файлу **web.inp**. Вже під час виконання програми дані з масивів цього файлу перетворюються віртуальною файловою системою WEB.C в HTML контент та відсилаються до браузеру по його запиту.

Ось, наприклад, як виглядає файл **Network.cgi** (див. рис.42), виконання котрого це сторінка мережевих налаштувань, яка була зображена вище на рис. 37. Це звичайний JavaScript файл в форматі HTML, але для його роботи в мікроконтролерній системі на самому початку кожного рядку коду були додані спеціальні ключові символи, які вказують віртуальній файловій системі

WEB.C як поводитись з тим чи іншим рядком коду JavaScript. Детальний опис ключових символів:

I – Вставка HTML файлу та пересилання його до браузеру.

T – Текстова послідовність яка без змін мусить бути переслана до браузеру.

C – Командний текст, для обробки якого потрібно визвати CGI інтерпретатор.

. – Крапка (.) мусить бути наприкінці кожного CGI файлу.

– Символ хєшу (#), після нього весь текст розглядається як коментар.

```
1  t <html><head><title>Network Settings</title>
2  t <script language=JavaScript>
3  t function changeConfirm(f){
4  t  if(!confirm('Are you sure you want to change\nthe Network settings?')) return;
5  t  f.submit();
6  t }
7  t </script></head>
8  i pg_header.inc
9  t <h2 align=center><br>Network Settings</h2>
10 t  <p><font size="2">This Form uses a <b>GET</b> method to send data to a Web server.</font></p>
11 t <form action=network.cgi method=get name=cgi>
12 t <input type=hidden value="net" name=pg>
13 t <table border=0 width=99%><font size="3">
14 t <tr bgcolor=#aaccff>
15 t  <th width=40%>Item</th>
16 t  <th width=60%>Setting</th></tr>
17 # Here begin data setting which is formatted in HTTP_CGI.C module
18 t <tr><td>LAN IP Address</td>
19 c a i <td><input type=text name=ip value="%d.%d.%d.%d" size=18 maxlength=18></td></tr>
20 t <tr><td>LAN Net Mask</td>
21 c a m <td><input type=text name=msk value="%d.%d.%d.%d" size=18 maxlength=18></td></tr>
22 t <tr><td>Default Gateway</td>
23 c a g <td><input type=text name=gw value="%d.%d.%d.%d" size=18 maxlength=18></td></tr>
24 t <tr><td>Primary DNS Server</td>
25 c a p <td><input type=text name=pdns value="%d.%d.%d.%d" size=18 maxlength=18></td></tr>
26 t <tr><td>Secondary DNS Server</td>
27 c a s <td><input type=text name=sdns value="%d.%d.%d.%d" size=18 maxlength=18></td></tr>
28 t <tr><td>Remote (PC) Port</td>
29 c a t <td><input type=text name=rport value="%d" size=18 maxlength=18></td></tr>
30 t <tr><td>MAC Address</td>
31 c a r <td><input type=text name=mac value="%02X:%02X:%02X:%02X:%02X:%02X" size=18 maxlength=24></td></tr>
32 t </font></table>
33 # Here begin button definitions
34 t <p align=center>
35 t <input type=button name=set value="Change" onclick="changeConfirm(this.form)">
36 t <input type=reset value="Undo">
37 t </p></form>
38 #i pg_footer.inc
39 . End of script must be closed with period.
40
```

Рисунок 42 – Файл JavaScript сторінки мережевих налаштувань.

Тепер про те, яким чином передаються в головний цикл програми дані, введені користувачем з сторінки свого браузеру. Як можна побачити на рядку 25 (рис. 42), змінна value приймає чотири числа: **value = ”%d.%d.%d.%d”** та має модифікатор імені **name = ip**. Це означає нову IP адресу, на яку користувач бажає в подальшому переключити стійку паркомату. Ддля обробки введених

через браузер даних до загального проекту має бути підключений файл «C» з назвою **HTTP_CGI.C**, який містить спеціальну функцію **cgi_process_var()**. Саме ця функція сканує змінні, які модифікуються в браузері та передає їх зміст в глобальні змінні всього проекту. Ось як виглядає початок цієї функції на рис.43, зокрема там відображено і обробку адреси IP та маски Subnet:

```
112  /*--------------------------- cgi_process_var -------------------------------*/
113
114 void cgi_process_var (U8 *qs) {
115    /* This function is called by HTTP server to process the Querry_String   */
116    /* for the CGI Form GET method. It is called on SUBMIT from the browser. */
117    /*.The Querry_String.is SPACE terminated.                                 */
118    U8 *var;
119    int s[8], port=0;
120
121    var = (U8 *)alloc_mem (40);
122    do {
123      /* Loop through all the parameters. */
124      qs = http_get_var (qs, var, 40);
125      /* Check the returned string, 'qs' now points to the next. */
126      if (var[0] != 0) {
127        /* Returned string is non 0-length. */
128        if (str_scomp (var, (U8*)"ip=") == __TRUE) {
129          /* My IP address parameter. */
130          sscanf ((const char *)&var[3], "%d.%d.%d.%d",&s[0],&s[1],&s[2],&s[3]);
131          tNetCfg.locip[0] = s[0];
132          tNetCfg.locip[1] = s[1];
133          tNetCfg.locip[2] = s[2];
134          tNetCfg.locip[3] = s[3];
135        }
136        else if (str_scomp (var, (U8*)"msk=") == __TRUE) {
137          /* Net mask parameter. */
138          sscanf ((const char *)&var[4], "%d.%d.%d.%d",&s[0],&s[1],&s[2],&s[3]);
139          tNetCfg.NetMask[0] = s[0];
140          tNetCfg.NetMask[1] = s[1];
141          tNetCfg.NetMask[2] = s[2];
142          tNetCfg.NetMask[3] = s[3];
143        }
144        else if (str_scomp (var, (U8*)"gw=") == __TRUE) {
```

Рисунок 43 – Функція обробки даних, введених користувачем в браузері: cgi_process_var().

Поза цієї бакалаврської роботи залишилось багато тем, які не були розглянуті за браком об’єму роботи: робота з чековим принтером та формування QR кодів, сканування QR кодів, управління виконуючим механізмом – шлагбаумом за допомогою кінцевих автоматів. Не було розглянуто також робота з базами даних в мікросхемах пам’яті DATAFLASH, особливості котрих потребує дуже великої уваги та кропіткої роботи. Робота над стійкою паркомату ще триває: не була завершена частина взаємодії з серверним ПЗ, не випробувана в комплексі з багатьма гейтами та багатьма POS-терміналами.

ВИСНОВКИ

В представленій роботі була описана окрема деталь парковочного комплексу - стійка паркомату. Розглянута її загальна архітектура, використані складові як сторонніх виробників так і власного виробництва. Були описані програмні інструменти для створення програмного забезпечення та критерії їхнього вибору. Були проведені математичні обчислення антени, а потім, на базі розрахунків, проведено багато експериментальних досліджень для отримання найкращого результату. Були виготовлені два екземпляри таких стійок та їхнє тестування на протязі 10 місяців, з серпня 2021 року до травня 2022 року підтвердило що обрана концепція ”**offline**” хоч і вважається застарілою, але вона більш надійна та стійка до негативних факторів впливу.

Як відзнаку успішності цього проекту можна розглядати факт, що Одеська міська мерія надала право маркувати ці вироби знаком «Вироблено на Одещині» (рис. 44).

Рисунок 44 – Отримане від Одеської мерії право маркування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mean Well. 200W Single Output Switching Power Supply. NES-200-Series.pdf
2. User Manual VKP80II-SX. Custom S.p.A. Via Berretine 2/B. 4310 Fontevivo (Parma) - Italy.pdf
3. Commands Manual VKP80II-SX_1.40. . Custom S.p.A. Via Berretine 2/B. 4310 Fontevivo (Parma) – Italy.pdf
4. EM20-80 OEM scan engine. User guide. Fujian Newland Auto-ID Tech. Co., Ltd. 2019.pdf
5. Sensors. Loop Detectors. Carlo Gavazzi Automation S.p.A. Via Milano, 13-I-20020, Lainate (MI) Italy.pdf
6. MF 1 IC S50. Functional specification. Rev.5.2. Product data sheet. 001052.pdf
7. MF 1S 70YYX. Mifare Classic EV1 4K. Rev.3.2. Product data sheet. 279332.pdf
8. MF 0 IC U1. Mifare Ultralight contactless single trip ticket IC. Rev.3.7. Product data sheet. 028637.pdf
9. MF 0 IC U2. Mifare Ultralight C. Rev.3.2. Product data sheet. 171432.pdf
10. ISO/IEC FCD 14443-3:1999(E). Identification cards. Contactless integrated circuits cards. Proximity cards. Part 3: Initialization and anticollision.pdf
11. Application Note. Design of MF RC500 Matching Circuits and Antennas, Philips Semiconductors, 2000.
12. AN11019. CLRC663, MFRC630, MFRC631, SLRC610 Antenna Design Guide, rev. 1.4. Application Note. 205814.
13. 13.56MHz RFID System Design Guide, Microchip Technology Inc., U.S.A., 2004
14. CLRC663. High performance multi-protocol NFC frontend CLRC663 and CLRC663plus.Product data sheet. Rev 4.6. 171146
15. Micore reader IC family. Directly Matched Antenna Design.rev.2.05. Application Note.pdf

16. STM32F427xx STM32F429xx Datasheet. ID024030. rev10.pdf
17. RM0090 Reference manual. rev19.pdf
18. AT070TN94. LCD module specification A070-94-TT-01. v01. InnoLux Corp.pdf
19. Getting Started Building applications with RL-ARM. ARM Ltd and ARM Germany GmbH.pdf