Міністерство освіти і науки України
ДВНЗ «Прикарпатський національний університет імені Василя Стефаника»
Кафедра комп'ютерної інженерії та електроніки
(повна назва кафедри)

Сеньків Назарій Богданович
Senkiv Nazarii

УДК 004:681.5

Спеціальність 123 «комп'ютерна інженерія»
(шифр та назва спеціальності)

Кваліфікаційна робота
на здобуття освітньо-кваліфікаційного рівня бакалавр
(бакалавр, спеціаліст, магістр)

Система безконтактного контролю відвідуваності студентів за допомогою RFID/NFC міток
Contactless student attendance system by using RFID / NFC tags

Науковий керівник:
кандидат технічних наук,
доцент Голота В.І.

Рецензент:
Доктор фіз.-мат. наук, професор
кафедри матеріалознавства і
новітніх технологій
Яремій І.П.

Івано-Франківськ
2020

Форм.	Зона	Поз.	Позначення	Найменування	К-ть	Прим.
			123. УДК 004:681.5	Блок-схема алгоритму	1	
			123. УДК 004:681.5	Пояснювальна записка	60	

Змн	Арк.	№ докум.	Підп.	Дата	123.КІ-41.24		
Розроб.		Сеньків Н.Б.			Специфікація	Літ.	Арк. 2 / Аркушів 60
Перев.		Голота В. І.					
Н.конт.							
Затверд.							

АНОТАЦІЯ

В бакалаврській роботі розроблено безконтактну систему контролю відвідуваності.

Система базується на мікроконтролері ESP8266. Для ідентифікації студента використовується безконтактна RFID/NFC технологія.

Розроблено блок-схему алгоритму роботи та електричні схеми з’єднання компонентів. Зроблено обґрунтований вибір апаратних та програмних засобів.

Обґрунтована економічна доцільність виготовлення даної системи.

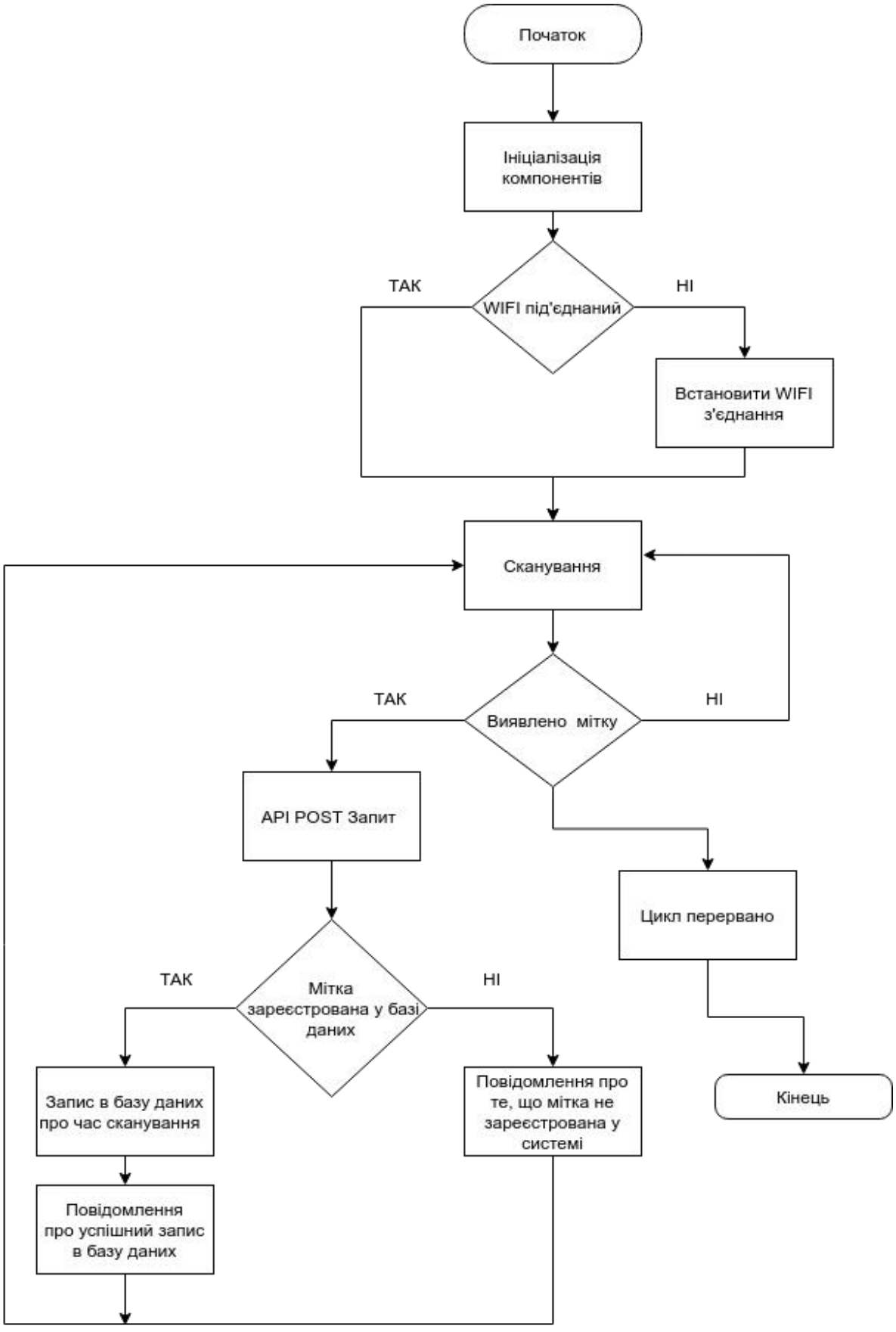
SUMMARY

The diploma project presents a contactless student attendance system.

The system is based on ESP8266 microcontroller. Contactless RFID/NFC technology is using for identification of students.

An block diagram of the algorithm and an electrical wiring diagram of the components have been developed. The best hardware and software components were selected.

The economic feasibility of manufacturing the system was substantiated.



					123.КІ-41.24		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Сеньків Н.Б.			Блок схема роботи системи	Арк.	Аркуш 5 / Аркушів 60
Перевірив		Голота В. І.					
Н. Контр.							
Затверд.							

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Фізико-технічний факультет
Кафедра «Комп'ютерної інженерії та електроніки»

Пояснювальна записка

до кваліфікаційної роботи на тему:

Система безконтактного контролю відвідуваності студентів за допомогою RFID/NFC міток

					123.КІ-41.24		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Сеньків Н.Б.			Пояснювальна записка	Арк.	Аркуш 6 / Аркушів 60
Перевірив		Голота В. І.					
Н. Контр.							
Затверд.							

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СКВ — система контролю відвідуваності.

СКД — система контролю доступу.

БД — база даних.

NFC — технологія безконтактного зв’язку, яка працює на відстані до 10 см.

RFID — технологія радіочастотної ідентифікації.

API — опис способів взаємодії двох або більше комп’ютерних систем.

ID — унікальний код.

СУБД — система управління базами даних.

ВСТУП

У сучасному світі завдяки стрімкому розвитку технологій з'явилась можливість оптимізувати багато буденних процесів. Як наслідок з'являються різні системи контролю відвідуваності. Їх використовують підприємства, навчальні заклади, державні установи, офіси. Завдяки системам контролю відвідуваності стало можливим вести зручний облік відвідування працівниками, учнями чи студентами тієї чи іншої установи.

Така система складається з ідентифікатора, спеціального зчитувального пристрою, контролера, а також програмного забезпечення. Ідентифікатор — це ключовий елемент, який дозволяє людині взаємодіяти із системою. Функцію ідентифікатора можуть виконувати: спеціальний брелок, безконтактна картка, картка із магнітною стрічкою, унікальні людські особисті ознаки(відбитки пальця, малюнки сітківки ока), цифровий код, який потрібно безпосередньо ввести із клавіатури. Інформація з ідентифікатора зчитується спеціальним пристроєм та передається на контролер. Контролер обробляє отримані дані та зберігає інформацію для подальшого використання. Також розробляють спеціальне програмне забезпечення, яке дозволяє налаштовувати систему та відображати зчитані дані.

Проблема контролю відвідування студентами університету є достатньо поширеною. Переважно в університетах використовують паперові журнали, в яких староста вручну відмічає присутність студентів, а також викладачі ведуть свої персональні журнали. Такий спосіб не дуже зручний, потрібно витрачати зайвий час, проблематично сформувати звіт по відвідуваності або переглянути статистику як той чи інший студент ставиться до навчального процесу. Вартість збереження інформації на електронних носіях, а не на папері значно менша.

Розроблений та описаний у цій дипломній роботі пристрій дозволяє безпосередньо автоматизувати процес контролю відвідування. На ринку існує багато готових систем відвідування. Але зазвичай вони досить дорогі.

Розроблена система базується на мікроконтролері ESP8266, для ідентифікації використовуються RFID або NFC мітки. Для написання програмної частини використано мову програмування Python та її фреймворки Micropython і Django. Усе це дозволило розробити дешеву і просту в реалізації систему, яка дозволяє зручно і легко вести статистику відвідування занять. Будь-який навчальний заклад може дозволити собі використовувати такий пристрій, що позитивно вплине на організацію навчального процесу.

1. ОПИС ОСНОВНИХ ЕЛЕМЕНТІВ ТА ПРИНЦИПИ РОБОТИ СИСТЕМ КОНТРОЛЮ ВІДВІДУВАННЯ

1.1. Загальні поняття СКВ

Контролювати та вести облік відвідування людьми певних закладів чи подій почали багато років тому. На кожному етапі розвитку людства такий облік був потрібен у воєнній сфері, медицині, політиці, освітній сфері. Завжди увесь облік вівся на папері. Часто потрібні були окремі люди, які займалися б виключно цією роботою. Усі ці дані досить проблематично відсортувати чи отримати певну статистику. Також присутній ризик втрати інформації, оскільки паперові носії легко піддаються фізичному впливу.

Завдяки розвитку технологій почали з'являтись цифрові системи контролю відвідуваності. Їхні переваги очевидні. Стало можливим безпосередньо обробляти велику кількість даних, систематизувати й структурувати їх, значно зменшились ризики їхньої втрати, відпала потреба витрачати зайвий час і ресурси, за мінімальний час можна ідентифікувати велику кількість людей.

Система контролю відвідуваності – це сукупність інструментів та програмного забезпечення, що контролює вхід, вихід та переміщення людей у певній будівлі чи на якійсь території. СКВ також обробляє інформацію і веде статистику. Найчастіше використовують системи, розроблені на основі RFID технологій.

Особа, якій потрібно відмітитись у системі використовує ідентифікатор. Щоб отримати інформацію із мітки потрібен зчитувальний прилад (зчитувач). Зчитані дані обробляє контролер і записує їх у локальну базу даних або за допомогою інтернет з'єднання передає їх на сервер. Уся інформація зберігається в базах даних. Завдяки цьому її можна швидко обробляти та переглядати, а також якщо використовується резервне копіювання, або зберігання на декількох серверах виключається можливість втрати даних,

оскільки це могло би бути якби використовувались паперові носії. Опрацьовані відомості виводяться у настільній програмі або веб додатку, для перегляду статистики, оформлення звіту чи виконання інших операцій.

1.2. Історія появи RFID

У 1935 році шотландський фізик Роберт Олександр Ватсон-Ватт винайшов радіолокаційний детектор, який пізніше широко використовувався Німеччиною та Альянсом, включаючи Америку та Британію. Обидві сторони використовували радіолокаційну технологію для того, щоб виявити літальний апарат у той період.

Однак радіолокаційна технологія на той час не могла точно визначити розміщення літака. Це працювало виключно на певній дистанції. Зрештою, німці виявили та розв’язали проблему, шляхом обертання літака, при поверненні на базу. Це спричинило зміну відбитого назад радіосигналу, що створило специфічний сигнал. Це називається пасивною системою RFID, оскільки вона здатна ідентифікувати об'єкти завдяки відбиванню сигналу.

Всупереч тому, що розв’язання проблеми було геніальним, воно вимагало багато зусиль та часу для пілотів. Тому Ватсон-Ватт, британський піонер, який винайшов радіолокаційну технологію, розробив систему під назвою "Ідентифікація друга або ворога", яка могла б ідентифікувати, чи літак від ворога чи від союзників.

Ця нова система полягає у встановленні передавача в британські літаки. Принцип полягає в тому, що літак міг виявити однаковий радіолокаційний сигнал із землі, і тоді він відправляв сигнал назад на базу щоб підтвердити, що він не ворог. Це представлення роботи RFID, яка передбачає зв'язок і передачу між одержувачем і відправником.

Починаючи з 1966 року в США, Європі та Японії RFID системи широко використовуються для запобігання крадіжкам у магазинах.

На початку 1973 р. Маріо В. Кардулла отримав патент на винахід активного тегу RFID, який може бути перезаписаний.

У тому ж році Чарльз Уолтон отримав патент на винахід пасивного RFID тега. Ця система використовується в сучасних домофонах. Для ілюстрації він вставив тег RFID у карту і встановив зчитувач на двері. Двері автоматично відчиняться, якщо зчитувач зможе визначити та перевірити картку.

Система RFID також була розроблена і широко поширена в державних секторах. Вона використовувалась для відстеження ядерних матеріалів, які надходили на підприємства в Лос-Аламосі. Радіолокаційний передавач прикріплений до вантажівки, а зчитувач — до воріт. Якщо сигнал і ідентифікаційний номер від передавача збігаються з сигналом зчитувача, затвор відкриється. Пізніше ця система буде використана на платних місцях пропуску [6, с.8-9].

У 1987 році впроваджена перша платна дорога із використанням RFID технології. Ця система широко використовується і у наші дні.

До 2000 року було подано більше тисячі патентів, що базувались на RFID.

У 2015 році ринок RFID технологій оцінено у 26 мільярдів американських доларів. Ще у 2005 він оцінювався в майже 2 мільярди. За 10 років ринок виріс на 24 мільярди. Експерти прогнозують популярність цієї технології ще мінімум на найближчих 20 років. Тому не дивно, що зараз вона так активно використовується у багатьох різних сферах нашого життя.

1.3. Принципи роботи RFID

Системи радіочастотного розпізнавання не працюють на основі якогось єдиного стандарту чи технології, різні системи використовують різні стандарти. Проте метод передачі інформації у всіх них однаковий — радіочастотний.

Основними складовими системи RFID є зчитувач, антена та мітка. Але це лише вихідні точки, крім них — це програмне забезпечення для зчитувачів, драйвери, схема кодування та ідентифікації. Радіочастотна ідентифікація по суті виконує ті самі завдання, що штрих код чи магнітна стрічка. Але ця технологія є новішою, а також вона має багато переваг. Не потрібна пряма

видимість, можлива робота на відстані, набагато стійкіша до зовнішніх факторів .

Рисунок 1.1 — Приклад роботи RFID.

Технологія RFID не потребує прямого контакту або видимості між міткою та зчитувачем, дозволяючи одночасно читати кілька міток в межах антени. Системи RFID можна також використовувати в агресивних середовищах, теги RFID можна захистити від зовнішніх впливів радіопрозорим корпусом. Інформація обмінюється між зчитувачем та тегом крізь забруднення, фарбу, пару, воду, пластик, дерево.

Мітка складається з інтегральної схеми, яка зберігає інформацію та антени, яка забезпечує радіозв’язок, приклад на рисунку 1.2.

Мітки мають різну подачу енергії. Є активні, пасивні й напів пасивні [6, с.9-12]. У активних є власне джерело живлення. Їх можна використовувати в складних умовах. Пасивні не мають свого живлення. Вони отримують енергію від зчитувача через магнітне або електричне поле. Зазвичай вони використовуються для ідентифікації. Напів пасивні працюють від власного джерела живлення, але вони не використовують його для з’єднання зі зчитувачем. Воно потрібне лише для запуску мікросхеми. Детальніше порівняння наведено у таблиці 1.1.

Таблиця 1.1 — Порівняння активних та пасивних міток.

	Активний тег	**Пасивний тег**
Батарея	є	нема
Джерело живлення	вмонтоване в мітку	живиться від зчитувача
Наявність живлення	постійне	тільки в полі дії зчитувача
Необхідна потужність зчитувача	низька, потрібна лише для перенесення даних	висока, мусить живити тег
Сила сигналу від мітки до зчитувача	висока	низька
Діапазон дії	великий. більше 100 метрів	короткий або дуже короткий. 3 метри і менше
Термін служби тега	обмежений зарядом батареї(залежить від типу енергозбереження)	дуже довгий
Фізичний розмір	великий	малий
Одночасне зчитування декількох міток	зчитує 1000 тегів на площі 28000 квадратних метрів. 20 тегів, які рухаються зі швидкістю більше ніж 160км/год.	зчитує 100 тегів в межах 100метрів від зчитувача. 20 тегів, які рухаються із швидкістю менше 8км/год.
Передача даних із сенсорів	можливість безперервного контролю давача. записується час подій.	зчитує і передає дані із давача лише, якщо працює від зчитувача, не записується час подій
Пам'ять	велика кількість пам'яті. вимірюється в кілобайтах. є можливість пошуку.	невеликий розмір. вимірюється в байтах
Типове застосування	динамічні бізнес процеси. безпека, зондування, безпека та збереження даних, моніторинг залізничних вагонів, моніторинг безпеки вантажу	жорсткі бізнес процеси. маркування предметів, багажу, коробок, етикеток. контроль доступу. трекер відвідуваності
Ціна	висока. від 5$ до 100$	низька. до 0,5$

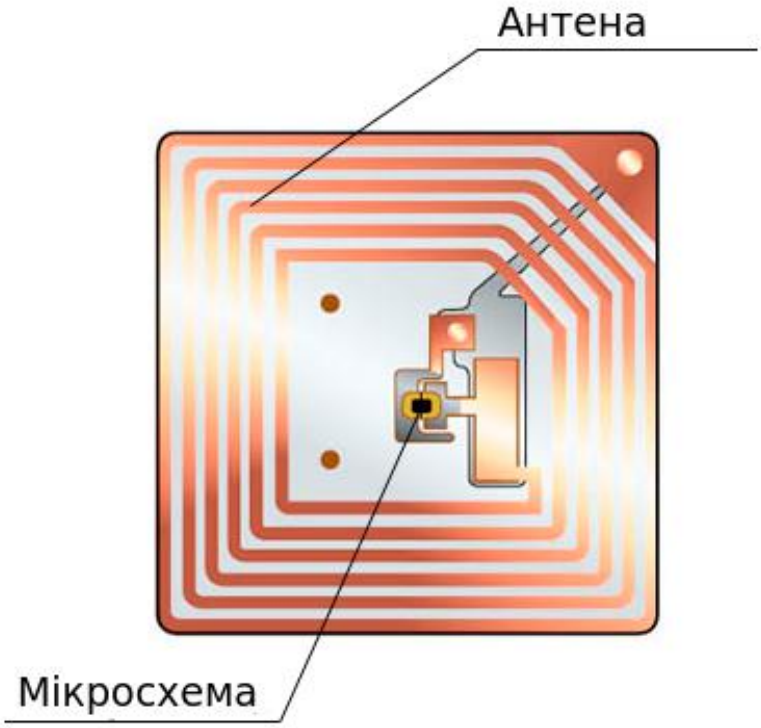


Рисунок 1.2 — Будова RFID тега.

Теги також відрізняються між собою по частотному діапазону. Є LF низькочастотні (в межах 130 кГц), HF високочастотні (в межах 13МГц), UHF надвисокочастотні (біля 900Мгц), приклад на рисунку 1.3. Найбільше використовуються високочастотні теги. Їх використовують в банківських системах, системах ідентифікації та контролю доступу, в транспортній сфері, в торговій промисловості. Але вони не можуть бути зчитані з великих відстаней та в поганих погодних умовах навколишнього середовища. Цю проблему частково вирішують надвисокочастотні мітки [6, с.85].

RFID мітки використовують три типи пам’яті: RO, WORM, RW.

RO (read only) — при виробництві в них записують дані лише один раз. Вони стійкі до підробки чи перезапису. В них є унікальний ідентифікаційний номер, який використовується для ідентифікації.

WORM (write once read many) — так само як і RO, вони мають унікальний id, але також у них є блок пам’яті в який можна записати інформацію один раз. Більше його перезаписати неможливо, але зате можна безліч раз зчитати.

RW (read and write) — у них крім ідентифікатора є блок пам’яті. Його можна перезаписувати і зчитувати скільки завгодно.

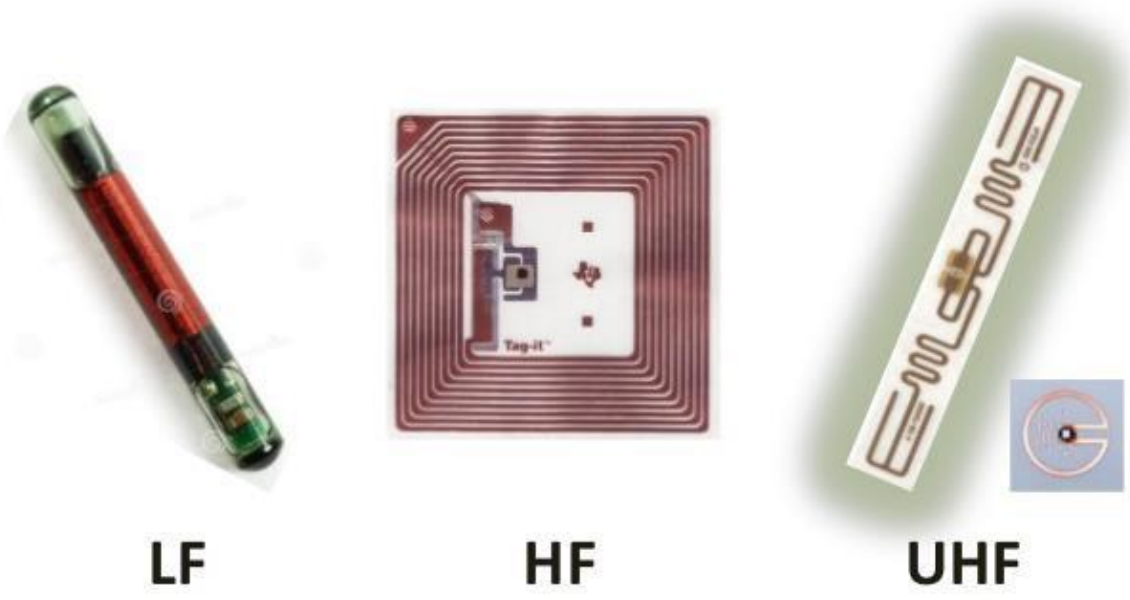


Рисунок 1.3 — Три приклади пасивних міток.

За способом виготовлення мітки мають багато різних варіацій. Їх виготовляють у вигляді пластикових карток, пластикових брелоків, наклейок, у вигляді колби яку вживлюють під шкіру.

Існує багато різних зчитувачів. Головне їхнє завдання це зчитати унікальний id, а також інформацію яка записана у пам’ять тега. Ще однією функцією є запис даних у пам’ять мітки. За дистанцією зчитування є ближні, середні (до 50 см) і дальньої дії (більше метра). Існують стаціонарні та переносні зчитувачі.

Оскільки зараз технологія RFID є достатньо доступною, надійною і простою у використанні, саме її найчастіше використовують під час розробки систем контролю відвідуваності або систем контролю доступу.

1.4. Історія розвитку NFC

Near Field Communication (NFC) — це відносно нова технологія безконтактного зв'язку, за основу якої взято технологію RFID.

Перший патент, пов'язаний з абревіатурою RFID, був виданий Чарльзу Уолтону в 1983 році. У 1997 NFC році вперше запатентована і використана в іграшках персонажів "Зоряних війн" Ендрю Уайтом і Марком Боретом. Пристрій дозволяв передавати дані між двома екземплярами, коли вони перебували в безпосередній близькості один від одного. Пізніше, у 2002 р. компанії Sony та Philips домовились між собою встановити технічну специфікацію та створили технічний план. Рік по тому NFC був затверджений як стандарт ISO/IEC, а пізніше як стандарт ECMA. Ще через рік компанії Nokia, Sony та Philips створили форум NFC. Того ж року Nokia запустила додаток для NFC на своїх найновіших моделях телефонів. У 2006 році з'являються перші технічні характеристики для тегів. На NFC Форумі, всього через 18 місяців після заснування офіційно окреслили архітектуру технології. У наступному році теги NFC Innovision вперше були випробувані споживачами, це відбулось у Великобританії. Через два роки, у січні, форум NFC випустив стандарти Peer-to-Peer, які використовувались для передачі контактів, URL адрес, з'єднання Bluetooth, тощо.

2010 року Innovision випустив набір дешевих розробок і патентів для масового ринку мобільних телефонів та інших пристроїв. Анонсовано Nokia C7: перший Symbian NFC телефон. Впродовж того ж року Samsung Nexus S: перший Android NFC смартфон. Ніцца (Франція) запускає проєкт "Ніцца місто безконтактного мобільного зв'язку", надаючи своїм жителям мобільні телефони, банківські NFC карти та пакет послуг, що охоплюють транспорт, туризм і послуги для студентів.

У 2011 році відбувається Google I/O, де демонструється можливість ініціювати гру, надіслати додаток, відео чи контакт. Випускається Symbian Anna у якій NFC став частиною операційної системи. Пристрої Research in Motion отримують першу сертифікацію від MasterCard Worldwide на свою

послугу PayPass. Sony у 2012 році представила публіці “Smart Tag” для зміни режимів і профілів на смартфоні, при близькому контакті із тегом. 2013 рік — Samsung і Visa оголошують про співпрацю для розробки мобільних платежів. Вчені IBM розробляють технологію захисту мобільної аутентифікації на основі NFC, яка працює на принципах схожих на безпеку двофакторної автентифікації.

У 2014 році AT&T, Verizon і T-Mobile випустили Softcard. Він працює на Android смартфонах із підтримкою NFC та на iPhone 4 і 5, якщо приєднується зовнішній NFC корпус. У 2015 році була запущена функція Google Android Pay, яка стала прямим конкурентом Apple Pay. Тепер технологія NFC широко використовується у розвинених країнах для безконтактної оплати у громадському транспорті, метро, супермаркетах, розважальних закладах. В Україні, як і в інших країнах світу також активно розвивається технологія NFC.

1.5. Принципи роботи NFC

NFC — це технологія зв'язку короткого діапазону для бездротової передачі даних, яка дозволяє обмінюватися даними між пристроями на відстані близько 10 сантиметрів. В принципі NFC — це один з варіантів RFID, механізм обміну радіочастотами, що зберігається в так званих мітках, заснований на стандартах ISO/IEC 18092 NFC IP-1, JIS X 6319-4 та ISO/IEC 14443, для безконтактних смарткарт [14]. Складається зі зчитувача та антени або мітки та антени. Зчитувач генерує радіочастотне поле, яке може взаємодіяти з міткою чи з іншим зчитувачем. Зчитувач — це пристрій, який працює в активному режимі зв'язку. Тег — це пристрій, який працює в пасивному режимі очікування.

NFC працює, використовуючи магнітну індукцію між двома петлевими антенами, розташованими близько одна до одної.Працює на частоті 13,56 МГц. Швидкість передачі даних варіюється від 106 кбіт/с до 424 кбіт/с на відстані приблизно 10 см. NFC використовує ініціатор і принаймні один цільовий пристрій; ініціатор активно генерує радіочастотне поле, яке може живити мітку.

Були й інші стандарти, які згодом були включені до стандарту NFC, такі як ISO 14443. Він описує діапазон частот, метод модуляції та протокол для пасивного обміну карт короткого діапазону RFID (до 10 сантиметрів).

Таким чином телефони з підтримкою NFC можуть взаємодіяти з уже наявною інфраструктурою зчитувачів. Зокрема, в режимі емуляції картки, пристрій NFC повинен передавати хоча б унікальний ідентифікаційний номер наявного зчитувача RFID. NFC — це технологія з відкритою платформою. Стандартизована в ECMA-340 та ISO/IEC 18092. Ці стандарти визначають модуляцію, кодування, передачу та радіочастотну структуру інтерфейсу пристрою NFC, а також схеми ініціалізації та умови, необхідні для контролю конфліктних ситуацій. Вони також визначають протокол передачі, включаючи протокол активації та спосіб зв'язку.

Рисунок 1.4 — Схема роботи NFC.

На сьогодні NFC Форум випустив 16 специфікацій, які дозволяють усім зацікавленим виробникам створювати нові продукти. Окрім чинних стандартів, Форум зібрав найкраще з них у документи, що описують роботу пристроїв, які підтримують NFC. Наприклад, NFC Analog Technical Specification стосується аналогових радіочастотних характеристик пристрою з підтримкою NFC. Ця специфікація описує вимоги потужності антени, передачі, вимоги приймача та форми сигналу. Специфікація NFC Analog 2.0 ввела активний режим зв'язку

P2P та у режимі опитування технологію NFC-V. Версія 2.0 забезпечує повну сумісність із пристроями, які відповідають ISO/IEC 14443 або ISO/IEC 18092.

Відповідно до цих специфікацій для пристроїв NFC, існують такі способи зв'язку: NFC-A, NFC-B, NFC-F та п'ять типів тегів NFC. Пристрої NFC можуть мати активний або пасивний зв’язок і підтримувати один або декілька з трьох режимів.

Тип зв'язку **NFC-A** заснований на ISO/IEC 14443A для безконтактних карт. Типи зв'язку відрізняються режимами кодування та модуляції сигналу. NFC-A використовує код Міллера та амплітудну модуляцію. Бінарні дані передаються зі швидкістю приблизно 106 Кбіт с, а сигнал повинен змінюватись від 0% до 100%, щоб розрізняти двійкові 0 та 1.

Тип зв'язку **NFC-B** заснований на ISO/IEC 14443B для безконтактних карт. NFC-B використовує метод манчестерського кодування. Двійкові дані також передаються зі швидкістю 106 кбіт/с. Так тут замість 100% використовується амплітуда 10% для низького рівня та 100% для високого. У манчестерському кодуванні перехід від низького до високого рівня є бінарним нулем, а перехід від високого до низького — бінарною одиницею.

Тип зв'язку **NFC-F** заснований на стандарті FeliCA JIS X6319-4, також відомому як FeliCa. Стандарт регулюється японським JICSAP. У них ця технологія і найпопулярніша. Використовується швидкість завантаження від 212 кбіт/с до 424 кбіт/с, використовується манчестерське кодування та амплітудна модуляція.

NFC пристрої мають три режими: читання або запис, емуляція картки і одноранговий зв’язок.

У режимі **емуляції карт** пристрій відтворює роботу банківської карти, перепустки, дисконтної карти, тощо. Програма не потребує окремого криптопроцесора, вона працює на основному процесорі пристрою. Під час з’єднання він емулює роботу картки і зчитувач без проблем передає і отримує інформацію. Найчастіше ця операція виконується при оплаті за допомогою NFC, рисунок 1.5.

Рисунок 1.5 — Режим емуляції електронної картки.

У **одноранговому (P2P) режимі** два пристрої зв’язуються між собою і отримують змогу передавати файли або іншу інформацію, таку як різні налаштування WiFi чи Bluetooth з’єднань, або дані з телефонної книги. Для з’єднання достатньо піднести на близьку відстань два пристрої із підтримкою технології, рисунок 1.6. Зазвичай NFC використовується лише для ідентифікації між пристроями, а уже безпосередньо передача даних відбувається через Bluetooth або WiFi.

Рисунок 1.6 — Одноранговий режим.

У режимі **читання/запису** пристрій зчитує інформацію, із пасивного тега, який може бути вбудований в рекламні щити або вивіски. Також може обмінюватись даними з іншими пристроями, які працюють у режимі читання/запису, рисунок 1.7. Апарат, може зчитати або перезаписати

інформацію на іншому апараті, який працює у цьому режимі. У теги можуть записати багато корисної інформації. Їх можна використовувати на зупинках громадського транспорту, отримувати маршрути чи певні знижки у магазинах та супермаркетах. Також в теги можна записати дані, які потрібно буде час від часу перезаписувати. Наприклад, це може бути пароль від доступу до WiFi, тощо.

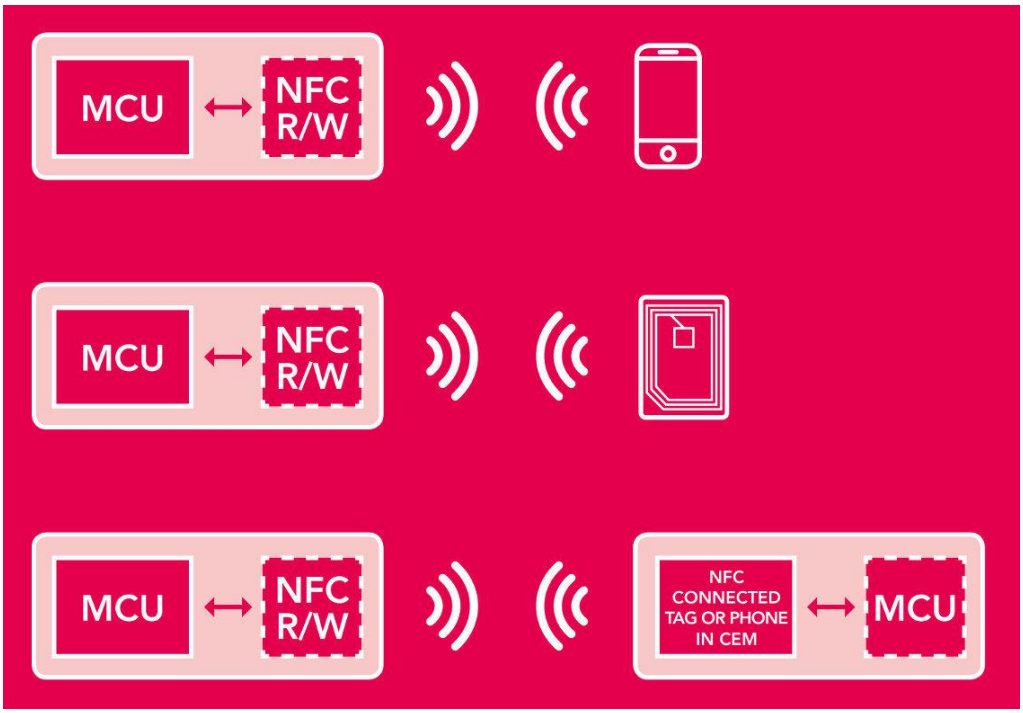


Рисунок 1.7 — Режим читання/запису NFC.

Пристрої можуть працювати в активному режимі або в пасивному режимі. В активному режимі обидва пристрої з чіпом NFC генерують електромагнітне поле та обмінюються даними. У пасивному режимі існує лише один активний пристрій, а інший (наприклад, тег) використовує це поле для обміну інформацією.

NFC тег — це пасивний пристрій, який підтримує режим читання та запису. За специфікацією NFC Форум є п’ять типів міток, таблиця 1.2.

Таблиця 1.2 — Технічні характеристики NFC тегів.

Тип мітки	I тип	II тип	III тип	IV тип	V тип
Стандарт	ISO 1443A	ISO 14443A	ISO 18092 JIS-X-63 19-4	ISO 14443A	ISO 14443A MF1 IC S50
Об’єм пам’яті	96 байт	48-144 байта	1/4/9 кілобайт	4/32 кілобайт	192/768/3584 байт
Швидкість обміну	106 кбіт/с	106 кбіт/с	212/424 кбіт/с	106/212/424 кбіт/с	106 кбіт/с
Доступ до даних	Читання та запис або лише читання				
Захист даних від зіткнення	Нема	Є			

1.6. Застосування технології NFC/RFID для ідентифікації та доступу

Безконтактні технології широко застосовуються для управління доступом чи ідентифікації. Один з найпоширеніших прикладів застосування це готельний бізнес. NFC чи RFID безпосередньо забезпечує контроль доступу до конкретних номерів або груп номерів. Переважно це просто номер, але якщо замовник має якісь особливі привілеї він може отримати доступ наприклад до відпочинкової зони чи будь-якого іншого місця з обмеженим доступом. Ця технологія також допомагає власникам відстежувати кількість відвідувачі готелю чи кількість людей, які проходять через якусь певну зону.

Перед відвідувачами з’являються абсолютно нові можливості. Вони можуть попередньо за допомогою своїх смартфонів забронювати номер, а після прибуття самостійно заселитись чи оплатити рахунок. Не потрібно зайвий раз чекати поки видадуть ключі чи картку. Після приїзду до готелю можна одразу

ж відкрити двері зарезервованої кімнати за допомогою свого смартфона. Це однаково вигідно і для відвідувачів і для власників бізнесу. Відпадає необхідність носити із собою фізичний ключ. Відповідно ви не можете більше його загубити, а це також дозволить зекономити кошти готелю.

Коли NFC застосовується для контролю доступу на якийсь захід, виставку чи розважальний комплекс, власники можуть зменшити витрати. Можна проводити електронну реєстрацію відвідувачів.

Індустрія систем контролю та управління доступом розробляє рішення для різних сегментів ринку. Історично склалось, що в якості ідентифікаторів використовувались низькочастотні RFID теги. Їх застосовували разом з програмами, які дозволяють в режимі реального часу, підключеним в систему пунктам доступу зчитувати теги і перевіряти сервер або контролер, для ідентифікації чи підтвердження доступу [2].

В останні роки галузь доклала значних зусиль для модернізації цієї інфраструктури та переходу від підтримки лише низькочастотного обладнання до більш функціональних високочастотних пристроїв. Це дозволяє розширити функціонал систем щодо звичайного контролю доступу.З’являється можливість створення пропуску, ідентифікація особистості, оплата проживання та інші можливості. Компанія Mifare стала найвідомішою, серед тих хто реалізує новий стандарт.

Також не варто забувати, що основним пристроєм у системах управління доступом є контролер. Він також повинен підтримувати відповідний функціонал. Найважливішим є вміння контролера працювати із секторами карт, які мають криптозахист.

У наш час хмарні сервіси дозволяють розробникам без проблем мігрувати на смартфони програми для безконтактних карт контролю доступу. Смартфон може обробляти всі необхідні права і функції. Якщо смартфон підтримує стандарт NFC, то він може зберігати і надавати потрібні дані зчитувачу, так само як це роблять безконтактні карти. Сертифікати можна генерувати в режимі реального часу та зберігати в додатку з підтримкою Host Card Emulation.

Смартфон, крім своїх звичних функцій, починає виконувати усі функції смарт карт. Може бути електронним квитком, ключем чи системою контролю відвідування.

Контроль доступу за допомогою NFC є надзвичайно зручним на великих підприємствах, заводах чи установах, де відбувається контроль та управління фізичним доступом. Відпадає потреба зберігати величезну кількість звичайних ключів від усіх замків, а також легко можна контролювати доступ осіб до тієї чи іншої кімнати. Також великим плюсом є можливість використання замків без власного джерела живлення. В його ролі буде смартфон із NFC, який під час з'єднання буде передавати на замок достатньо електроенергії для його відмикання.

Також зараз почали застосовувати NFC та RFID в автомобільній сфері. Приклавши смартфон до спеціального місця на дверцях автомобіля або панелі приладів можна відкрити або заблокувати двері, а також завести автомобіль.

На жаль поки що NFC не доступний у кожному смартфоні, але з кожним роком їх стає усе більше, технологія стрімко набирає популярність. Технологія безконтактного контролю доступу стрімко розвивається. Завдяки цьому на ринку з'являється багато нових компаній. А завдяки конкуренції покращується якість пристроїв та їх доступність. Ці технології вже використовуються у майже кожній сфері нашого життя. Це безперечно зручно і плюсів від їх використання значно більше ніж мінусів.

Висновки до розділу

У даному розділі було описано загальні принципи роботи систем контролю відвідування. Також було здійснено опис історії створення та розвитку таких безконтактних технологій, як RFID і NFC. Було описано також принципи їх роботи.

2. ВИБІР АПАРАТНИХ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ КОНТРОЛЮ ВІДВІДУВАНОСТІ

2.1. Компоненти СКВ

На ринку як закордонному так і вітчизняному існує велика кількість готових систем контролю відвідуваності або систем контролю та управління доступом. Їхніми послугами користується чимало державних установ, представників бізнесу та навчальних закладів. Найпростішим прикладом може бути домофон, який встановлений практично у кожному багатоквартирному домі біля вхідних дверей. Він теж використовує технологію радіочастотної ідентифікації.

Абсолютна більшість СКВ для взаємодії із користувачем використовує RFID/NFC технології. Є масштабні системи, які охоплюють цілі будівлі чи мережу закладів та мають безліч функцій. Також є невеликі пристрої, які використовують буквально на одному чи декількох входах (домофон, про який було сказано вище). Для виготовлення тих чи інших систем використовують багато різних компонентів. Від вибору цих компонентів напряму залежить ціна системи, її функціональність, зручність використання, можливість масштабування, покращення, надійність та безпека.

Рисунок 2.1 — Комплектація бюджетної СКВ.

До основних апаратних компонентів СКВ можна віднести: мікроконтролер, зчитувач тегів, інструмент сповіщень (дисплей, динамік або світлова індикація). До програмних відносяться: прошивка контролера, база даних (локальна або віддалена), веб або мобільний застосунок (потрібен не завжди). На ринку електроніки є великий вибір усіх цих компонентів. Це дозволяє підібрати їх так, щоб співвідношення ціна — якість було оптимальним. У даному розділі описано вибір складових для розробки власної системи контролю відвідуваності студентів при мінімальній вартості та із забезпеченням головних функцій.

2.2. Вибір мови програмування для розробки плати та веб застосунку

При створенні СКВ потрібно здійснити вибір програмних засобів для двох напрямів. Для розробки першої частини системи це прошивка апаратної частини й відповідно мова, на якій буде створюватись програма для взаємодії з апаратними складовими системи. Для розробки другої частини потрібно вибрати мову для написання веб застосунку, який буде працювати на віддаленому сервері та там же буде розміщена база даних. Від вибору програмних інструментів залежить швидкість створення коду, гнучкість системи, можливість підтримки коду в майбутньому.

Найпопулярнішими мовами програмування на даний час є Java, C++, Python, JavaScript, C#, PHP. З їх допомогою створюють сервера, десктопні програми, веб додатки. Для програмування мікроконтролерів зазвичай використовують низькорівневі C/C++, або Assembler. C-подібні мови чи асемблер безумовно краще пристосовані для роботи з апаратною частиною завдяки своїй низькорівневості. Проте часто програмісти витрачають багато часу не конкретно на розв’язання поставленої задачі, а на усунення різних помилок, які виникають при розробці, тому що ці мови досить складні і

потрібні безпосередньо глибокі знання інформатики і добре розуміння, як працює обладнання на низькому рівні.

Тому в останні роки, разом із збільшенням потужності контролерів і плат, почалось створення та розвиток інших альтернативних програмних інструментів, які можна використати замість C/C++. Це фреймворк Python'a — Micropython, та навіть фреймворк на базі JavaScript — Espruino. Вони працюють повільніше, але зате значно спрощують написання програм під обладнання, оскільки базуються на високорівневих мовах програмування. Для створення більшості нескладних систем, де не є критично важливою швидкість роботи, вони цілком підходять.

Для розробки програмних компонентів системи контролю відвідування студентів було вибрано мову програмування Python. Це високорівнева інтерпретована мова, зі строгою динамічною типізацією, яка підтримує функціональну, імперативну та об'єктноорієнтовану парадигми програмування [4]. Вона дозволяє програмісту безпосередньо зосередитись на вирішенні завдань, оскільки її синтаксис є достатньо зручний і зрозумілий.

Не потрібно тратити багато часу на освоєння цієї мови. Завдяки підтримці великої кількості модулів та пакетів модулів, можна використовувати уже певні готові рішення, це також суттєво збільшує швидкість програмування на мові Python. Важливо є те, що стандартна бібліотека теж багата на різні засоби. Вона забезпечує користувача різними модулями, які дозволяють працювати із мережею, файлами різних форматів, операційною системою, тестуванням, регулярними виразами та багатьма іншими речами.

Дану мову застосовують у багатьох різних галузях для написання сценаріїв та створення прикладних програм. Також за потреби, коли потрібно збільшити швидкість виконання програми, код можна розширити вставками із C чи C++. Python може працювати на усіх відомих платформах. Не зважаючи на усі плюси є також і декілька мінусів, один із них це низька швидкодія. Але навіть цей недолік можна знівелювати зменшенням швидкості написання коду. За даними із різних джерел Python є однією із найпопулярніших мов

програмування, а також належить до мов, які найшвидше розвиваються. Оскільки за допомогою Python'а можна і розробити сайт, і запрограмувати контролер, повністю зосередившись на виконанні поставлених задач, то він ідеально підійде для розробки власної нескладної СКВ.

2.2.1. MicroPython.

Для роботи з апаратною частиною було вирішено використати MicroPython. Це програмне забезпечення, яке значною мірою, окрім кількох винятків, використовує синтаксис Python третьої версії, написане на C і оптимізоване для роботи на мікроконтролері.

Основна відмінність від Python це те, що MicroPython створений для роботи з обмеженими ресурсами. Тому він не постачається із повною стандартною бібліотекою Python. Включені лише невелика підмножина стандартної бібліотеки, якої зазвичай достатньо для комфортної роботи. Але зате доступні бібліотеки для доступу до апаратного забезпечення на низькому рівні. Це означає, що є модулі, які легко можуть отримати доступ та взаємодіяти із GPIO. GPIO — це інтерфейс, за допомогою якого відбувається взаємодія між контролером чи мікропроцесором із периферійними пристроями. Крім того для пристроїв, які підтримують WiFi зв'язок, є модулі для роботи із мережевим з'єднанням [8].

Поява MicroPython'а робить програмування цифрової електроніки надзвичайно простим та швидким. Особливо добре він підходить для тих, хто раніше не мав такого досвіду. Його мета зробити програмування апаратних засобів якомога простішим. MicroPython використовують дослідники, викладачі, в домашніх проєктах, а також навіть у деяких комерційних.

Ще однією хорошою особливістю є те, що прошивка поставляється із так званим інтерактивним середовищем REPL. Воно дозволяє під'єднатися до плати й виконувати команди в режимі реального часу без необхідності компіляції чи завантаження коду у пам'ять пристрою. Створено декілька

зручних IDE: uPyCraft, Thonny, можна використовувати термінал або веб сервіс.

За замовчуванням у пам’яті обов’язково має бути присутній файл boot.py. При запуску пристрою спочатку виконують команди саме з цього файлу. Зазвичай у нього записують налаштування конфігурацій. Для запуску головного коду рекомендується створити файл main.py. Він виконується одразу після boot.py.

2.2.2. Django.

Для створення веб додатку системи відвідування було обрано Django. Це як і MicroPython фреймворк мови програмування Python. Він безкоштовний і відкритий, створений для того щоб веб сайти можна було розробляти швидше і з мінімальними часовими витратами, використовує шаблон проєктування MVC [1]. При веб розробці дуже часто потрібно виконувати дії, які постійно повторюються, саме для цього і створюють веб фреймворки. Для цих цілей і був створений Django у 2005 році. Він став одним із найбільш використовуваних інструментів.

Він надає усі необхідні знаряддя для швидкого і прозорого написання коду. Вони спрощують роботу із аутентифікацією, адмініструванням, шаблонами, безпекою. Створені сайти можна без проблем масштабувати, вони витримують високий трафік. Сайт складається із декількох додатків, які потрібно розмежовувати між собою.

Для створення бази даних описується модель за допомогою класів і на її основі уже генерується схема бази. Підтримується робота із всіма найпопулярнішими системами управління базами даних. Не обов’язково витрачати час на розроблення адмін панелі, можна використовувати уже готову, яку по замовчуванню надає фреймворк. Для створення API зручно використовувати Django Rest Framework. Він працює із стандартними моделями і дозволяє створити зручний API інтерфейс для будь-якого проєкта.

Що є дуже важливо, оскільки контролер із сервером буде взаємодіяти саме через API.

Отже, Django ідеально підходить для створення СКВ. Він надає усі потрібні інструменти, розробка не повинна бути надто складною і систему можна зробити достатньо гнучкою.

2.3. Вибір мікроконтролера

Мікроконтролер — це мікропроцесорна система на одній інтегральній схемі. Він складається з одного або кількох процесорів, пам'яті і пристроїв вводу та виводу. Мікроконтролери використовуються для керування вбудованими системами чи електронними пристроями. На них базується робота різних систем контролю доступу, управління двигунами, різних медичних систем, іграшок, пультів управління, побутових приладів, широко застосовуються в сфері Інтернету речей і в багатьох інших пристроях. Їх розмір і вартість значно менші ніж у пристроїв які використовують окремий мікропроцесор. Для вибору доступна велика кількість контролерів, які відрізняються між собою за різними параметрами. Їх можна розділити на три основні класи: 8-розрядні, 16 та 32-розрядні, цифрові сигнальні процесори [5].

Для зменшення споживання енергії деякі екземпляри можуть працювати на низьких частотах. Переважно вони зберігають свою функціональність безпосередньо під час очікування тих чи інших подій. Енергоспоживання під час сну є дуже малим, що дуже добре підходить для систем, в яких є важливим збереження заряду акумулятора протягом якнайдовшого періоду часу.

При проєктуванні не лише СКВ, а й будь-якої іншої автоматизованої системи вибір мікроконтролера є одним із найважливіших факторів. Широко набрали популярності так звані плати розробника. Вони складаються із контролера, а також додатково присутні і інші компоненти, такі як WiFi чи GSM антена, GPS модуль, тощо. Таке рішення дозволяє суттєво зекономити час і витрати на розробку власної плати. Уже готові екземпляри підійдуть для

вирішення більшості задач. Найпопулярнішими є плати на базі мікроконтролерів ESP, AVR та STM. Також широко використовується Raspberry Pi. Вона значно дорожча, але відповідно її можливості набагато вищі.

Для розробки нескладної студентської СКВ буде достатньо недорогої плати, яка має взаємодіяти із зчитувачем та зберігати дані у власній пам’яті або передавати їх на сервер. Оскільки система буде взаємодіяти із сервером потрібно вибрати плату, з якою буде легко реалізувати такий зв’язок. Не менш важливим параметром є взаємодія із платою, тобто складність її підключення та прошивки мають бути мінімальними.

Відштовхуючись від усіх параметрів було прийняте рішення вибрати для роботи плату NodeMCU Lolin третьої версії. Це недорога платформа для Інтернету речей з відкритим кодом. Спочатку вона складалась із програмного забезпечення, яке працювало на ESP8266 Wi-Fi системи на чіпі від Espressif Systems і апаратного забезпечення, яке базувалось на модулі ESP-12. Пізніше була додана підтримка 32-бітного ESP32. На рішення також вплинуло те, що для цієї плати є портована прошивка MicroPython і різні бібліотеки для периферійних пристроїв, які можна безкоштовно використовувати.

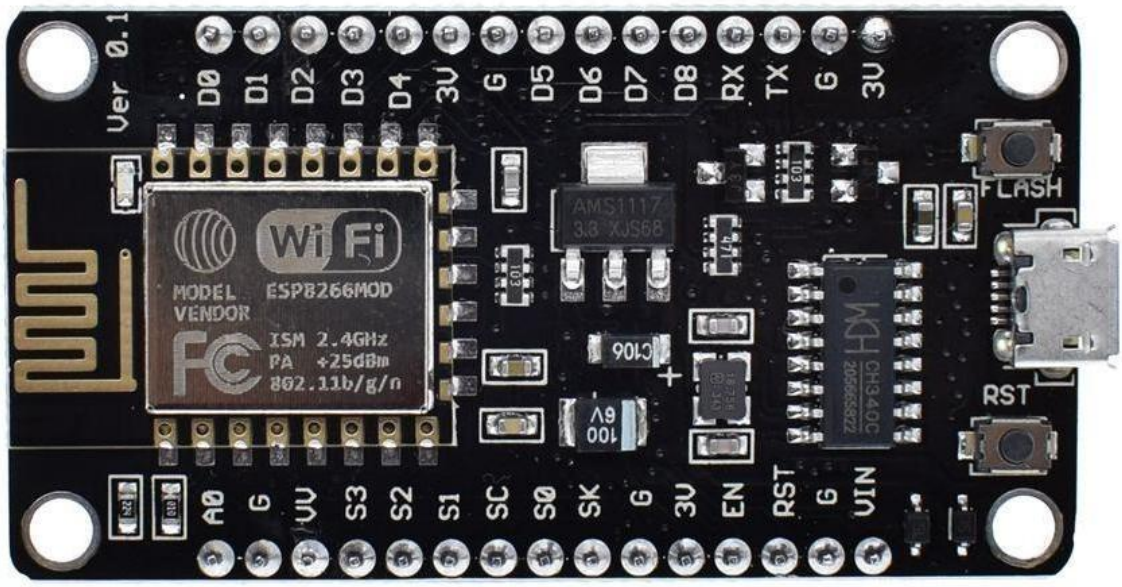


Рисунок 2.2 — Плата NodeMCU v3.

NodeMCU — це мікропрограмне забезпечення з відкритим кодом, для якого доступні проєкти плати для прототипування з відкритим кодом. Назва складається із англійських слів “Node” (вузол) та “MCU” (мікроконтролер). Цей термін більше стосується прошивки, а не пов'язаних з ним комплектів розробки. І прошивки, і конструкції плати є відкритим продуктом.

Прошивка використовує сценарій скриптової мови Lua. Вона базується на проєкті eLua та побудована на Espressif Non-OS SDK для ESP8266. Він використовує багато проєктів з відкритим кодом, таких як lua-cjson та SPIFFS. Через обмежені ресурси користувачі повинні вибрати модулі, що відповідають їхньому проєкту, та створити програмне забезпечення з урахуванням їх потреб. Також була реалізована підтримка 32-бітного ESP32.

Зазвичай обладнання для створення прототипу — це друкована плата, яка функціонує як подвійний вбудований пакет (DIP), котрий інтегрує USB-контролер з меншою поверхневою платою, що містить контролер та антену. Вибір формату DIP дозволяє легко створювати прототипи на макетних платах. Спочатку проєкт був заснований на модулі ESP-12 ESP8266, який являє собою Wi-Fi SoC, інтегрований з ядром Tensilica Xtensa LX106, широко використовується в додатках Інтернету речей.

NodeMCU був створений майже одразу після виходу ESP8266. 30 грудня 2013 року компанія Espressif Systems почала випускати ESP8266. Проєкт NodeMCU стартував 13 жовтня 2014 року, коли Хонг створив комміт на GitHub першого файлу nodemcu-прошивки. Два місяці по тому, проєкт був розширений для включення відкритої апаратної платформи, коли розробник Хуанг Р. зробив гербер файл на ESP8266, названий DevKit v0.9. Пізніше в тому ж місяці, Туан РМ портував MQTT клієнтську бібліотеку з Contiki на платформу ESP8266, і створив комміт у проєкті, плата змогла отримати підтримку протоколу IoT MQTT, використовуючи Lua для доступу до MQTT. Ще одне важливе оновлення було зроблено 30 січня 2015 року, коли Девсаурус портував u8glib до проєкту, що дозволило платі легко керувати LCD, OLED і навіть VGA-дисплеями. Влітку 2015 року оригінальні творці відмовилися від проєкту

і група незалежних розробників взяла на себе участь у розвитку платформи. До літа 2016 року NodeMCU включав понад 40 різних модулів.

Плата зручна тим, що можна без проблем працювати із Wi-Fi з'єднанням. Підтримує WiFi стандарт 802.11 b/g/n. Для живлення потрібно від 4.5 до 9 В, живиться від USB. При обміні даними споживає близько 70 мА, а в режимі очікування менше ніж 200 мкА. Можна зробити висновок, що плата є досить енергоефективною. Підтримує UART та GPIO інтерфейси обміну даних. Можна прошивати за допомогою комп'ютера з'єднавшись по USB, або із сайту через WEBRepl, по WiFi з'єднанню. Це досить зручно, оскільки не потрібно ніяких додаткових програматорів.

2.4. Вибір зчитувача

При виборі зчитувача тегів потрібно враховувати протоколи з якими він вміє працювати, швидкість зчитування, робочу напругу та інтерфейс по якому він з'єднується із мікроконтролером, його компактність. Якщо пристрій буде використовуватись надворі, потрібно врахувати його стійкість до важких погодних умов.

Найбільш популярними і доступними є два варіанти: PN532 і RFID-RC522 (рисунок 2.3). Було прийнято рішення використовувати саме останній, оскільки у нього нижча вартість і усі параметри повністю підходять для розробки СКВ. Даний модуль працює на частоті 13.56 Мгц. Це означає, що для ідентифікації можна використовувати як NFC так і RFID теги. Його часто використовують студенти, радіолюбителі, а також комерційні розробники [2, с.211]. Використовуючи цей модуль, можна отримати доступ до усіх потрібних безконтактних функцій. Він підтримує інтерфейси SPI, I2C та UART. Вибрати потрібний із них можна установкою логічного рівня на певних виводах мікросхеми. Для живлення потрібно 3.3 В. Зчитує мітки на відстані до 6 сантиметрів. Може як читати так і записувати інформацію. Максимальна швидкість передачі даних близько 10 мегабітів на секунду.

Рисунок 2.3 — RFID модуль RC522.

Є готові бібліотеки для роботи із даним модулем. З мінусів є те, що він не підтримує роботу із NFC смартфона, працює тільки із мітками. Проте для використання у даній СКВ цього буде достатньо.

2.5. Вибір модуля індикації

У будь-якій системі контролю доступу повинен бути реалізований певний спосіб індикації. Користувач має розуміти чи успішно відбулась ідентифікація і бачити її результат. Для цього можна використовувати світлове або звукове сповіщення, або виводити інформацію на екран. Останній спосіб найзручніший, оскільки можна виводити будь-яке потрібне сповіщення. Це значно розширює можливості і підвищує зручність користування системою.

Для реалізації індикації було вибрано OLED I2C 128x32 дисплей, показано на рисунку 2.4. Як уже зрозуміло із назви він працює по I2C протоколу і має компактні розміри, яких для виводу основної інформації цілком достатньо. Завдяки технології OLED даний дисплей досить енергоефективний. Переважно його використовують у при виготовленні компактних пристроїв. Найбільшими його плюсами є яскравість, зображення добре видно на яскравому світлі і безумовно ціна.

Рисунок 2.4 — OLED дисплей.

Висновки до розділу

В даному розділі було проведено аналітичний вибір апаратних і програмних компонентів для створення системи відвідування, а також обґрунтовано причини вибору тих чи інших модулів та програмних засобів. Здійснено короткий опис кожного елемента.

3. РОЗРОБЛЕННЯ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СКВ

3.1. З’єднання мікроконтролера із периферійними пристроями

Безконтактна система контролю відвідуваності буде працювати на платі NodeMCU v3, яка в свою чергу працює на мікроконтролері ESP8266. Також потрібні такі периферійні пристрої, як зчитувач тегів та екран для сповіщень. У попередньому розділі було описано і обґрунтовано вибір зчитувача RFID-RC522 та OLED дисплея. Плата живиться від 5В, які подаються через microUSB, а компоненти уже в свою чергу живлення з плати через 3.3 вольтові виводи. Загальна схема підключення периферії до контролера показана на рисунку 3.1.

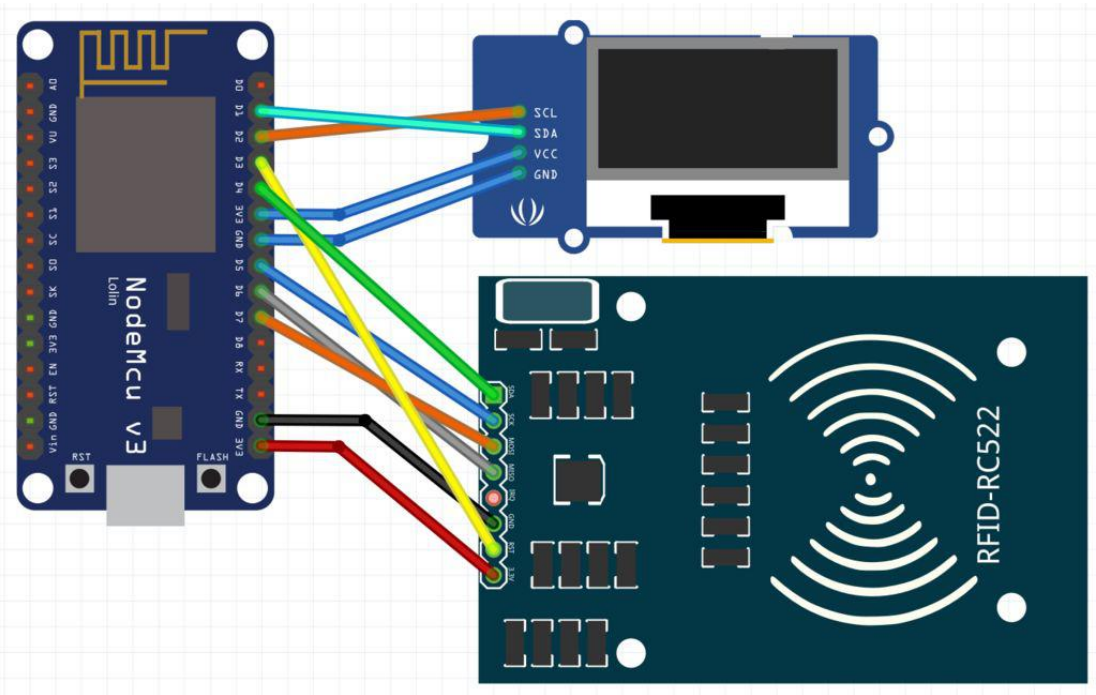


Рисунок 3.1 — Схема з’єднання контролера з периферією.

3.1.1. Підключення зчитувача.

Зчитувач підключається через інтерфейс SPI. Він використовується для простого і недорогого швидкого з’єднання. Це синхронна 4-провідна шина.

З’єднання відбувається за конфігурацією ведучий/ведений, лише головний генерує імпульси синхронізації [3]. У схемі існує лише один головний майстер, кількість ведених може змінюватися. Завжди працює у повнодуплексному режимі [3]. Дані передаються завдяки таким сигналам:

MOSI - інформація передається від головного пристрою до залежного.

MISO - інформація передається від залежного пристрою до головного.

SCK - передається тактовий сигнал до залежних пристроїв.

SS - сигнал, який контролює початок і завершення сеансу з’єднання.

В RC-522 ще є вихід IRQ, призначений для переривання (в такому випадку він не використовується), GND — це земля, Vcc — живлення, а RST використовується для скидання. Їх розміщення показані на рисунку 3.2.

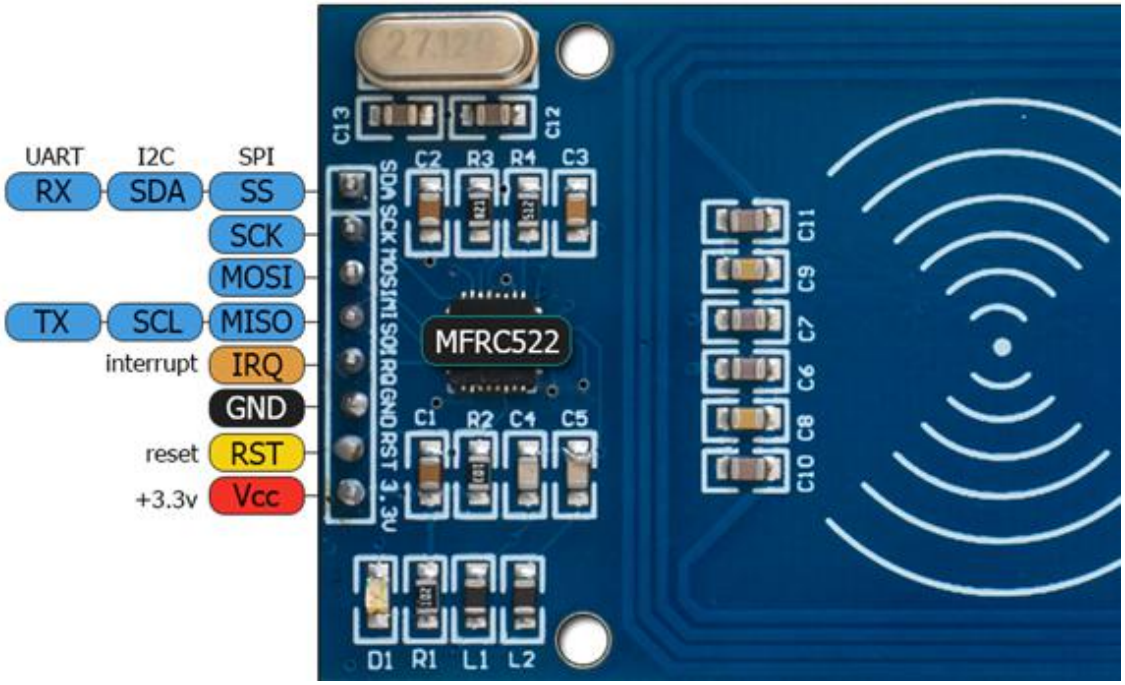


Рисунок 3.2 — Призначення виводів на модулі MFRC-522.

Розміщення виводів на платі можна переглянути на рисунку 3.3. Виходам від D1 по D8 можна програмно задати тип сигналу. Але для кращої швидкодії при використанні SPI інтерфейсу краще використовувати виводи D5 — D8, бо

у них вже на апаратному рівні задані сигнали SPI [2]. Якщо використовувати їх, то не потрібно витрачати час на ініціалізацію неоголошених виходів. Шлях підключення модуля до контролера показаний у таблиці 3.1.

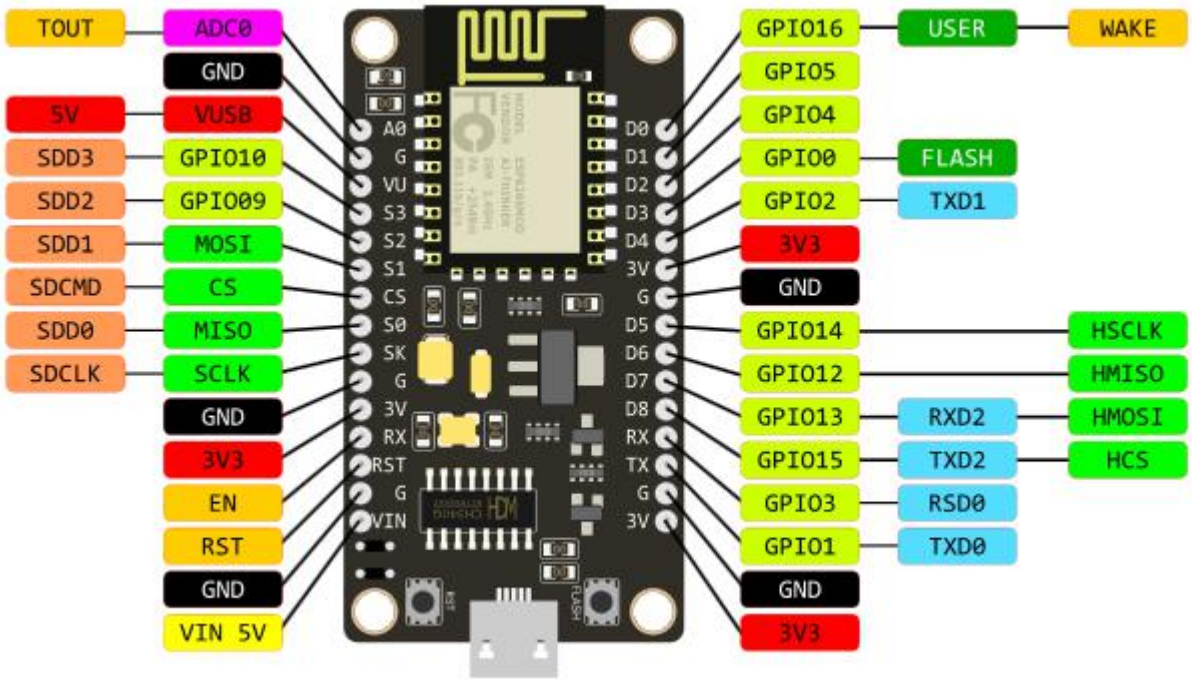


Рисунок 3.3 — Позначення виводів плати NodeMCU V3.

Таблиця 3.1 — Підключення зчитувача до плати.

Позначення виводу на модулі	Позначення виводу на платі
SDA	D8
SCK	D5
MOSI	D7
MISO	D6
IRQ	-
GND	G
RST	D4
3.3 V	3V

3.1.2. Підключення дисплея.

Дисплей буде взаємодіяти із платою по шині I2C. Дані передаються по кабелю даних і кабелю тактів. Є основний і залежний, основний генерує такти. Швидкість значно менша ніж у SPI. В екрана є такі виходи, як GND — земля, VCC — живлення, SCL — лінія послідовного тактування, SDA — послідовна лінія даних [3]. Схема підключення зображена на рисунку 3.4. На таблиці 3.2 показано шлях підключення ніжок екрана до плати.

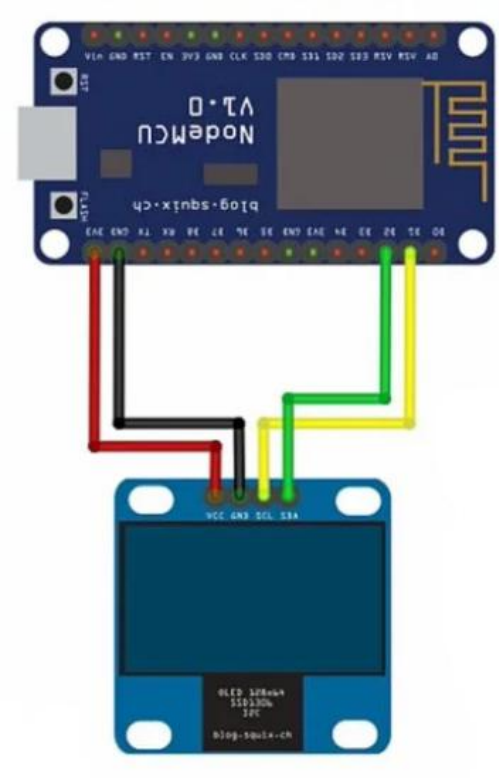


Рисунок 3.4 — Схема з’єднання OLED дисплея із мікроконтролером.

Таблиця 3.2 — Підключення портів дисплея до портів мікроконтролера.

Позначення виводу на дисплеї	Позначення виводу на контролері
SCL	D1
SDA	D2
GND	GND
VCC	3.3V

3.2. Програмування мікроконтролера

Після того як усі апаратні частини підключені потрібно запрограмувати контролер. Як уже було сказано вище, для цієї задачі буде використано MicroPython. Для початку NodeMcu треба прошити. Інструкція описана в офіційній документації [13]. Щоб плата могла взаємодіяти із різними модулями потрібно записати на контролер відповідні бібліотеки для кожного модуля. Для того щоб не створювати їх з нуля, можна використати уже готові, які знаходяться на github.com і надаються для вільного доступу з безкоштовною ліцензією [11; 12].

В бібліотеці для зчитувача при ініціалізації шини використовувався програмний SPI. Оскільки є можливість підключення зчитувача через апаратний SPI, було переписано частину драйверу. Це дозволило зменшити час зчитування тега із приблизно 1.5 с до 0.9 с., що є значно відчутним покращенням. Після того як підготовлені драйвери завантажені на контролер було розроблено код, який дозволяє ідентифікувати мітки через зчитувач, виводити інформацію на дисплей та взаємодіяти із сервером через з'єднання WiFi, видрук коду наведено у додатку 1.

Для запуску системи потрібно під'єднатись до плати або через порт комп'ютера або безпровідним способом через REPL, при умові, що подано живлення через microUSB. Зразу при появі живлення, виконується код у файлі boot.py. Там відбувається перевірка підключення до WIFI. Дана система налаштована для запуску у ручному режимі. Тому щоб запустити головний код потрібно в консолі імпортувати модуль main.py, викликати функцію start() і передати їй як аргумент назву робочого режиму, main.start(scan) або main.start(register), приклад коду наведено в додатку 1.

Система може працювати у двох режимах: сканування міток або реєстрація нових міток у системі. Незалежно від вибраного режиму алгоритм роботи контролера однаковий. В безперервному циклі мікроконтролер опитує зчитувач, якщо мітка потрапляє в радіус його дії, контролер зчитує унікальний код мітки і передає у POST запиті, за допомогою стандартної бібліотеки

urequests, через API дані на сервер. Після обробки сервером запиту, контролер виводить на дисплей результат операції. Незалежно від режиму, на сервер завжди передається ID тега.

Якщо вибрано режим сканування, то сервер перевіряє чи ID зареєстрований у системі, якщо ні, то відсилає назад відповідне повідомлення. Якщо зареєстрований, то відповідно записується час і дата, а також ім’я студента, який просканував мітку.

Якщо вибрано режим реєстрації, то сервер теж спочатку перевіряє чи ID уже зареєстрований у системі і відправляє відповідне повідомлення. Якщо ж ідентифікатор у БД відсутній, то він записується час і дата зчитування і власне сам номер. Після цього його можна призначити будь-якому студенту.

Ще дану СКВ можна налаштувати для автоматичного запуску. Для цього у файлі boot.py потрібно прописати відповідні інструкції. Вони виконаються одразу після підключення контролера до живлення. Проте система сконструйована так, що в такому випадку система зможе запускатись лише в одному режимі, або реєстрації, або сканування. Для того щоб змінити режим потрібно буде перезаписувати boot.py. При умові коли один пристрій використовується адміністрацією для реєстрації нових міток, а решта використовуються для зчитування студентських міток, така реалізація є зручною. Це також зручно тим, що при потребі без складних зусиль можна змінити режим на будь-якому зчитувачі даної системи.

3.3. Розробка веб додатку

Створена СКВ передає і отримує дані із віддаленого сервера. Тому другою частиною створення системи була розробка API інтерфейсу, адмін панелі та сайту для користувача за допомогою Django. Оскільки поки що система не розроблялась для масштабного використання, то як БД було вибрано реляційну СУБД SQLite. Вона не потребує додаткового налаштування на сервері, поставляється за замовчуванням у Django. Її зручно

використовувати у невеликих проєктах і якщо не потрібно виконувати якісь складні SQL запити чи зберігати в бази величезні масиви інформації, її цілком достатньо.

Спочатку було описано схему БД у файлі models.py [7]. Для роботи системи достатньо чотирьох, пов'язаних між собою різним типом відношень, таблиць: Групи, Студенти, Просканові мітки та Мітки для реєстрації. З часом при розширенні системи можна додати таблиці із даними про персонал університету, розклад, аудиторії, факультети і інститути. Можливості Django дозволяють масштабувати системи. Наступним кроком є налаштування адмін панелі. В Django є готові інструменти для цього.

Взаємодію контролера і серверного додатка зручно проводити через API інтерфейс. Для його реалізації найчастіше використовується бібліотека Django REST Framework [7]. Для правильного функціонування треба налаштувати серіалізатор, який перетворює складні дані в типи даних Python. Доступ до даних можна отримати за двома адресами, залежно від того в якому режимі працює СКВ. У режимі сканування: https://tracker.senabo.site/api/scan/. У режимі реєстрації міток: https://tracker.senabo.site/api/register/. Так виглядає типовий API запит до сервера: **'{"body":{"tag":"ax673478", "student":"", "scanned":null}}'.** При зверненні до певної адреси виконується відповідний код у views.py, який розрізняє POST і GET запити. Відповідно до запиту повертаються певні дані із БД і відправляються на мікроконтролер. Приклад коду серіалізатора, опису моделей та відображення можна переглянути у додатку 2.

Для того щоб студенти чи викладачі могли переглядати статистику відвідування було створено просту веб сторінку, яка доступна за адресою https://tracker.senabo.site/. На ній можна переглянути загальну інформацію по групах і детальну інформацію про кожного студента.

Увесь Django додаток розгорнуто на VPS сервері https://www.linode.com/. На машині було встановлено Ubuntu 18.04, налаштовано Django для

функціонування у робочому середовищі, налаштовано Nginx і uWSGI для роботи у зв’язці, налаштовано DNS.

Висновки до розділу

У поточному розділі було послідовно описано створення системи контролю відвідуваності студентів. Було створено пристрій для зчитування міток, а також API інтерфейс, адмін панель і сайт. Усе це разом утворює одну систему.

Вона вийшла не дуже масштабною, проте повністю справляється зі своїм прямим призначенням. Зараз її можна застосовувати на рівні факультету чи інституту. Вона без проблем справиться із навантаженням. Якщо додати певний функціонал, і змінити СУБД, то дану систему можна використовувати на рівні університету. Потрібно лише розробити для неї додаткову інфраструктуру, таку як живлення пристроїв зчитування і доступ кожного пристрою до WIFI.

4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Після того, як прототип системи готовий, потрібно оцінити економічну доцільність виготовлення даної СКВ. За даними інтернет-магазину [17], який спеціалізується на продажі СКВ та СКД, найдешевший безконтактний зчитувач у готовому корпусі коштує 211 грн. Вартість готових контролерів із вмонтованим зчитувачем стартує від 1222 грн. В такому випадку уся інформація про користувачів зберігається у внутрішній пам'яті мікроконтролера. Недорогих систем із готовим веб додатком практично немає на ринку. Тому коли потрібно використати СКВ у місцях із невеликим навантаженням і щоб був доступний веб інтерфейс, доводиться переплачувати за непотрібні функції.

Представлена СКВ, в першу чергу розроблялась із метою зробити максимально дешевий пристрій, який буде мати усі необхідні функції для проведення контролю відвідування студентів із зручним веб додатком. Загальну вартість можна побачити в таблиці 4.1. Усі компоненти були замовлені з китайського інтернет магазину. Аналоги в Україні коштують майже у два рази дорожче. Можна суттєво зекономити, якщо закупити компоненти гуртом.

Загальна вартість склала близько 345 грн або 12.76 дол. До даної суми також буде потрібно додати витрати на корпус, який ще не спроєктований. Але також потрібно врахувати, що домен потрібно оплачувати лише раз в рік. До щомісячних витрат на утримання системи відноситься утримання сервера. Ці витрати також можна зменшити, якщо запустити систему на університетських серверах. Крім фінансових затрат слід врахувати затрати на проєктування, складання схеми, програмування. Було витрачено приблизно 50 годин, половина з яких на роботу із апаратною частиною інша половина на роботу із програмною.

Як бачимо вартість даної системи значно нижча ніж вартість представлених аналогів на ринку. Представлену СКВ можна вважати

економічно вигідною. Уже зараз за невелику ціну можна отримати базові потрібні функції. А без великих фінансових вкладень можна розширити функціонал і зробити повноцінну масштабну систему, котра нічим не буде поступатись аналогам на ринку і буде дешевшою в приблизно два-три рази.

Таблиця 4.1 — Усі фінансові витрати на розробку прототипа СКВ.

	Ціна $
NodeMCU v3	2.19
MFRC-522	1.28
OLED module	1.57
Провідники	0.85
Реєстрація домена (на рік)	1.87
VPS Linode (в місяць)	5
Сума	**12.76**

Висновки до розділу

В цьому розділі було проведено економічне обґрунтування розробленої СКВ. Було оцінено часові і фінансові затрати. Систему вигідно виготовляти і якщо її вдосконалити, то можна випускати на комерційній основі.

5. ОХОРОНА ПРАЦІ

Охорона життя і здоров'я людей в процесі їх трудової діяльності, забезпечення безпечних умов праці — одна з найбільш важливих завдань сучасності. Тому зростає актуальність розробки та впровадження нових способів нормалізації умов праці, забезпечення здоров'я працівників, чия діяльність тісно пов'язана з комп'ютером чи іншими пристроями, які працюють завдяки електроживленню.

Електричний струм може стати причиною важких нещасних випадків, велика частина яких відбувається через нехтування небезпекою, яку становить собою електричний струм. Колективні електротехнічні засоби захисту запобігають можливість потрапляння людини під напругу (ураженням електричним струмом), що можливо в разі пошкодження ізоляції електричного обладнання або зіткнення з обірваними проводами. Колективні електротехнічні засоби захисту — одне з найважливіших засобів забезпечення безпеки людей, які під час проведення робіт можуть випадково опинитися в небезпечній зоні.

Дія електричного струму на людський організм залежить від цілого ряду причин: від сили струму і його частоти, від часу проходження струму через тіло людини, від ділянки ураження, стану організму в момент удару та іншого.

Основним фактором, від якого залежить міра ураження людини, є сила струму. Для характеристики впливу електричного струму на людину встановлено три критерії: пороговий відчутний струм (найменше значення сили електричного струму, що викликає при проходженні через організм людини відчутні подразнення); пороговий невідпускальний струм (найменше значення сили електричного струму, що викликає судомні скорочення м'язів руки, в якій затиснутий провідник) і пороговий фібрилярний струм (найменше значення сили електричного струму, що викликає при проходженні через тіло людини фібриляцію серця).

Як відомо, сила струму в колі залежить від прикладеної напруги і від опору цього кола. Опір тіла людини залежить від ряду причин і перш за все від

стану шкіри в точках дотику до полюсів джерела струму, оскільки опір інших тканин людського тіла дуже малий в порівнянні з опором поверхневого шару шкіри. Величина опору тіла коливається в широких межах: від сотень Ом до сотень тисяч Ом.

Різке зменшення опору тіла відбувається в тому випадку, коли збільшується площа його зіткнення з предметами, які проводять струм, наприклад, при роботі з плоскогубцями або металевою викруткою, при торканні до металевих шасі або корпусів приладів або ж коли людина стоїть на сирій землі. Зі зниженням опору тіла небезпека ураження електричним струмом збільшується.

Небезпечною є будь-яка напруга, що перевищує значення в 50 В змінного і 120 В постійного струму. Але водночас небезпека ураження людини струмом визначається не тільки напругою, під яку вона потрапила, але і від умов, при яких відбувається дотик до провідника, і головним чином опором ланцюга, через яку пройшов струм.

Все сказане про небезпеку електричного струму відноситься як до постійного, так і до змінного струму промислової частоти (50 Гц). Зі збільшенням частоти струму спостерігається зменшення ступеня небезпеки. Струми високих частот (понад 10 000 Гц) вже не викликають дратівної дії і в цьому відношенні не становлять великої небезпеки для організму людини. Однак вважати ці струми зовсім безпечними не можна, тому, що при високих частотах проходження струму через тіло викликає дуже сильні, іноді смертельні, опіки.

Тяжкість ураження струмом значною мірою залежить від шляху його проходження через тіло людини. Найбільш небезпечні випадки, коли струм проходить через область серця, дихальних органів або через голову.

Чим довше проходить струм через тіло, тим сильніші його наслідки. При тривалому проходженні через тіло навіть слабкий струм може завдати організму людини важкі ушкодження. Тому при нещасних випадках дуже важливо швидко звільнити потерпілого від струму.

При експлуатації електрообладнання найчастіше застосовують захисне заземлення, занулення, відключення.

Під захисним заземленням розуміють навмисне з'єднання металевих частин електроустаткування (корпусів електродвигунів, електроапаратури тощо) з землею за допомогою заземлювачів і заземлювальних провідників з метою створення між корпусом пристрою, що захищається і землею досить малого опору.

Захисне занулення, так само як і захисне заземлення, призначене для усунення небезпеки ураження електричним струмом при замиканні на корпус електроустановок. Захисне занулення здійснюється приєднанням корпусу та інших конструктивних частин, які не проводять струм, електроустановок до неодноразово заземленого нульового проводу.

Захисне відключення є швидким захистом, що забезпечує автоматичне відключення електроустановки при виникненні в ній небезпеки ураження струмом. Така небезпека може виникнути при порушенні ізоляції частин, які проводять струм, і пробої на корпус, зниженні рівня ізоляції, дотику людини до частин, які проводять струм. Ефективність систем відключення визначається їх швидкодією, оскільки при малому часі впливу струму на людину його допустима величина може бути значно більшою.

Висновки до розділу

В цьому розділі розглянуто які ризики бувають при роботі з електропристроями. Розглянуто основні причини ураження електричним струмом та способи його уникнення.

ВИСНОВКИ

В дипломному проєкті розроблена система контролю відвідуваності студентів.

Проведено загальний огляд і опис роботи СКВ. Було описано принципи роботи технологій RFID та NFC, історію їх створення і розвитку. Розказано у яких сферах і як застосовують ці технології.

Серверна частина розроблена на Django. Зв’язок сервера із пристроєм реалізовано через API інтерфейс. Створено сайт адміністрування, де можна реєструвати нові мітки та студентів у системі, і веб-сторінку на якій виводиться інформація по відвідуванню.

Проаналізовано доступні компоненти, які підходять для створення системи. Для реалізації вибрано найбільш вдалі компоненти за функціональною і ціновою ознаками.

Для розробки програмного забезпечення вибрано мову Python і фреймворк Django.

Розроблено електричну схему з’єднання компонентів системи. Розроблено алгоритми і програми для керування апаратною частиною системи.

Обґрунтовано економічно доцільність виготовлення даної СКВ. Порівняно із ринковими аналогами. Рекомендовано, при деякому допрацюванні, впровадити систему в експлуатацію на рівні усього університету.

Описано способи та причини ураження електричним струмом при роботі із електропристроями. Наведено основні методи для запобігання ураженню струмом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Владимир Дронов. Django 2.1 Практика создания веб-сайтов на Python.— СПБ.: БХВ-Петербург, 2019. — 672 с.
2. Максим Власов. RFID: 1 технология – 1000 решений: Практические примеры использования RFID в различных областях. — М.: Альпина Паблишер, 2014. — 218 с.
3. Марко Шварц. Интернет вещей с ESP8266. — СПБ.: БХВ-Петербург, 2019. — 224 с.
4. Олексій Миколайович Васильєв. Програмування мовою Python. Навчальна книга - Богдан, 2018. — 504 с.
5. Тонкошкур О. С. Мікроконтролерні пристрої : навчальний посібник для студентів спеціальності «Мікро- та наноелектроніка» / Тонкошкур О., Гомілко І., Коваленко О. ; — Дніпропетровський нац. ун-т ім. О. Гончара. – Д. : Видавництво ДНУ, 2011. — 264 с.
6. Шарфельд Т. Системы RFID низкой стоимости. Москва: «Горячая линия Телеком», 2006. — 197 с.
7. Antonio Melé. Django 2 by Example: Build powerful and reliable Python web applications from scratch. Packt Publishing; 2nd Revised edition, 2018. — 526 с.
8. Nicholas H. Tollervey. Programming with MicroPython: Embedded Programming with Microcontrollers and Python. O'Reilly Media, 2017. — 214 с.
9. Django documentation [Електронний ресурс]: https://docs.djangoproject.com/en/3.0/
10. Django REST framework [Електронний ресурс]: https://www.django-rest-framework.org/
11. Driver ssd1306 [Електронний ресурс]: https://github.com/micropython/micropython/tree/master/drivers/display
12. MicroPython class to access the MFRC522 RFID reader [Електронний ресурс]: https://github.com/wendlers/micropython-mfrc522

13. MicroPython tutorial for ESP8266 [Електронний ресурс]: https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html

14. NFC от «А» до «Я» [Електронний ресурс]: https://securityrussia.com/blog/nfc.html

15. NodeMCU Documentation [Електронний ресурс]: https://nodemcu.readthedocs.io/en/master/

16. The history of RFID technology [Електронний ресурс]: https://medium.com/micro-tracking-macro-insights/the-history-of-rfid-technology-over-the-past-80-years-1f7f69dc0ccb

17. Інтернет магазин систем безпеки [Електронний ресурс]: https://security-shop.com.ua/

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ МІКРОКОНТРОЛЕРА

Модуль read.py :

```
def read(url):
    import mfrc522
    from oled import oled, show_loading
    import urequests
    # Init rfid reader
    rdr = mfrc522.MFRC522(14, 13, 12, 2, 15)
    try:
        while True:
            # Check antenna
            (stat, tag_type) = rdr.request(rdr.REQALL)

            # Show animation on esp display
            oled.fill(0)
            oled.show()
            show_loading(56, 8, 16, 16, 1)

            print(tag_type)

            if stat == rdr.OK:
                show_loading(56, 8, 16, 16, 0)
                # Get data about tag
                (stat, raw_uid) = rdr.anticoll()

                if stat == rdr.OK:
                    # get tag id
                    tag_id = "0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1],
raw_uid[2], raw_uid[3])
                    oled.fill(0)
                    oled.text('IDENTIFICATION', 0, 10)
                    oled.text('...', 0, 20)
                    oled.show()

                    data = '{"body":{"tag":"%s", "student":"", "scanned":null}}' %
str(tag_id)
                    headers = {'Content-Type': 'application/json'}
                    try:
                        # Share tag id to api
                        r = urequests.post(url, data=data, headers=headers)
                        oled.fill(0)
                        oled.text(str(r.json()).upper(), 0, 10)
                        oled.show()
                        print(r.json())
                    except:
```

```
                        oled.fill(0)
                        oled.text('ERROR. RETRY', 0, 10)
                        oled.show()
                        print('error post api')
                else:
                    oled.text('ERROR. RETRY', 0, 10)
                    oled.show()
                    print('error')

    except KeyboardInterrupt:
        print('Bye')
```

Модуль main.py :

```
def start(action):
    import read
    url_scan = 'https://tracker.senabo.site/api/scan/'
    url_register = 'https://tracker.senabo.site/api/register/'

    if action.lower() == 'scan':
        read.read(url_scan)
    elif action.lower() == 'register':
        read.read(url_register)
    else:
        print('Unknown action!')
```

ДОДАТОК Б

ЛІСТИНГ СЕРВЕРНОГО ДОДАТКУ

Опис Django моделей в файлі models.py:

```
from django.db import models

class Group(models.Model):
    group = models.CharField(verbose_name='група', max_length=120)

    def __str__(self):
        return self.group

    class Meta:
        verbose_name = 'Група'
        verbose_name_plural = 'Групи'

class Student(models.Model):
    name = models.CharField(verbose_name='ім\'я', max_length=250)
    group = models.ForeignKey(Group, verbose_name='група', on_delete=models.CASCADE)
    number_scan = models.IntegerField(verbose_name='кількість сканувань', null=True, blank=True)
    def __str__(self):
        return self.name

    class Meta:
        ordering = ('-name',)
        verbose_name = 'Студент'
        verbose_name_plural = 'Студенти'


class TagRegister(models.Model):
    tag = models.CharField(verbose_name='Мітка', max_length=120, unique=True)
    scanned = models.DateTimeField(verbose_name='Створено', auto_now_add=True)
    student = models.OneToOneField(Student, related_name='student_in_tag',verbose_name='Студент',null=True,blank=True, on_delete=models.SET_NULL)

    def __str__(self):
        return self.tag

    class Meta:
        ordering = ('-scanned',)
        verbose_name = 'Зареєстрована мітка'
        verbose_name_plural = 'Зареєстровані мітки'
```

```
class TagReader(models.Model):
    student=models.ForeignKey(Student, verbose_name='Студент',related_name = 'tags', on_delete= models.CASCADE)
    tag = models.CharField(verbose_name='Мітка', max_length=120)
    scanned = models.DateTimeField(verbose_name='Просканировано', auto_now_add=True)
    def __str__(self):
        return self.scanned.astimezone().strftime('%d-%m-%Y | %H:%M')

    class Meta:
        ordering = ('-scanned',)
        verbose_name = 'Мітка відсканована'
        verbose_name_plural = 'Відскановані мітки'
```

Код серіалізації serializers.py:

```
from rest_framework import serializers
from .models import TagReader, TagRegister
from django.db import IntegrityError


class TagScanSerializer(serializers.Serializer):
    tag = serializers.CharField()
    student = serializers.CharField(allow_blank=True)
    scanned = serializers.DateTimeField(allow_null=True)

    def create(self, validated_data):
        try:
            tag = validated_data['tag']
            student = TagRegister.objects.get(tag=tag).student
            print(student)
            res = TagReader.objects.create(tag=tag, student=student)
            return f'{student}'
        except TagRegister.DoesNotExist:
            return 'unknown tag'


class TagRegisterSerializer(serializers.Serializer):
    tag = serializers.CharField()
    student = serializers.CharField(allow_blank=True)

    def create(self, validated_data):
        try:
            res = TagRegister.objects.create(tag=validated_data['tag'])
            return f'{res} saved'
        except IntegrityError:
            return 'already in db'
        except:
            return 'unknown error'
```

Код відображення Django у файлі views.py:

```
from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.views import APIView
from .models import *
from .serializers import *
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger


class TagScan(APIView):
    def get(self, request):
        tags = TagReader.objects.all()
        serializer = TagScanSerializer(tags, many=True)
        return Response({'tags': serializer.data})

    def post(self, request):
        tag = request.data.get("body")
        serializer = TagScanSerializer(data=tag)
        if serializer.is_valid(raise_exception=True):
            tag_saved = serializer.save()
            print(tag_saved)
            return Response(tag_saved)
        return Response('error')


class TagRegistration(APIView):
    def get(self, request):
        tags = TagRegister.objects.all()
        serializer = TagRegisterSerializer(tags, many=True)
        return Response({'tags': serializer.data})

    def post(self, request):
        tag = request.data.get("body")
        serializer = TagRegisterSerializer(data=tag)
        if serializer.is_valid(raise_exception=True):
            tag_saved = serializer.save()
            print(tag_saved)
            return Response(tag_saved)
        return Response('error')


def index(request):
    students = Student.objects.order_by('-number_scan')
    groups = Group.objects.all()
    for s in students:
        s.number_scan = s.tags.all().count()
        s.save()

    context = {'students': students,
               'groups': groups,
               }
```

```
    return render(request, 'index.html', context)

def student_detail(request, pk, ):
    scans_object = TagReader.objects.filter(student=pk)
    if not scans_object.exists():
        name = Student.objects.get(pk=pk)
    else:
        name = scans_object.first().student
    show_all = request.GET.get('show_all')
    if show_all != "1":
        paginator = Paginator(scans_object, 12)
    else:
        paginator = Paginator(scans_object, scans_object.count())

    page = request.GET.get('page')
    try:
        scans = paginator.page(page)
    except PageNotAnInteger:
        scans = paginator.page(1)
    except EmptyPage:
        scans = paginator.page(paginator.num_pages)
    context = {'scans': scans,
                'name': name,
                'page': page,
                'show_all': show_all,
                }

    return render(request, 'student.html', context)
```