

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Розробка системи розумної полиці

Виконав: студент(ка) 4 курсу, групи КНТ-517

Спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерна інженерія»

Власенко Я. А.
(прізвище та ініціали)

Керівник Грушко С.С.
(прізвище та ініціали)

Рецензент Шило Г.М.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

« » 2021 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Власенку Ярославу Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка системи розумної полиці

керівник проекту (роботи) Грушко Світлана Сергіївна, к. т. н., доцент

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

затверджені наказом вищого навчального закладу від « 17 березня 2021 року № 81

2. Строк подання студентом проекту (роботи) 11 травня 2021 року

3. Вихідні дані до проекту (роботи) полиця, мережа Wi-Fi, хмарне сховище, робоча станція (ПК)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області

2) Дослідження кінцевих пристроїв інтернету речей

3) Розробка системи «розумна полиця»

4) Застосування і розвиток системи

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

Характеристика плати Raspberry Pi;

Архітектура системи;

Підключення тензодатчиків;

Робота з ioBroker

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	Грушко С.С., к. т. н., доцент		
Нормоконтроль	Щербак Н.В., ст. викл. каф. КСМ		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітки
1	Аналіз технічного завдання. Огляд сучасних рішень	01.03.2021 р.	
2	Розробка структури системи	15.03.2021 р.	
3	Розробка алгоритму роботи системи	20.03.2021 р.	
4	Реалізація проекту	01.04.2021 р.	
5	Дослідження роботи системи	10.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	15.04.2021 р.	
7	Оформлення графічного матеріалу	20.04.2021 р.	
8	Оформлення допоміжного матеріалу	01.05.2021 р.	

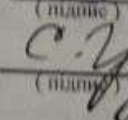
Студент


(підпис)

Я.А. Власенко

(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

С.С. Грушко

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 66 с., 34 рис., 4 табл., 2 додатки, 21 джерело.

РОЗУМНА ПОЛИЦЯ, РИТЕЙЛ, ТЕНЗОДАТЧИК, МІКРОКОНТРОЛЕР

Об'єкт розробки – система контролю товарних запасів в ритейлі.

Мета роботи – вивчення механізмів, технологій та розробка IoT- системи контролю товарних запасів в ритейлі.

В дипломній роботі проведено дослідження пристроїв і технологій, а саме які мікроконтролери і тензодатчики є і якими перевагами володіють ті чи інші пристрої, аналіз типу бездротових зв'язків і платформ IoT, щодо типів зв'язків і платформ, що підійдуть під таку задачу. В роботі наведено розробку своєї IoT- системи, а саме розробку IoT - системи контролю товарних запасів в ритейлі. Дана система - "розумна полиця", дозволяє управляти і стежити за динамікою товарів на полиці і визначати в яких полицях пора виставляти закінчені або відсутні товари, а також стежити за якістю роботи товарознавців.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

RFID	Radio Frequency IDentification
DDS	Data Distribution Service
CoAP	Constrained Application Protocol
MQTT	Message Queuing Transport
CWMP	CPE WAN Management Protocol
DNP3	Distributed Network Protocol
ОС	Операційна система

ЗМІСТ

Вступ	7
1 Аналіз технічного завдання	9
1.1 Проблеми в управлінні товарними запасами	9
1.2 Огляд рішень, що вже існують	9
1.2.1 Bizerba SmartShelf	9
1.2.2 Amazon Dash Smart Shelf	13
1.2.3 Розумна RFID полиця	14
1.2.4 Електронні цінники	17
2 Дослідження кінцевих пристроїв інтернету речей	21
2.1 Мікроконтролери	21
2.1.1 Мікроконтролери 8051	21
2.1.2 Мікроконтролери PIC	22
2.1.3 Мікроконтролери AVR	23
2.1.4 Мікроконтролери ARM	24
2.2 Тензодатчики	26
2.3 Технології бездротового зв'язку і хмарні IoT платформи	29
2.3.1 Технології бездротового зв'язку	29
2.3.2 Аналіз хмарних IoT платформ	30
2.3.3 Протоколи передачі даних Інтернету речей	34
3 Розробка системи «розумна полиця»	44
3.1 Загальна архітектура системи	44
3.2 Вибір і інтеграція кінцевих пристроїв і технологій передачі даних	44
3.3 Застосування і розвиток системи	56
Висновки	58
Перелік джерел посилання	59
Додаток А	61
Додаток Б	66

ПЕРЕЛІК ГРАФІЧНОГО МАТЕРІАЛУ

Пл1 - Характеристика плати Raspberry Pi;

Пл2 - Архітектура системи;

Пл3 - Підключення тензодатчиків;

Пл4 - Робота з ioBroker

ВСТУП

У нинішніх реаліях відбувається зміна економічних систем. Дане явище пов'язане з впливом діджиталізації і глобалізації економіки. При трансформації ми спостерігаємо зміну моделей бізнесів, появу різних нових напрямків діяльності, в зв'язку чим підвищується конкурентоспроможність галузей.

На сьогоднішній день економіка зазнає значну зміну і переходить на інший щабель розвитку. Пов'язане це з тим, що йде активне впровадження цифрових технологій, що говорить нам про настання четвертої промислової революції. В одному з економічних форумів в Женеві, Клаус Шваб зазначає, що "ми стоїмо біля витоків революції, яка фундаментально змінить те, як ми живемо, працюємо і спілкуємося один з одним" [1].

Аналітики вважають, що на сьогоднішній день в світі існує приблизно 15 мільярдів об'єктів, підключених до інтернету, на думку аналітиків в 2021 році в світі буде близько 50 мільярдів підключених до мережі пристроїв, а обсяг ринку досягне приблизно 1.5 трлн доларів [2]. Дані технології IoT в комплексі забезпечують можливість миттєвого взаємодії людини і машини. В кінцевому підсумку, люди, що знаходяться поруч з цими пристроями, пристрої реагують на їх присутність, рух і тд. Основою Інтернету Речей є датчики, підключені до глобальної мережі.

В даний час велику популярність мають "розумні помічники", які мають цілий ряд різних можливостей: реалізація електронних покупок, аналіз інформації про конкретного користувача [3]. Слід також звернути увагу на взаємодії штучного інтелекту і роботизації, які за допомогою датчиків вимірюють і контролюють зміни різних датчиків

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Проблеми в управлінні товарними запасами

Виявлено чотири ключових проблем в управлінні товарними запасами в ритейлі:

- складно виявити де мережа втрачає, так як відстежити і порахувати втрачені продажі по кожній товарній позиції в точці зберігання, без спеціальних систем є неможливим. Втрачені продажі виникають, коли покупець, приходячи в магазин не знаходить на полиці потрібного товару. Деякі магазини, щоб не втратити своїх клієнтів, розміщують ярлики, що "Товар очікується". Така ситуація може призвести до того, що клієнти можуть просто піти і купити потрібний їм товар у іншого виробника;
- високі витрати на управління і аналіз товарних запасів;
- централізація управління запасів товару (раніше близько 50% товарних замовлень здійснювали товарознавці на місцях);
- відсутність зручного інструменту, щоб відстежити причину надлишкових товарних запасів в магазинах і на центральному складі. Часто постачальники "виштовхують" товар в магазини, пропонуючи знижки і бонуси.

1.2 Огляд рішень, що вже існують

1.2.1 Bizerba SmartShelf

Буває так, що певні продукти користуються підвищеним попитом і швидко закінчуються, що призводить до їх дефіциту. Лояльність покупців знижується, в свою чергу магазин втрачає потенційний прибуток.

Виникає головне питання: як знайти золоту середину, яка дозволить отримати максимальний прибуток, залишити покупця задоволеним і при цьому не замовити у виробника надмірно багато продукції, щоб не довелося згодом її утилізувати? Поширеним рішенням даної проблеми є складний аналіз, який вимагає великих витрат часу і сил співробітників компанії, але не завжди він приносить очікуваний результат.

Ще одне рішення, яке останнім часом користується особливою увагою у ритейлерів, - технологія «розумної полиці».

Один з розробників, які представили «розумну полицю», - компанія Bizerba. Технологія цього виробника покликана полегшити проведення аналітичних робіт, роблячи аналіз не на довгострокову перспективу, а на необхідний відрізок часу і для конкретного магазину.

Саме рішення є ваговою платформою, яка розділена на сектори. Вона передає дані про масу кожного сектора на термінал з встановленим програмним забезпеченням, яке обробляє отримані дані і пересилає відповідального співробітника, після чого приймається рішення про необхідність поповнення сектора або здійснюються заходи, спрямовані на ліквідацію товару в разі, якщо продукт не користується популярністю.

Співробітникам магазину більше не потрібно постійно стежити за залишками товару, виходити в зал і контактувати з величезною кількістю покупців. У реаліях сьогоdnішнього дня, коли введені суворі правила для захисту свого здоров'я та оточуючих людей, немає необхідності контролювати наповнення продуктів на полицях, що дозволяє створити ту саму соціальну дистанцію, захистити співробітників від зайвого контакту і зменшити ризики зараження.

Для настройки розумної полиці необхідно лише задати початкове середнє значення товару, яке ми хочемо підтримувати на даній полиці постійно. У разі, якщо це значення опускається, «розумна полиця» сповіщає різні кінцеві системи або відповідального співробітника про необхідність поповнення продукції. Для

товарів власного виробництва замовлення відправляється безпосередньо на виробництво.

Якщо у товару закінчується термін придатності, «розумна полиця» повідомляє про це. Також вона сигналізує про зниження попиту на конкретний продукт, який необхідно реалізувати, щоб уникнути його списання. В такому випадку уповноважений співробітник магазину приймає рішення про акції та знижки на даний продукт. Bizerba SmartShelf самостійно стежить за товаром, економить час співробітників магазину, дозволяючи їм зосередитися на інших проблемах.

Системі можна призначити певні завдання. Наприклад: при наявності електронних цінників на полиці можна в автоматичному режимі знизити ціну на продукти за годину до закриття; відправити завдання співробітнику магазину зробити більше нарізок в вихідний і святковий день; приготувати більше салатів до певного часу.

Також можна призначити умова: обов'язкове виконання поставленого завдання або необов'язкове, якщо на полиці залишилося багато / мало продукту.

SmartShelf скоротить витрати часу персоналу на такі завдання, як аналіз і інвентаризація. «Розумна полиця» дозволить зменшити втрати від списань і збільшити продажі за рахунок своєчасно запропонованих спецпропозицій - знижок, акцій. При інтеграції з іншими рішеннями з'явиться можливість виробляти всебічний адресний аналіз попиту.

Реалізація та впровадження «розумної полиці» навіть у вже працюючому магазині не займе багато часу і сил. Внутрішнє ПО легко і швидко інтегрується зі сторонніми системами магазину без необхідності складних доробок.

Для впровадження «розумної полиці» вимог до ПО немає. Для того щоб полиця могла передавати автоматично дані, потрібна інтеграція з ERP-системою замовника. При такій інтеграції «розумна полиця», використовуючи заздалегідь закладені в ERP скрипти, може створити замовлення на поставку товару або виробництво продукції (в тому числі вказавши обсяг необхідного виробництва), а

також повідомити системі про проведення акції на товар (якщо налаштований даний сценарій в ERP).

Інтеграція з ERP також дозволить виробляти інвентаризацію, передаючи в реальному часі дані про залишок товару на полиці.

В ручному режимі «розумна полиця» просто передає дані на екран терміналу про фактичне наповнення продуктами. Даний екран може бути розташований в будь-якій точці магазину (виробництво, будь-якої цех і т.п.). У разі, якщо кількість товару опускається нижче заданого заздалегідь значення, на екрані з'являється сповіщення про необхідність поповнення полиці продукцією. Тільки в цьому випадку співробітникові буде необхідно виходити в зал.

При цьому вимог до стелажного обладнання немає. Система являє собою вагові датчики, які кріпляться до стелажу, фактично перетворюючи стелаж в вагову платформу. Далі відбувається калібрування датчиків і через ПО Bizerba поділ стелажу на зони, що відповідають одному певному товару. За допомогою алгоритмів, вбудованих в ПО Bizerba, магазин може зчитувати дані про вагу не всієї площі, а окремих зон (ділянок). Знаючи вагу одного продукту, з точністю до 100% можна розрахувати залишок на полиці і повідомити про нього. Точність залежить від постійного ваги продукту. У разі, якщо вага одного і того ж товару буде сильно відрізнятися, точність буде падати. Якщо відмінність між вагою однакових товарів буде мінімально (в діапазоні приблизно $\pm 10\%$), точність показань залишків буде завжди висока, так як в даному випадку відбудеться корегування похибки.

У перспективі ви зможете збільшити продажі і зменшити втрати від списань товару, в першу чергу з обмеженим терміном придатності.

Рішення дозволить повністю оптимізувати процес наповнення полиць магазину, при цьому скоротивши витрати на персонал.

Вирішивши проблему списань, знизивши їх або, в ідеалі, виключивши, компанія заробить позитивний імідж, вказуючи на боротьбу зі збільшенням харчових відходів. Адже щороку на планеті утворюється близько 1,3 мільярда тонн

харчових відходів. До третини всіх вироблених продуктів харчування йде на утилізацію, не знаходячи свого покупця.

1.2.2 Amazon Dash Smart Shelf

Американський інтернет-гігант представив свій новий винахід - розумну полицю Amazon Dash Smart Shelf. Її основне завдання - допомогти малим підприємствам автоматично поповнювати запас канцелярського приладдя в міру необхідності.

Пристрій являє собою полицю для канцелярії з вбудованими датчиками. Коли вони фіксують недолік того чи іншого витратного матеріалу, то автоматично відправляють замовлення в інтернет-магазин. Наприклад, якщо полиця для блокнотів стала легше, смарт-пристрій зрозуміє, що продукт закінчується і замовить ще.

Amazon буде пропонувати розумні полиці в декількох розмірах. Вони будуть працювати від розетки і підключатися до Wi-Fi.

Щоб мотивувати клієнтів замовляти свій новий винахід, Amazon запропонував їм знижку в 15% на деякі бренди.

Ідея автоматичного замовлення витратних матеріалів не нова для Amazon. У 2015 році компанія випустила кнопки Dash, призначені для домашнього використання. Кожна кнопка відповідала за замовлення якогось певного товару. Наприклад, її можна було прикріпити на пральну машину і поповнювати резерви порошку одним натисканням.

Однак через чотири роки після запуску продукту в компанії зрозуміли, що клієнти швидше оформлять підписку на потрібний товар, ніж куплять таку кнопку. Тому гаджет був знятий з продажу. Передбачається, що компанія працює над тим, щоб вбудувати функцію замовлення витратних матеріалів всередину самих приладів. Наприклад, щоб принтер міг самостійно замовити чорнило, коли ті будуть під кінець.

1.2.3 Розумна RFID полиця

Розумна RFID полиця - це система складається з RFID зчитувача і RFID антен, які є безпосередньо полицями стелажів, шаф або подіумів в магазинах, складах або архівах. Розумна RFID полиця призначена для ідентифікації об'єктів розміщених на ній (магазин, склад або архів) і передачі інформації в реальному часі на сервер системи.

Використання розумної RFID полиці дає можливість для аналізу і отримання точних даних про кількість об'єктів і їх місцезнаходження. Це дозволяє своєчасно поповнювати запаси необхідної продукції, швидше проводити інвентаризацію, а так само значно спрощує пошук необхідного товару і виключає можливість пересортиці. Крім того розумна RFID полиця може виконувати антикражних функцію.

Розумні RFID полиці знайшли своє застосування в роздрібних магазинах, таких як продуктові магазини, магазини одягу, ювелірні магазини, книжкові магазини, аптеках та фармацевтичних магазинах. На складах з різною готовою продукцією або на складах з комплектуючими. В юридичних, судових або бухгалтерських архівах. У приватних бібліотеках, а так само в якості бюджетної системи повернення книг в публічних бібліотеках.

RFID являє собою метод автоматичної ідентифікації через радіосигнал. Система складається з зчитувачів, міток і програмного забезпечення. Мітка - це мікросхема, в якій зберігаються дані, а також антена для бездротової передачі інформації. Зовнішній зчитувач сканує пам'ять мітки радіочастотної ідентифікації і обробляє отримані дані. Програмне забезпечення відповідає за цілісну роботу системи.

Спосіб автоматичної ідентифікації об'єктів з кожним роком використовується все частіше. За допомогою Radio Frequency IDentification проводять звірку наявності майна організації та стану фінансових зобов'язань, автоматизують виробничі процеси, встановлюють справжність об'єктів,

відстежують логістику поставок. Постараємося розібратися детальніше, як працює система і в чому її перевага.

Radio Frequency IDentification по суті є вдосконаленим алгоритмом радіолокаційного розпізнавання «Свій-чужий». Апаратно-програмний технічний комплекс для автоматичного пошуку відмінностей використовує радіохвилі певної частоти. Дані зберігаються в спеціальній мітці, яку можна приймати, а потім дешифрувати віддалено за допомогою спеціальних зчитувачів.

Технологія має досить простий механізм роботи:

- інформація записується за допомогою радіохвиль на мікрочіп;
- дані надходять на зчитувач за допомогою радіосигналу вбудованої антени;
- визначення випромінюваної частоти, настройка і зчитування даних здійснюється автоматично за рахунок сканера.

Попит на Мірчі з'явився ще на початку двадцять першого століття. Система постійно вдосконалюється, використовується для маркування та автоматизованої ідентифікації продукції. Але перелік функцій RFID набагато ширше. За допомогою сучасної технології можна виконати ідентифікацію та контролювати рух маркованих об'єктів - як живих, так і неживих.

Сфер використання системи стає все більше. Ідентифікація особливо затребувана при проведенні інтелектуального обліку переміщення об'єктів, прийнятті управлінських рішень, автоматизації роботи.

У промисловому виробництві технологія використовується в наступних цілях:

- маркування ДП;
- автоматизація управлінських процесів;
- маркування упаковки та транспортних засобів;
- маркування закупівлі і доставки матеріалів;
- координація роботи, доступу і безпеки працівників;
- удосконалення процесу приймання і вивантаження;

- автоматизація междошового планування.

RFID в складському обліку:

- ідентифікація місць зберігання предметів;
- маркування процесів складування, доставки та переміщення об'єктів;
- координування руху ТЗ;
- управління безпекою і доступом до території.

У частині рітейлу сучасна технологія використовується для досягнення таких завдань:

- облік прибуткових і видаткових операцій, доставка товару замовникам;
- автоматизація перевірки наявності майна організації та стану її фінансових зобов'язань, інших процесів обліку;
- боротьба з крадіжками і підробками;
- організація касового місця, торгових залів, дисконтних програм.

Автоматизація платних доріг включає кілька напрямків:

- забезпечення захисту від порушень;
- організація платного проїзду;
- відстеження роботи пристроїв;
- можливість визначення ТС на високій швидкості руху;
- забезпечення автоматичної роботи пунктів оплати за проїзд.

RFID-технології затребувані також і при контролі доступу:

- забезпечують індивідуальний доступ працівників в приміщення;
- запобігають розкрадання і небажані проникнення;
- координують автоматичну роботу паркінгу;
- налагоджують дистанційне керування шлагбаумами, дверима.

У тваринництві сучасна система ідентифікація застосовується, коли знадобляться:

- саморегулюючі процеси годування;
- маркування поголів'я і подальшого контролю стану тварин;

- автоматичне забезпечення умов збереження життя;
- контроль показників вагового приросту і надойв;
- запобігання проникнення в будівлю сторонніх осіб.

Автоматична ідентифікація платіжних сервісів спрямована на реалізацію наступних цілей:

- оплата пропуску через механізми, що обмежують вхід на територію приміщення;
- контроль справності пристроїв;
- оплата проїзду в метрополітені або транспорті;
- оплата за відвідування аквапарків, атракціонів.

1.2.4 Електронні цінники

Електронні цінники (ЕЦ) - сучасні електронні пристрої, що дозволяють легко міняти ціни на полицях і використовувати динамічне ціноутворення. Вони стали альтернативою застарілих паперових цінників. Системи електронних цінників - комплекс з електронних цінників, програмного забезпечення і додаткового устаткування, необхідний для автоматичного оновлення цін.

Є два типи електронних цінників, такі як: сегментні (рис.1.1) і графічні (рис.1.2).



Рисунок 1.1 - Сегментний цінник

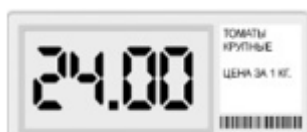


Рисунок 1.2 - Графічний цінник

Електронні цінники на розумній полиці вже не в новинку. Розумні полиці зараз займають простір магазинів. Невід'ємною частиною розумних полиць стали RFID (рис. 1.3) - мітки і софт, які в режимі реального часу дозволяють виявити інтереси клієнтів. Таким чином магазини можуть дізнатися, які товари найбільше розглядалися і купувалися, а які поверталися назад на полиці. Також інтелектуальні полиці вказують актуальну вартість товарів без потреби переписування цінника. Це дозволяє робити різні акції і надавати знижки не тільки в широкому часовому діапазоні (тиждень, місяць), але також в конкретні дні або навіть в конкретні години.



Рисунок 1.3 - Графічний цінник

У RFID-міток багато переваг, але також і є недоліки даної технології.

RFID технологія має наступні переваги:

- висока фізична надійність засобів ідентифікації, підвищена надійність системи в цілому за рахунок відсутності механічного зносу і децентралізації обробки інформації;
- можливість таємного розташування ідентифікатора, в тому числі його вбудовування в об'єкт (наприклад, всередину пластмасової деталі при її литві);
- висока незалежність від умов експлуатації (температура, гази, пил, бруд, мастило, фарба, дим, вібрації, вода, світло, механічні вібрації, електричні шуми і т.п.);
- RFID може використовуватися навіть в агресивних середовищах, а RFID- мітки можуть читатися через бруд, фарбу, пар, воду, пластмасу, деревину;

- простота користування та обслуговування;
- системна гнучкість, легкість інтеграції в будь-які системи підприємства.

Недоліки технології RFID:

- неможливість розміщення під металевими і електропровідними поверхнями (електромагнітне поле екранізується струмопровідними поверхнями);
- схильність до перешкод у вигляді електромагнітних полів. Системи радіочастотної ідентифікації можуть бути чутливі до перешкод у вигляді електромагнітних полів від включених комп'ютерів (моніторів);
- вплив на здоров'я людини.

Мережі Wi-Fi встановлюються в торгових центрах і магазинах, щоб підвищити лояльність покупців, але багато і не знають, що за допомогою Wi-Fi-аналітики можна виявляти і контролювати "місця тяжіння" покупців, перспективні і неперспективні торгові зони та багато іншого [4].

Від відвідувача в даному випадку не потрібно ніяких дій, досить мати з собою пристрій з включеним Wi-Fi. Система аналітики буде збирати дані і за цими даними в режимі онлайн формується підсумкова статистика, наприклад, в які магазини частіше заходять люди, в яких зонах проводять більше часу.

7Fresh - високотехнологічний супермаркет (Китай), в якому представлені "розумні кошики для покупок", які супроводжують покупця (кошика не потрібно штовхати), і руки клієнтів в процесі вибору завжди залишаються вільними.

X5 Retail Group (фуд-рітейл) тестує технологію відеорозпізнавання осіб для зменшення довжини черг на касах (відправляється сигнал про необхідність додаткової каси) і оптимізації торгового простору.

Walmart працює над "електронним керуючим", який буде підключений до датчиків на касах, вході і аналітиці по чеках. "Менеджер" буде роздавати типові завдання співробітниками магазину, контролювати швидкість виконання цих доручень і фіксувати продуктивність кожного працівника [5].

Одна з найбільших роздрібних торгових мереж Walmart представив власну розробку на базі технологій розпізнавання осіб. Система здатна визначити настрій покупця в процесі шопінгу, а також задоволеність клієнта після відвідування супермаркету.

Таким чином, якщо система знайде покупця з незадоволеним обличчям, співробітник магазину отримає повідомлення. У Walmart вважають, що така інновація допоможе розумним полицям підвищити якість обслуговування клієнтів.

1.2 Архітектура IoT рішень

Архітектура IoT-технологій складається в основному з 4 складових.

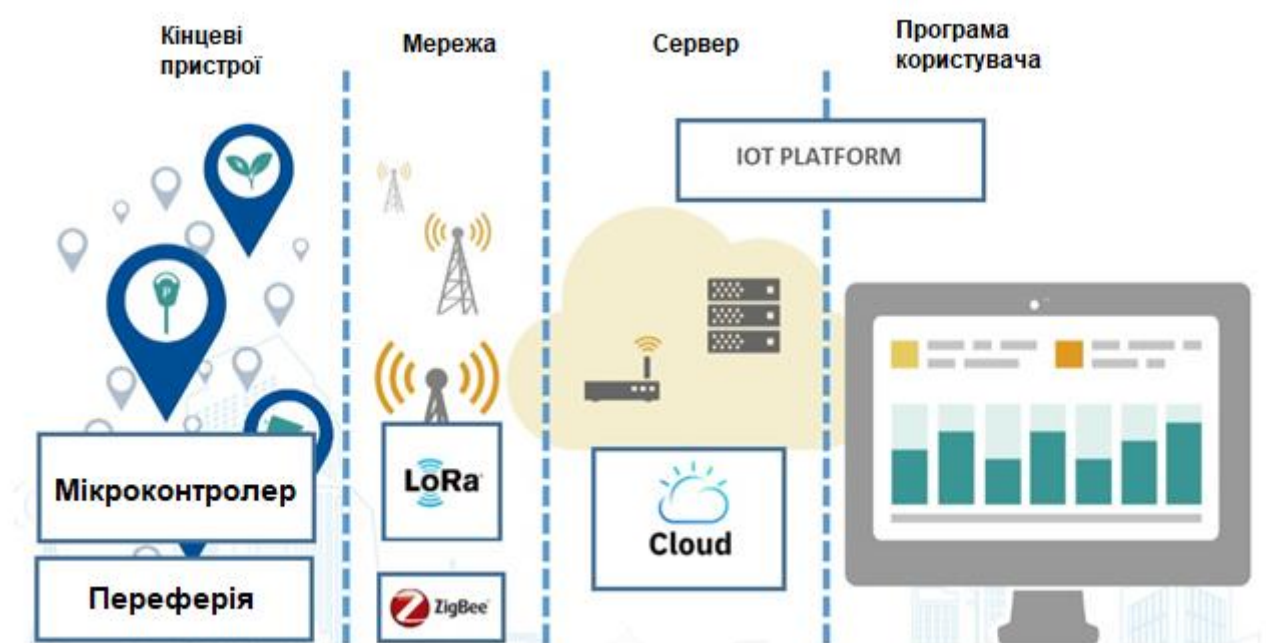


Рисунок 1.4 - Архітектура IoT-технологій

На самому початку знаходяться кінцеві пристрої. Кінцевим пристроєм може бути, наприклад, різні датчики, такі як: датчик температури, вологості, датчик протікання, аксіометр і тд.

Після того, як ми зібрали дані з кінцевих пристроїв, у нас з'являється необхідність кудись передати ці дані. В даному етапі нам допоможе мережа. Існує різні види мереж, вибір їх залежить від поставлених завдань, наприклад, ZigBee, LTE-M, Wi-fi, Bluetooth, NB-IoT, LoRa і тд.

Наступним етапом є передача даних на сервер. Сервер може бути фізичним, наприклад, вдома стоїть фізичний пристрій, який спілкується з іншими пристроями, але це може бути і хмарним сервером, який надає послугу і місце за певну плату. Після цього все потрапляє в програму користувача, тобто якийсь зручний інтерфейс, щоб дивитися і будувати графіки, і в цілому управляти пристроєм.

І на стику між сервером і інтерфейсом користувача є платформа, IoT платформа, яка включає в себе, як сервер, так і інтерфейс для користувача.

2 ДОСЛІДЖЕННЯ КІНЦЕВИХ ПРИСТРОЇВ ІНТЕРНЕТУ РЕЧЕЙ

2.1 Мікроконтролери

На ринку сьогодні представлена велика кількість мікроконтролерів. Розглянемо різницю між основними сімействами мікроконтролерів: AVR, ARM, 8051 і PIC.

2.1.1 Мікроконтролери 8051

Даний мікроконтролер є 8-бітовим сімейством мікроконтролерів, розроблений Інтел ще в 1981 році (рис. 2.1). Цей мікроконтролер є одним з популярних з сімейств мікроконтролерів, що використовуються у всьому світі. Мікроконтролер 8051 спочатку називався "системою на кристалі", оскільки дана

модель має 128 байт оперативної пам'яті. Даний процесор може обробляти до 8 біт даних одночасно.



Рисунок 2.1 - Мікроконтролер 8051

Дані мікроконтролери використовуються в великих кількостях у пристроях, в основному тому, що їх легко інтегрувати в проект.

Вони широко застосовуються в автомобільних рішеннях, широко застосовуються в гібридних транспортах для обробки даних отриманих з двигунів і управління ними.

Також даний мікроконтролер застосовується і в медичних пристроях. Наприклад, в таких, як моніторинг глюкози і вимірі артеріального тиску, мікроконтролери використовуються для відображення даних, що забезпечує високу надійність.

2.1.2 Мікроконтролери PIC

Ця серія мікроконтролерів, випущена компанією Microchip (рис. 2.2). Мікроконтролер PIC швидше і простіше реалізує програми, якщо порівнювати з іншими мікроконтролерами, такими як 8051. Простота програмування і простота взаємодії з іншими периферійними пристроями робить даний мікроконтролер більш успішним.



Рисунок 2.2 - Мікроконтролер PIC

Цей мікроконтролер також підтримує протоколи, такі як UART, CAN, SPI для використання з іншими периферійними пристроями. Даний мікроконтролер більшою мірою використовує модифіковану гарвардську архітектуру і також підтримує RISC. Завдяки цій архітектурі, PIC істотно швидше, ніж мікроконтролери, засновані на ядрі 8051, які побудовані на Фоннеймановській архітектурі.

2.1.3 Мікроконтролери AVR

Перші мікроконтролери типу AVR були випущені в 1996 році компанією Атмел (рис. 2.3). Розроблено AVR було Альф-Егіл Богеном і Вегара Волланом, тому даний мікроконтролер отримав дві перші букви від імен розробників.



Рисунок 2.3 - Мікроконтролер AVR

Даний мікроконтролер доступний в трьох основних підсімействах.

Тип TinyAVR має невеликий розмір, меншим кількості пам'яті, що робить його придатним тільки для простих додатків [6].

Тип MegaAVR, є популярним мікроконтролером, через великий обсяг пам'яті до 256 КБ, більшу кількість вбудованих додаткових пристроїв і даний тип підходить для досить складних додатків.

І останній тип XmegaAVR, цей мікроконтролер використовується в комерційних додатках для рішень в таких завданнях, де потрібна максимальна швидкість і великий обсяг пам'яті.

2.1.4 Мікроконтролери ARM

Дані мікроконтролери теж є одним з сімейств, побудованих на архітектурі RISC, що випускає компанією Advanced RISC Machines.



Рисунок 2.4 - Мікроконтролер ARM

Мікроконтролери даного типу засновані на 32-bit і 64-bit багатоядерних процесорах RISC. Даний процесор призначений для виконання невеликих кількостей інструкцій, щоб робота проводилася з більшою швидкістю, виконуючи інструкції в секунду.

Приклади використання процесора ARM: смартфони, планшети та інші різні мобільні пристрої. У зв'язку з невеликим набором команд їм потрібно менше транзисторів. Дані процесори з невеликими розмірами полегшують складність проектування і зменшують електроспоживання, що отримує популярність у використанні мініатюрних пристроїв.

Таблиця 2.1 - Основні відмінності між мікроконтролерами

	8051	PIC	AVR	ARM
Розрядність	8 біт	8/16/32 біт	8/32 біт	32 біт, іноді 64 біт
Інтерфейси	UART, USART, SPI, I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, іноді CAN, USB, Ethernet	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI, IrDA
Швидкість	12 тактів на інструкцію	4 такти на інструкцію	1 такт на інструкцію	1 такт на інструкцію
Пам'ять	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
Шинна архітектура	CLSC	частково RISC	RISC	RISC
Архітектура пам'яті	фон-неймановська	Гарвардська	модифікована	модифікована гарвардська
Енерго-споживання	середнє	низьке	низьке	низьке
Сімейства	варіації 8051	PIC16, PIC17, PIC18, PIC24, PIC32	Tiny, Atmega, Xmega, спец. AVR	ARMv4,5,6,7 ...
Виробники	NXP, Atmel, Silicon Labs, Dallas, Cypress, Infineon ...	Microchip	Atmel (Microchip)	Apple, Nvidia, Qualcomm, Samsung Electronics, TI ...
Вартість	низька	Середня	Середня	низька
Популярні мікроконтролери	AT89C51, P89v51	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32; варіації для Arduino	LPC2148, ARM Cortex-M0, ARM Cortex-M3, ARM Cortex-M7

2.2 Тензодатчики

Тензометричний датчик - це пов'язана з корпусом вагової платформи металева конструкція, яка є основним силовимірювальним елементом і використовується з будь-якими видами ваг. Принцип роботи приладу полягає в врівноваженні механічної сили датчика і маси певного вантажу з подальшим перетворенням її в пропорційний електричний сигнал. Фіксуючи тип і ступень деформації, тензодатчик ваги систематизує і автоматизує процес обробки показників з точністю до 0,03-0,25% [7].

Існує різні види тензодатчиків ваги, застосування яких залежить від типу вантажоприймальної платформи.

Консольні (балкові). Датчики даного типу мають низький профіль і характеризуються відсутністю чутливості до зсувів ліній дії сили (рис. 2.5).



Рисунок 2.5 - Консольні (балкові)

Конструктивне виконання тензодатчиків дозволяє використовувати їх у всіх ваговимірювальних системах з НПВ (найбільшою межею зважування) до 7 тонн. Наприклад, в якості чутливого елемента їх встановлюють на конвеєрних і платформних вагах.

S-образні. В основі функціонування датчиків даного типу лежить сила розтягування-стиснення по осі симетрії (рис. 2.6).



Рисунок 2.6 - S-образний датчик

Зменшення часу на установку і запуск обладнання відбувається за рахунок входять до комплект шарнірних підвісів. S-образні датчики застосовуються при виготовленні підвісних бункерних, кранових і резервуарних ваг, тестових систем і розривних машин.

Колонні ("бочка"). Датчики цього типу, маючи в основі сферичні опорні поверхні, працюють на стиск і самостійно приймають вихідне горизонтальне положення (рис. 2.7).



Рисунок 2.7 - Колонні ("бочка")

Із застосуванням колонних датчиків виготовляються вагонні, автомобільні та бункерні багатотонні ваги.

Циліндричні ("шайба"). Даний вид датчиків, верхня частина яких має форму сфери, працює за принципом «бочки», але має менші габарити і ступінь кочення, що виключає передачу моменту, що вигинає (рис. 2.8).



Рисунок 2.8 -Циліндричні ("шайба")

Тензодатчики типу «шайба» застосовуються для вимірювання навантажень до 100 тонн і використовуються при виготовленні та модернізації вагонних, бункерних, автомобільних і платформних ваг.

Полумостовий. Датчик виконаний з алюмінію, має форму квадрата, в центрі якого виступає частина з тензорезистером (рис. 2.9).



Рисунок 2.9 - Тензодатчик полумостовий

Тензодатчик даного типу застосовується для вимірювання навантажень до 50 кг.

2.3 Технології бездротового зв'язку і хмарні IoT платформи

2.3.1 Технології бездротового зв'язку

Комунікація є однією з найважливіших частин будь-якого проекту IoT. Хоча існує безліч протоколів зв'язку, кожному з них не вистачає тих чи інших характеристик, що робить їх "не зовсім підходящими" для додатків IoT. Основними проблемами є енергоспоживання, радіус покриття і пропускна здатність. Більшість комунікаційних радіотехнологій, таких як Zigbee, BLE, WiFi та інші, мають малу дальність дії, а інші, такі як 3G і LTE, споживають багато енергії, і дальність дії їх роботи не може бути гарантована, особливо в країнах, що розвиваються. Незважаючи на те, що ці протоколи і режими зв'язку працюють для певних проєктів, вони мають велике обмеження, наприклад, труднощі в розгортанні рішень IoT в областях без стільникового зв'язку (GPRS, EDGE, 3G, LTE / 4G) і необхідність в придбанні дорогих ліцензій.

Таблиця 2.2 - Основні відмінності між технологіями бездротового зв'язку

Технологія	LoRaWan	LTE-M	NB-IoT	NFC
Стандарт зв'язку	власний	3GPP	3GPP	ISO 13157
Частота	433 МГц, 868 МГц, 915 МГц.	700 МГц-2,2 ГГц. 452,5-467,5 МГц.	700 МГц-2,2 ГГц, 452,5-467,5 МГц.	13,56 МГц.
Швидкість передачі даних	RX 290 біт / с TX 50 Кбіт / с	1 Мбіт / с	~ 200 Кбіт / с	424 Кбіт / с
Топологія мережі	зірка	зірка	зірка	P2P

Продовження таблиці 2.2

Кількість учасників мережі	Дуже велике	Дуже велике	Дуже велике	2
Дальність	5-15 км	5 км	5 км	1-10 см
Потужність	Середня	висока	висока	низька
Спосіб застосування	Мобільний / локальний	Мобільний / локальний	Мобільний / локальний	локальний

Таблиця 2.3 - Основні відмінності між технологіями бездротового зв'язку

Технологія	BLE	Wi-Fi	Z-Wave	ZigBe
Стандарт зв'язку	Bluetooth SIG	802.1	власний	IEEE 802.1
Частота	2,4 ГГц.	2,4 / 5 ГГц.	868-926 МГц.	2,4
Швидкість передачі даних	125 Кбіт / с - 2Мбіт / с	до 150 Мбіт / с	100 Кбіт / с	250 Кбіт / с
топологія мережі	P2P, broad	зірка	Mesh	Mesh
Кількість учасників мережі	20	100	232	250+
дальність	40-1000м	40-100м	40-100м	40-100
потужність	низька	Середня	Середня	Середня
Спосіб застосування	локальний	локальний	локальний	локальний

2.3.2 Аналіз хмарних IoT платформ

IoT платформа - програмне забезпечення, призначене для підключення інтернет речей (датчиків, контролерів та інших пристроїв) до хмари і віддаленого доступу до них.

Являє собою проміжний рівень між апаратним рівнем (рівнем сенсорів) і прикладним.

Існує безліч платформ для підключення інтернет речей, але в даній роботі наводиться тільки кілька популярних платформ, а саме Open Source платформи.

Open Source - в буквальному перекладі означає "відкритий код". Це означає, що розробники не приховують свій програмний код, і його може модернізувати або використовувати для своїх цілей абсолютно будь-який бажаючий.

Home Assistant - open source система автоматизації, написана на мові Python3. Вона дуже активно розвивається, виходять постійні оновлення та з'являється підтримка дуже великої кількості нових пристроїв. Також вона досить простими способами інтегрується з більшістю популярних екосистем і в ній здійснюється підтримка популярних онлайн-сервісів.



Рисунок 2.9 - Home Assistant

Дану систему можна встановити, як на практично будь-який, досить продуктивний одноплатний ПК, так і на будь-який повноцінний комп'ютер.

Що стосується особливостей системи, то одна з найголовніших полягає в тому, що система має два варіанти - простий Home Assistant і Hass.io. Різниця між ними в тому, що в Hass.io є розділ з аддонами, які можна встановити прямо з інтернету інтерфейсу. У простому Home Assistant такої можливості немає.

Homebridge - сервер, написаний на Nodejs з використанням API Apple, який дозволяє додати пристрої в ваш будинок, які не мають сертифікацію і з самого початку не підтримують інтеграцію в Apple HomeKit.



Рисунок 2.10 - Homebridge

Завдяки злагодженій роботі розробників, на сьогоднішній день, в систему інтегровано дуже велика кількість пристроїв. Homebridge є кросплатформним рішенням, яке можна встановити на Windows, Linux і Mac OS X.

MajorDoMo - система автоматизації, написана на мові PHP. Система швидко розвивається і так само, як і всі системи, є OpenSource проектом. Має свою армію фанатів. Але щоб налаштувати дану систему ідеально під себе, потрібні чималі знання мови PHP.

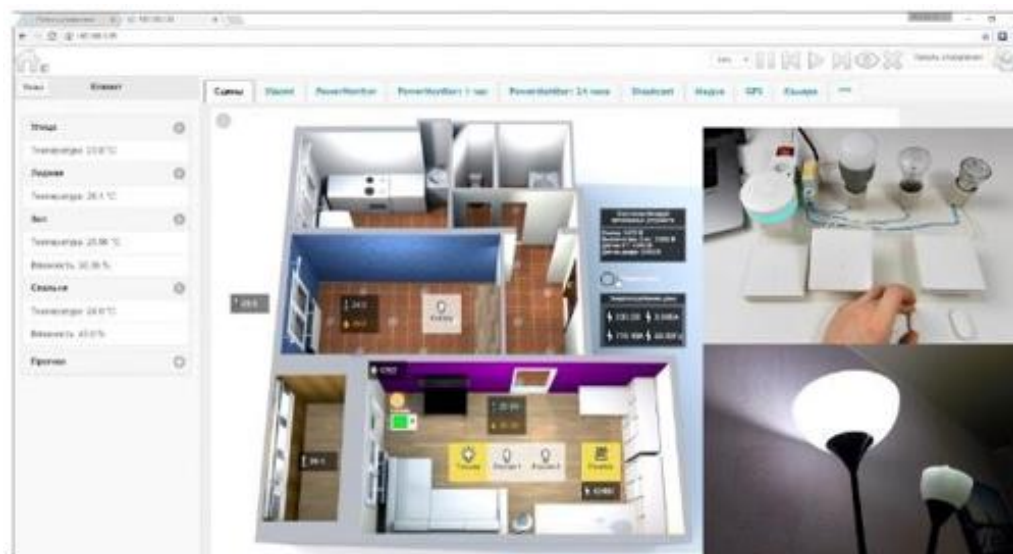


Рисунок 2.11 - MajorDoMo

Хоча є підтримка модульної структури і велика кількість готових модулів. До того ж, на відміну від більшості продуктів, мажордомом має дуже потужну команду розробників.

IoBroker - ще одна система для побудови розумного будинку, написана на NodeJS як і Homebridge.

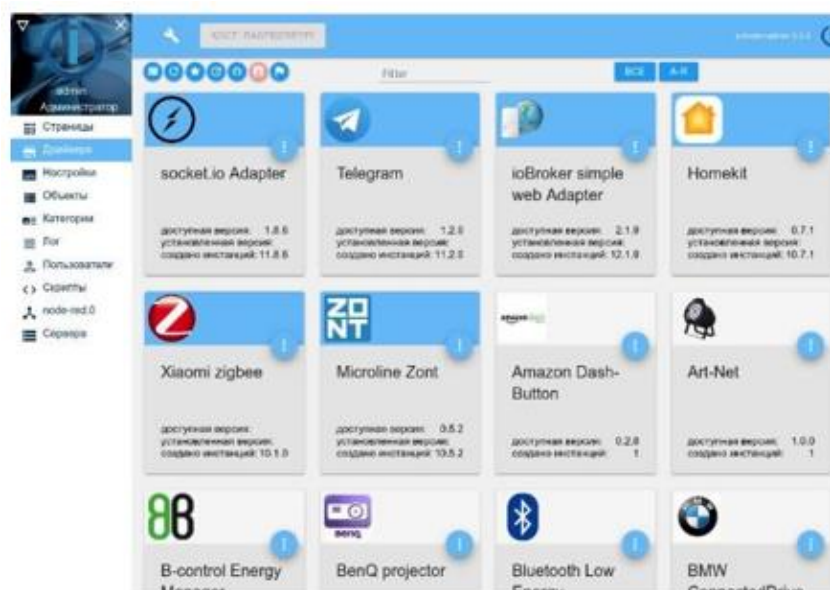


Рисунок 2.12 - IoBroker

Система теж дуже добре розвивається і має велику кількість інтегрованих пристроїв. Має армію своїх фанатів, але інтерфейс програми залишає бажати кращого.

Хоча деякі учасники, що для високонавантажених систем з великою кількістю пристроїв, ioBroker - це єдине відповідне рішення. Це, звичайно, може ставитися до промислової автоматизації, але факт наявності подібної думки означає, що систему використовують навіть в таких цілях.

2.3.3 Протоколи передачі даних Інтернету речей

Протоколи передачі даних IoT - це визначальні способи обміну даними між пристроями в мережі IoT.

Так заміною в мережі IoT популярного протоколу HTTP є протокол CoAP, службовець для використання пристроями з малою швидкістю, великими втратами. Протокол MQTT (Message Queuing Transport) був представлений в 1999 році, але даний протокол отримав популярність лише в 2010 році. Даний протокол спеціалізується на низькошвидкісних середовищах з великою затримкою, в зв'язку з чим підходить для спілкування між машинами.

Крім того, нерідко розробники використовують свої пропрієтарні протоколи, однак використання саме стандартних протоколів на IoT - платформах значно прискорюють впровадження і розробку систем і додатків IoT. Через пару років, а саме в 2002 році була сформована робоча група XMPP, яка потребувала для створення протоколу швидкого обміну повідомленнями. Дана група сформулировала чотири специфікації, які були висунуті в якості стандартів в 2004 році.

У загальному випадку протоколи ділять на групи в залежності від ділянки мережі, на якому вони використовуються. Так в рамках даної концепції Інтернету речей існують такі види:

Сенсорний вузол - сервер (CoAP, MQTT, XMPP, STOMP);

Сенсорний вузол - сенсорний вузол (найпопулярніший протокол DDS);
Сервер - сервер (AMQP).

Загалом існує безліч протоколів передачі даних, для прикладу наведено кілька з них.

- DDS (Data Distribution Service) - даний протокол використовує тип публікації - підписки для прийому і відправки даних, подій і команд серед кінцевих вузлів. Вузли формують інформацію, по-іншому - топик, (наприклад, температура, тиск і тд.) і відправляють шаблони. Вузлам, які зацікавилися в даних розділах, даний протокол доставляє створені шаблони. Транспорт є - UDP.

- CoAP (Constrained Application Protocol) - даний протокол досить сильно схожий на протокол HTTP, але різниться невеликим розміром заголовків, що робить зручним для мереж з обмеженими можливостями. Архітектура даного протоколу є клієнт-сервер і підходить для передачі інформації про стан вузла на сервері (повідомлення GET, PUT, HEAD, POST, DELETE, CONNECT). В якості транспорту - UDP.

- XMPP (Extensible Messaging and Presence Protocol) - даний протокол вже давно використовується в мережі Інтернет для передачі даних в режимі реального часу. Цей протокол працює на основі архітектур видавець-передплатник і клієнт-сервер. Також використовується для адресації пристроїв в невеликих мережах (адресація виду "name@domain.com").

- MQTT (Message Queue Telemetry Transport) - цей протокол використовує збір інформації від безлічі вузлів і передачу даних на сервер. Архітектура має тип, видавець-передплатник з використанням брокера (проміжного сервера), брокер потрібен для складання пріоритетів повідомлень і формування черг. Транспорт є TCP. На основі MQTT був сформований спеціалізований протокол MQTT-SN для сенсорних мереж.

- CWMP (CPE WAN Management Protocol). Протокол управління і моніторинг абонентським обладнанням (CPE) відповідно до специфікації TR-0695. Протокол призначений для автоматичної конфігурації та динамічної підготовки сервісів до роботи, управління версіями програмного забезпечення, моніторингу станів і продуктивності, діагностики пристроїв і мережі в цілому.

- DNP3 (Distributed Network Protocol). Протокол передачі даних між об'єктами мережі IoT: читання / запис, вибір даних і управління процесами: пряме управління, управління подіями і т. д. Для безпечної аутентифікації використовується розширення Secure Authentication.

- GPS / GLONASS і M2M Data. Система для отримання довільних звітів від будь-яких супутникових датчиків і інших пристроїв M2M через протоколи TCP або UDP. Обробка команд здійснюється на основі бізнес-правил. У IoT дана система лежить в основі управління транспортом і служить для визначення і відстежування місцеположення, збору, зберігання, обробки і візуалізації різних телеметричних даних. З її допомогою можна також здійснювати моніторинг поточного стану автомобіля (швидкість, паливо / рівень заряду батарей, поточну витрату, рівень масла, і ін.), Експлуатацію транспортних засобів (віддалене вимикання двигуна, відправлення повідомлень водієві або оператору і т. Д.), моніторинг профілактичних і ремонтних робіт, настройку обміну даними з системою управління запасами в режимі реального часу.

- МЕК 60870-5-104. Протокол телемеханіки, призначений для передачі даних в центри управління⁸. Передача здійснюється після установки TCP з'єднання. Використовується в рішеннях домашньої і промислової автоматизації (системи кондиціонування, освітлення, відеоспостереження, отримання даних від лічильників, вимірювальних перетворювачів), контролю і управління центрами обробки даних.

- KNX. Протокол для взаємодії з пристроями KNX через IP⁹. Система KNX служить для домашньої і промислової автоматизації (керування освітленням, кліматом, забезпечення безпеки та ін.). Всі пристрої мережі (в термінах KNX - датчики, актуатори і системні пристрої) підключаються в провідній шині, через яку і обмінюються інформацією з використанням протоколу KNX. Обмін інформацією обов'язково супроводжується підтвердженням на кожне повідомлення, тому дана технологія не підходить мереж критичним до затримок.

MQTT - це протокол обміну повідомленнями за шаблоном видавець-передплатник (pub / sub). Первинну версію в 1999 році опублікували Енді Стенфорд-Кларк з IBM і Арлен Ніппер з Cirrus Link. Вони розглядали MQTT як спосіб підтримки зв'язку між машинами в мережах з обмеженою пропускнуою спроможністю або непередбачуваною зв'язком. Одним з перших варіантів його використання було забезпечення контакту фрагментів нафтопроводу один з одним і з центральними ланками через супутники.

З урахуванням суворих умов експлуатації протокол зроблений маленьким і легким. Він ідеальний для пристроїв слабкої потужності і з обмеженим часом автономної роботи. До їх числа зараз відносяться і смартфони, і постійно зростаюче число датчиків і підключених пристроїв.

Таким чином, MQTT став протоколом для потокової передачі даних між пристроями з обмеженою потужністю CPU і / або часом автономної роботи, а також для мереж з дорогою або низькою пропускнуою здатністю, непередбачуваною стабільністю або високою затримкою. Саме тому MQTT відомий як ідеальний транспорт для IoT. Він побудований на протоколі TCP / IP, але є відгалуження MQTT-SN для роботи по Bluetooth, UDP, ZigBee і в інших мережах IoT, відмінних від TCP / IP.

MQTT - не єдиний у своєму роді протокол обміну повідомленнями pub / sub в реальному часі, але він вже набув широкого поширення в різних середовищах, які залежать від міжмашинна зв'язку. Серед його однолітків - Web Application Messaging Protocol, Streaming Text-Oriented Messaging Protocol і Alternative Message Queueing Protocol.

MQTT - логічний вибір для розробників, які хочуть створювати додатки з надійною функціональністю і широкої сумісності з підключеними до інтернету пристроями і додатками, включаючи браузері, смартфони і пристрої IoT.

Як працює MQTT. Система зв'язку, побудована на MQTT, складається з сервера-видавця, сервера-брокера і одного або декількох клієнтів. Видавець не вимагає яких-небудь налаштувань за кількістю або розташуванню передплатників,

які отримують повідомлення. Крім того, передплатникам не потрібно настрійка на конкретного видавця. В системі може бути кілька брокерів, що поширюють повідомлення.

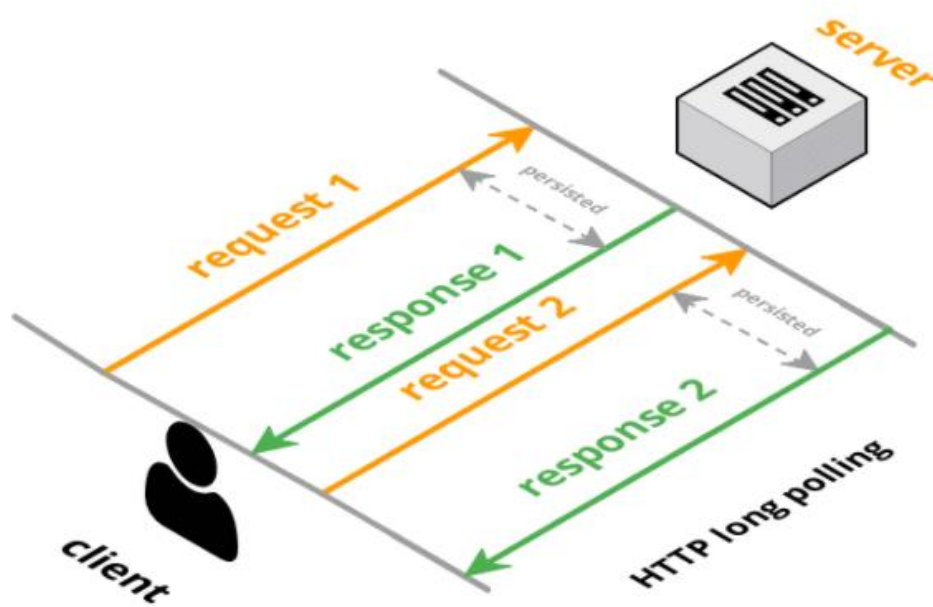


Рисунок 2.13 – Принцип роботи MQTT

MQTT надає спосіб створення ієрархії каналів зв'язку - свого роду гілку з листям. Всякий раз, коли у видавця є нові дані для поширення серед клієнтів, повідомлення супроводжується приміткою контролю доставки. Клієнти більш високого рівня можуть отримувати кожне повідомлення, в той час як клієнти більш низького рівня можуть отримувати повідомлення, що відносяться тільки до одного або двох базових каналах, «відгалужується» в нижній частині ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 мегабайт.

Будь-які дані, опубліковані або отримані брокером MQTT, будуть закодовані в довічному форматі, оскільки MQTT є бінарним протоколом. Це означає, що для отримання справжнього вмісту потрібно інтерпретувати повідомлення.

Брокери MQTT іноді можуть накопичувати повідомлення, пов'язані з каналами, у яких немає поточних передплатників. У цьому випадку повідомлення будуть або відкинуті, або збережені, в залежності від інструкцій в керуючому повідомленні. Це корисно в тих випадках, коли новим абонентам може знадобитися найостанніша записана точка даних, замість того, щоб чекати наступної відправки.

Примітно, що MQTT передає облікові дані безпеки відкритим текстом, інакше не підтримується аутентифікація або функції безпеки. Ось де вступає в гру фреймворк SSL, допомагаючи захистити передану інформацію від перехоплення або інший підробки.

Крім того, в MQTT можна використовувати аутентифікацію Ably на токенах, якщо ви взагалі не хочете розкривати свій ключ API фактичного клієнту MQTT (в разі MQTT без SSL токени обов'язкові, щоб запобігти передачі ключів API відкритим текстом).

Згідно IBM, MQTT має такі властивості:

- нейтральний до перегляду повідомлення;
- ідеально підходить для розподілених комунікацій «один до багатьох» і роз'єднаних додатків;
- оснащений функцією LWT (Last Will and Testament, «остання воля і заповіт») для повідомлення сторін про аномальний відключенні клієнта;
- покладається на TCP / IP для базових завдань зв'язку;
- розроблено для доставки повідомлень по шаблонах «максимум один раз», «мінімум один раз» і «рівно один раз».

Учасник системи MQTT може взяти на себе роль видавця, споживача або обидві ролі відразу.

Однією з відмінних характеристик MQTT є унікальне розуміння каналів: кожен з них обробляється як шлях до файлу, наприклад: `channel = "user / path / channel"`.

Канали гарантують, що кожен клієнт отримує повідомлення, призначені для нього. Обробляючи канали як шляху до файлів, MQTT виконує всі види корисних

функцій зв'язку, в тому числі фільтрацію повідомлень на основі того, де - на якому рівні або в якій гілці - клієнти підписуються на шлях до файлу.

Формат повідомлень MQTT. Подивіться на два компонента, з яких складається кожне повідомлення по протоколу MQTT:

Байт 1: містить тип повідомлення (запит клієнта на підключення, підтвердження підписки, запит ping і т. Д.), Прапор дублювання, інструкції для збереження повідомлень і інформацію про рівень якості обслуговування (QoS).

Байт 2: містить інформацію про залишилася довжині повідомлення, включаючи корисне навантаження і будь-які дані в заголовку необов'язковою змінної.

Прапор QoS в байті 1 заслуговує на особливу увагу, так як він лежить в основі змінної функціональності, яку підтримує MQTT. Прапори QoS містять такі значення, засновані на намір і терміновості повідомлення:

0 = не більше одного разу: сервер спрацьовує і забуває. Повідомлення можуть бути втрачені або продубльовані.

1 = принаймні один раз: одержувач підтверджує доставку. Повідомлення можуть дублюватися, але доставка гарантована.

2 = рівно один раз: сервер забезпечує доставку. Повідомлення надходять точно один раз без втрати або дублювання.

Давайте розглянемо, як використовувати різні рівні QoS в пристроях IoT та інших додатках.

Оскільки додатки IoT нині впроваджуються в величезних масштабах, MQTT потрапив в центр уваги як відкритий, простий і масштабований спосіб розгортання розподілених обчислень і функціональності IoT для ширшої призначеної для користувача бази - як на споживчому, так і на промисловому ринках.

Як зазначено вище, MQTT - легкий протокол обміну повідомленнями, побудований для ненадійних мереж і пристроїв з обмеженнями на джерело живлення і CPU. Однак це не означає, що зв'язок з потенційною втратою пакетів - його єдиний додаток. MQTT надає різні рівні обслуговування для різних типів

інфраструктури IoT, від повторюваної вибірки даних до управління промисловими машинами:

Дані датчиків навколишнього середовища: як уже згадувалося, MQTT підтримує модель доставки повідомлень «не більше одного разу». У мережах з частковим покриттям території або високою затримкою це означає, що інформація може бути втрачена або дублюватися. В областях, де вилучені датчики записують і передають дані з заданими інтервалами, це не є проблемою, так як нові свідчення надходять на регулярній основі. Датчики у віддалених середовищах - зазвичай малопотужні пристрої, що робить MQTT ідеальним рішенням для сенсорів IoT з відносно низьким пріоритетом передачі даних.

Дані про працездатність машин: для швидкого реагування на виникаючі проблеми і запобігання простоїв. Наприклад, для вітроелектричної установки потрібна гарантована доставка поточних показників про працездатність місцевим командам ще до того, як ця інформація потрапить в центр обробки даних. У таких ситуаціях доставка повідомлень «принаймні один раз» гарантує, що відповідні прапори своєчасно помітять потрібні фахівці, навіть якщо вони надходять як дублікати. Це важливо для міжмашинна зв'язку з більш високим пріоритетом.

Білінгові системи: є ще більш пріоритетні і точні повідомлення, які потрібно правильно обробляти. У бізнес-ситуаціях, де дублювання записів неприйнятно, в тому числі в білінгових системах, стане в нагоді прапор QoS передачі «точно один раз». Це усуває дублювання або втрату пакетів в системах виставлення рахунків або білінгу, скорочує кількість аномалій і непотрібних протиріч за погодженням.

У розробників є багатий вибір протоколів для проектування і розгортання двонаправлених каналів зв'язку IoT, включаючи MQTT, HTTP, CoAP, WebSockets (якщо дозволяє CPU / батарея) та інші. Чи є MQTT кращим вибором, залежить від обладнання і завдання програми.

Розроблений для середовищ з надзвичайно низькою пропускну здатністю, протокол MQTT може бути досить негнучким в своєму прагненні зберегти кожен байт. Наприклад, специфікація визначає лише п'ять повідомлень про помилку, за

допомогою яких сервер може відхилити з'єднання (наприклад, невірне ім'я користувача / пароль або неприйнятна версія протоколу). Якщо сервер хоче вказати на якусь іншу помилку, йому не пощастило. Гірше того, якщо помилка виникає після запуску з'єднання, немає ніякого механізму для повідомлення про помилку взагалі. Сервер може тільки потиснути плечима і різко перервати TCP-з'єднання, залишивши клієнта без поняття, чому його скинули (і без будь-якого способу відрізнити навмисне відключення від тимчасової мережевої проблеми). Для людей, які звикли до більш гнучких і простим в налагодженні (хоча і менш економічним по пропускній здатності) протоколам pub / sub, такий спартанський підхід може здатися трохи примітивним.

MQTT часто згадується разом з HTTP, тому компанія Google провела дослідження, порівнявши їх за часом відгуку, обсягом трафіку і іншим важливим для розробників атрибутом. MQTT зайняв перше місце в тестах Google, але тільки в умовах, коли з'єднання можна повторно використовувати для відправки кількох корисних навантажень.

HTTP і MQTT є хорошим вибором для додатків IoT через досить малого обсягу трафіку, низьких вимог до батареї і пам'яті.

CoAP - ще один протокол, який часто порівнюють з MQTT для розробки систем IoT. Вони схожі, але є помітні відмінності. MQTT - це протокол «багато до багатьох», в той час як CoAP - це в основному протокол «один до одного» для зв'язку між сервером і клієнтом. У той же час CoAP надає функції метаданих, виявлення і узгодження вмісту, яких немає у MQTT.

У тих випадках, коли клієнти повинні тільки отримувати дані, Server-Sent Events - теж підходящий варіант.

Eclipse Mosquitto - брокер повідомлень з відкритим вихідним кодом (ліцензії EPL / EDL), який реалізує протоколи MQTT версій 5.0, 3.1.1 та 3.1. Mosquitto легкий і підходить для використання на всіх пристроях: від малопотужних одноплатних комп'ютерів до повноцінних серверів.

Коли мова йде про масштабування MQTT, слід врахувати два міркування: 1) правильний це протокол; 2) незалежно від вибору протоколу, які інфраструктура і мережеві можливості необхідні для обробки зростаючого трафіку між пристроями по MQTT.

Lightweight Machine-to-Machine (LWM2M) - ще один протокол, який можна використовувати разом з MQTT на рівні підприємства. У порівнянні з MQTT, він іноді краще підходить для довготривалих систем IoT. MQTT ідеально підходить для пробного запуску IoT без особливих зусиль, в той час як LWM2M надає функції для довгострокової універсальної інфраструктури. LWM2M також надає чудові інструменти управління пристроями, такі як моніторинг підключення, оновлення прошивок і віддалені дії на пристроях. Для підприємств з великою кількістю некерованих пристроїв, які відправляють великі обсяги даних на центральну платформу, LWM2M є найкращим вибором. Проте, ми говоримо про масштабні розгортання IoT, тому зазвичай MQTT більш ніж адекватний варіант. Крім того, MQTT більш поширений і у нього більш широка підтримка.

Тепер про можливості інфраструктури. Коли мова заходить про завантаження сервера, то рідко вузьким місцем є кількість одночасних підключень. Більшість хороших серверів / брокерів MQTT підтримують тисячі одночасних підключень, але яка робоче навантаження, необхідна для обробки і відповіді на повідомлення після того, як сервер MQTT отримав фактичні дані? Як правило, є всілякі потенційні проблеми, такі як читання і запис в базу даних і з неї, інтеграція з сервером, розподіл і управління ресурсами для кожного клієнта і т. Д. Як тільки одна машина перестає справлятися з навантаженням, потрібно додавати додаткові сервери, тобто думати про балансуванні навантаження, синхронізації повідомлень між клієнтами, підключеними до різних серверів, узагальненому доступі до стану клієнта незалежно від терміну з'єднання або конкретного сервера, до якого підключений клієнт - список продовжується і продовжується.

3 РОЗРОБКА СИСТЕМИ "РОЗУМНА ПОЛИЦЯ"

3.1 Загальна архітектура системи

Для даної системи була розроблена дана архітектура (рис.3.1).

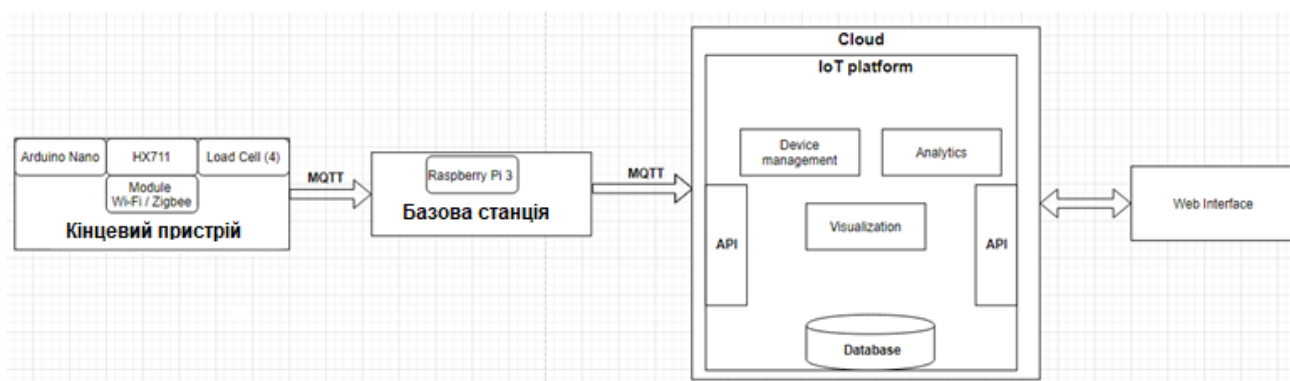


Рисунок 3.1 - Архітектура рішення

У нас є кінцеві пристрої, в даному рішенні кінцевими пристроями у нас є: мікроконтролер Arduino Nano, мікросхема HX711, 4 тензодатчика і модуль Wi-Fi / Zigbee. Дані пристрої передають дані по протоколу MQTT на базову станцію, базовою станцією є плата Raspberry Pi 3, далі ці дані відправляються на платформу, а в подальшому і на інтерфейс користувача, більш конкретно технології та устаткування описуються далі.

3.2 Вибір і інтеграція кінцевих пристроїв і технологій передачі даних

З множин пристроїв для розробки системи "розумна полиця" було обрано такі рішення.

Основою системи є мікроконтролер Arduino Nano (рис. 3.2) побудована на чіпсеті Atmega328.

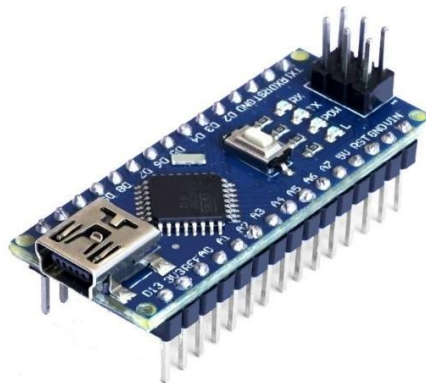


Рисунок 3.2 - ArduinoNano

Платформа Arduino на сьогоднішній день є однією з популярних в світі, для неї існує безліч бібліотек і різних інструментів, які знадобляться для рішень простих і складних завдань [8]. Для своїх проєктів на Arduino не важко буде знайти інструкції та відповідні для нього сумісні платформи. Також з плюсів для даного мікроконтролера можна виділити, низьке електроспоживання і низьку вартість. Виходячи з поставленого завдання, розробки IoT-системи для рітейлу, а саме розробки системи "розумна полиця" був обраний мікроконтролер Arduino Nano.

При роботі з Arduino Nano використовується Arduino IDE, дана програма - це кроссплатформний додаток, написаний на функціях з C і C++. Він використовується для написання і завантаження програм на Arduino-сумісній платі, а також, за допомогою ядер сторонніх виробників, на плати розробки інших виробників.

На даному рисунку зображено середовище розробки Arduino IDE і написаний в ній код для роботи системи "розумна полиця" (рис. 3.3). Дане середовище є зручним для написання коду і прошивки мікроконтролерів, багато різних інструкцій для даного середовища. Функції, бібліотеки і важливі речі підсвічуються певним кольором, що дозволяє робити код легко і швидко читається.



Рисунок 3.3 - Arduino IDE

Для роботи системи "розумна полиця" потрібні датчики, які змогли б визначити вагу предметів, які знаходяться полиці, яка оснащена даною системою. Таке завдання вирішують такі датчики, як тензодатчики. Існує безліч видів даних датчиків, але для даної системи були обрані саме полумостові тензодатчики до 50 кг. (рис. 3.4).



Рисунок 3.4 - Полумостовий тензодатчик

Таких датчиків в даній системі використовується чотири штуки і розміщуються вони по кутах полиці, для більш точних вимірювальних даних. Перевага таких датчиків є, низька вартість і при використанні в комплексі таких чотирьох датчиків можна отримати вимірювані ваги до 200 кг.

Для отримання даних з тензодатчиків (датчиків, що перетворюють величину деформації в електричний струм) використовується мікросхема HX711 (рис. 3.5).

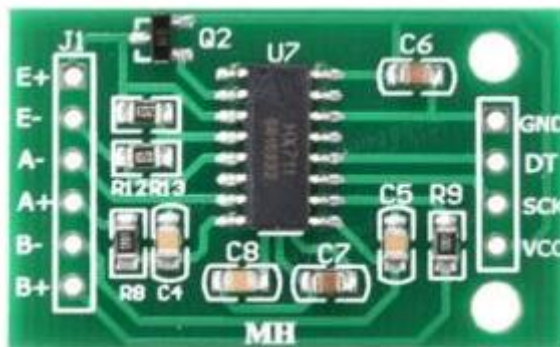


Рисунок 3.5 - Мікросхема HX711

Основа модуля - спеціалізована мікросхеми HX711, має АЦП з роздільною здатністю 24 біт, завдяки чому забезпечується висока точність вимірювань (для порівняння - дозвіл АЦП, вбудованого в Arduino 10 біт). Модуль має 2 канали А і В, до яких можна підключити два тензодатчика. Для каналу А коефіцієнт посилення дорівнює 64 або 128, для каналу В коефіцієнт посилення дорівнює 32. Модуль працює з частотою вимірювання 80 Гц [9].

У комплексі ці пристрої є кінцевими пристроями. Кінцеві пристрої - це ті пристрої, які ініціалізують процес передачі даних, тобто саме вони починають передавати або запитувати у інших кінцевих пристроїв. Були підключені чотири тензодатчика з мікросхемою ХН711 (рис. 3.6).

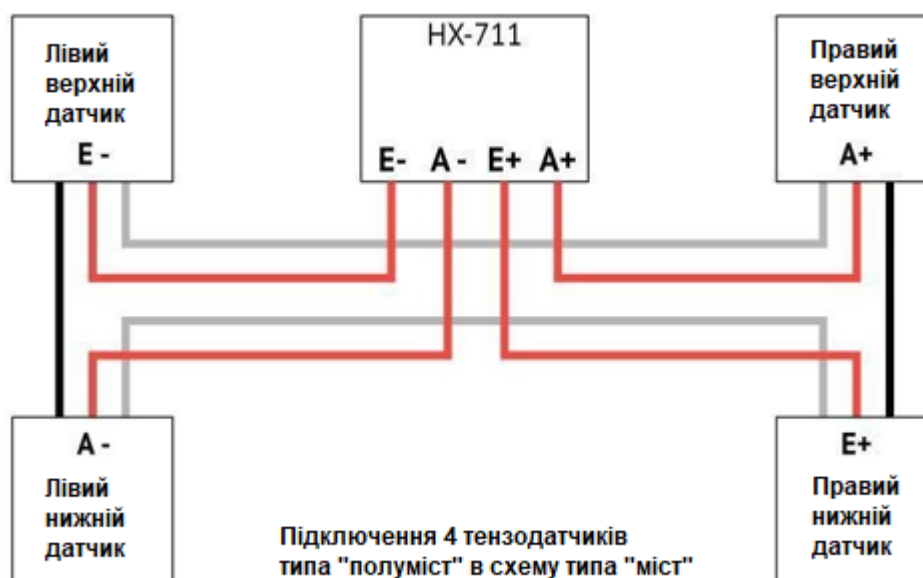


Рисунок 3.6 - Схема підключення тензодатчиків

Після того, як підключили тензодатчики з мікросхемою HX711, підключили мікросхему до Arduino Nano, що і є кінцевим пристроєм (рис. 3.7).

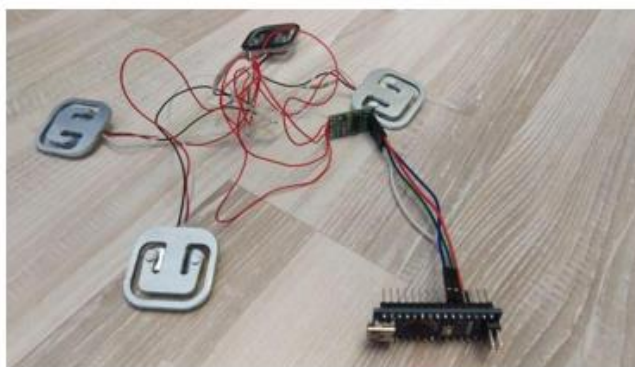


Рисунок 3.7 - Кінцевий пристрій

Наступним етапом є написання коду для даних пристроїв і калібрування тензодатчика. Для написання коду для даних пристроїв існує дуже зручна програма, яка допомагає нам генерувати код. Програма називається Visuino.

Visuino - це візуальна графічне середовище розробки CONTROLLINO. Вона дозволяє створювати складні промислові системи автоматизації та рішення IoT. Візуалізація даної програми для нашої задачі (рис. 3.8).

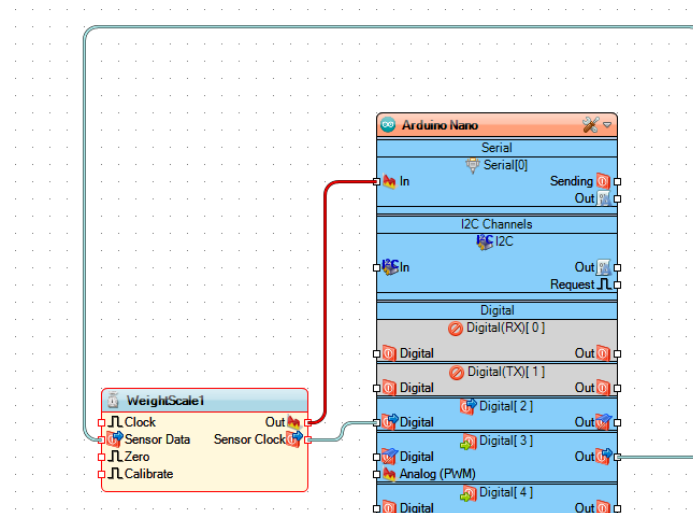


Рисунок 3.8 - Графічне середовище Visuino

Після цього переходимо до згенерованого коду і дивимося, що і де потрібно додати або змінити під свою задачу (ліст.3.1).

Лістинг 3.1 – Підключення бібліотек

```
// Підключаємо потрібні нам бібліотеки
#include <OpenWire.h>
#include <Mitov.h>

#include <Mitov_StandardSerial.h>
#include <Mitov_HX711_Weight.h>
namespace BoardDeclarations {} // в даній функції оголошуються
вхідні і вихідні порти, піни, які підключені, встановлюється
швидкість передачі даних через UART
namespace Declaration {} // в даній функції декларуються
зв'язок між пінами
namespace PinCalls {} // спілкування між пінами
namespace ComponentsHardware {

void SystemUpdateHardware ()} // компоненти пристрою

void setup () {} // функція настройки викликається один раз при
запуску ескізу
void loop () {} // функція крутиться в нескінченному loop
```

Даний код завантажений в мікроконтролер, знімає показання з чотирьох тензодатчиків і ми вже можемо бачити цілком робочу систему.

Після того, як ми отримали дані з тензодатчиків та провели їхнє калібрування, нам потрібно передати ці дані на базову станцію. Базовою станцією в нашій системі служить плата Raspbbery Pi.

До цієї плати була встановлена ОС (операційна система), а саме система Raspberry Pi OS заснована на Debian. Було встановлено безліч програм потрібних для нашої задачі. Наприклад, такі програми як: Arduino IDE, Visuino, Mosquitto, платформа IoT ioBroker.

Raspbbery Pi - одноплатний комп'ютер (рис. 3.9) розміром з банківську карту, дана плата потрібна для того, щоб приймати отримані дані і за допомогою Wi-fi / Ethernet відправляти ці дані на платформу.



Рисунок 3.9 - Плата Raspbbery Pi

Таблиця 3.1 - Характеристика плати Raspbbery Pi

Процесор	64-бітний чотирьохядерний ARM Cortex- A53 з тактовою частотою 1,2 ГГц на однокристальному чипі Broadcom BCM2837
Оперативна пам'ять	1ГБ LPDDR2 SDRAM
Цифровий відеовихід	HDMI
Композитний вихід	3,5 мм (4 pin)
USB порти	USB 2.0 × 4
Мережа	WiFi 802.11n, 10/100 M6 RJ45 Ethernet
Bluetooth	Bluetooth 4.1, Bluetooth Low Energy
Роз'єм дисплея	Display Serial Interface (DSI)
Роз'єм відеокамери	MIPI Camera Serial Interface (CSI-2)
Карта пам'яті	MicroSD
Порти введення-виведення	40
Габарити	85x56x17 мм

Наступним етапом є передача даних з базової станції на платформу, для передачі даних було обрано протокол MQTT.

MQTT (Message Queue Telemetry Transport) - це легкий, компактний і відкритий протокол обміну даними створений для передачі даних на віддалених локаціях, де потрібний невеликий розмір коду і є обмеження по пропускній здатності каналу. Перераховані вище якості дозволяють застосовувати його в системах M2M (Машинно-Машинне взаємодія) і ПоТ (Промисловий Інтернет речей).

Основні особливості протоколу MQTT:

- асинхронний протокол;
- компактні повідомлення;
- робота в умовах нестабільного зв'язку на лінії передачі даних;
- підтримка декількох рівнів якості обслуговування (QoS);
- легка інтеграція нових пристроїв.

Протокол MQTT працює на прикладному рівні поверх TCP / IP (рис. 3.10) і використовує за замовчуванням +1883 порт (8883 при підключенні через SSL) [10].

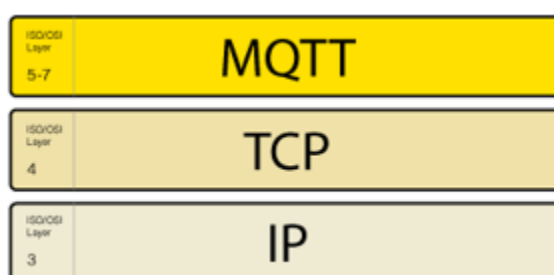


Рисунок 3.10 - Протокол MQTT

Обмін повідомленнями в протоколі MQTT (рис. 3.11) здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher / subscriber) повідомлень, і брокером (broker) повідомлень (наприклад, Mosquitto MQTT).



Рисунок 3.11 - Схема простої взаємодії

Видавець відправляє дані на MQTT брокер, вказуючи в повідомленні певну тему, топик (topic). Підписники можуть отримувати різні дані від безлічі видавців залежно від підписки на відповідні топіки.

Код для прийому і відправки даних на платформу писався на мові Python. Для роботи з протоколом MQTT в Python використовується бібліотека `paho-mqtt`.

Лістинг 3.2 - Код для прийому і відправки даних

```
// Підключення бібліотек paho-mqtt для відправки даних
import paho.mqtt.client as mqtt
import paho.mqtt.publish as publish
Підключення бібліотеки для зчитування даних з порту
import serial
// Підключення бібліотеки для створення таймера (через який час
будуть надходити дані) import time
// Ця функція зчитує дані з порту і повертає їх
def get_weight ()
// Функція публікує дані
def pub (topic, val)
// І нескінченний цикл while в якому використовується функція
публікації і таймера (через який час повторно відправити дані)
pub ()
time.sleep ()
```

Після цього дані відправляються на платформу IoT, а саме на платформу IoBroker.

З усіх платформ була обрана платформа ioBroker, так як дана платформа дуже зручна і має ту функціональність, яка підходить для поставленого завдання.

Для початку треба було поставити дану платформу на плату Raspberry Pi, після того, як ми все успішно встановили, нам потрібно запустити її. Дана платформа запускається за допомогою команд (рис. 3.12).

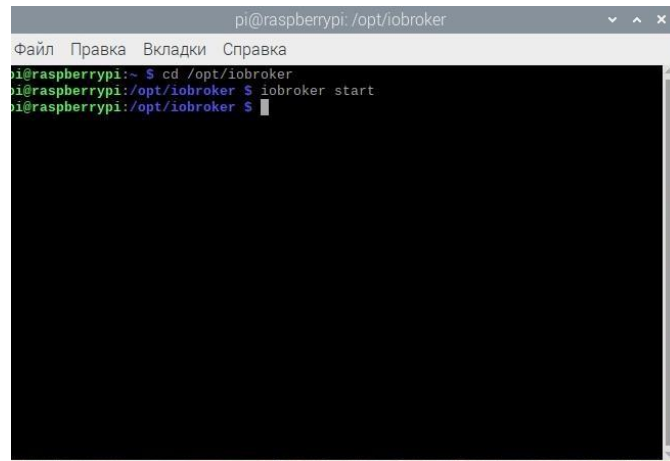


Рисунок 3.12 - Запуск платформы ioBroker

Після цього переходимо за адресою і потрапляємо в панель управління адміністратора платформи ioBroker (рис. 3.13).

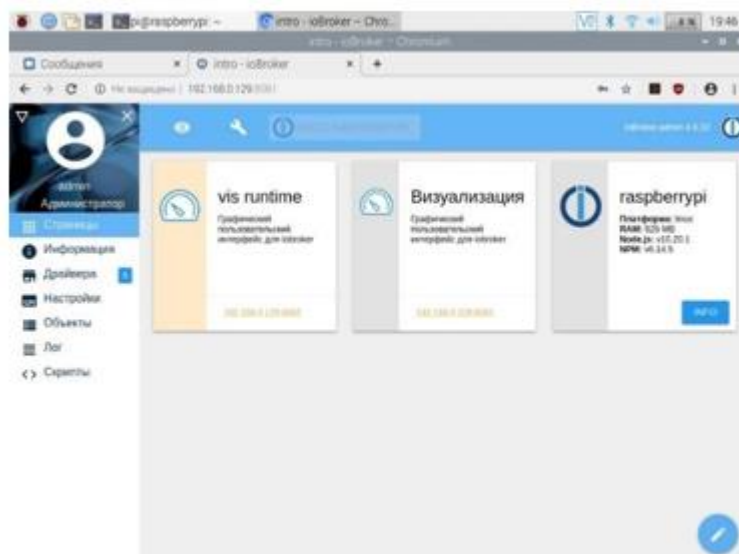


Рисунок 3.13 - Панель управління адміністратора платформи ioBroker

Далі встановлюємо всі необхідні нам інструменти для вирішення нашої задачі, вже встановлені і налаштовані драйвера показані на даному малюнку (рис. 3.14).

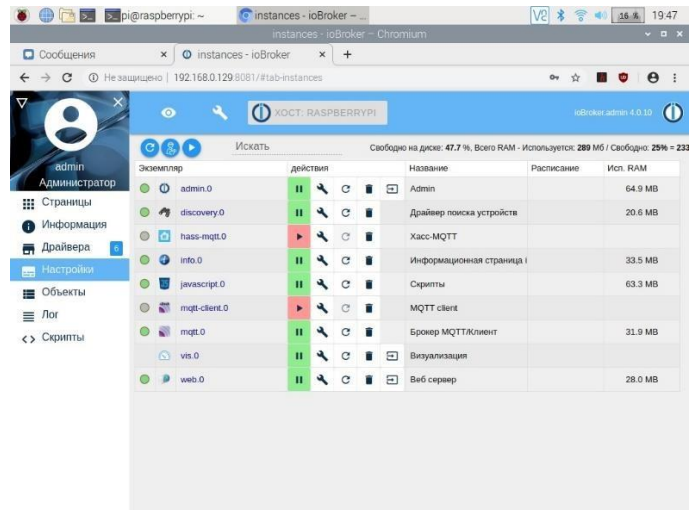


Рисунок 3.14 - Встановлені і налаштовані драйвера

Тепер запускаємо MQTT сервер за допомогою команди `mosquitto` і запускаємо наш скрипт, написаний на Python, створює топик з назвою "q" і багатозначно знятих з датчиків і передається на платформу (рис. 3.15).

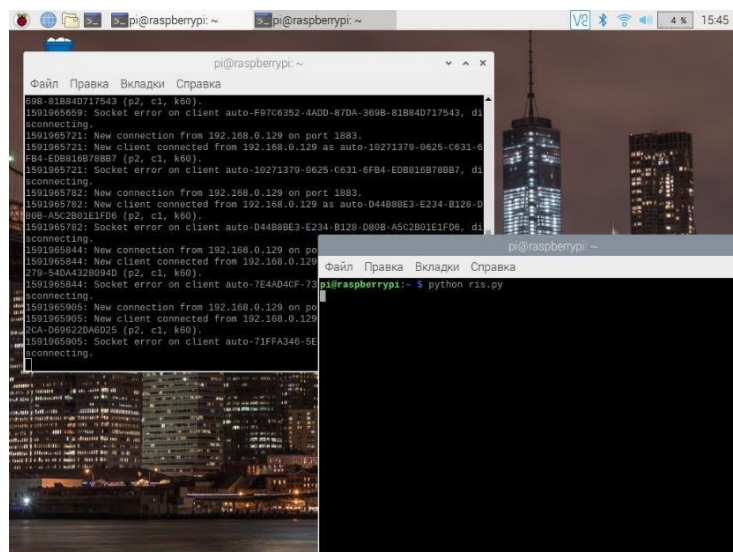


Рисунок 3.15 - Запуск MQTT і скрипта на Python

Після переходу на платформу в вкладку "об'єкти" ми бачимо топик з назвою "q" і отримані дані з датчиків (рис. 3.16).

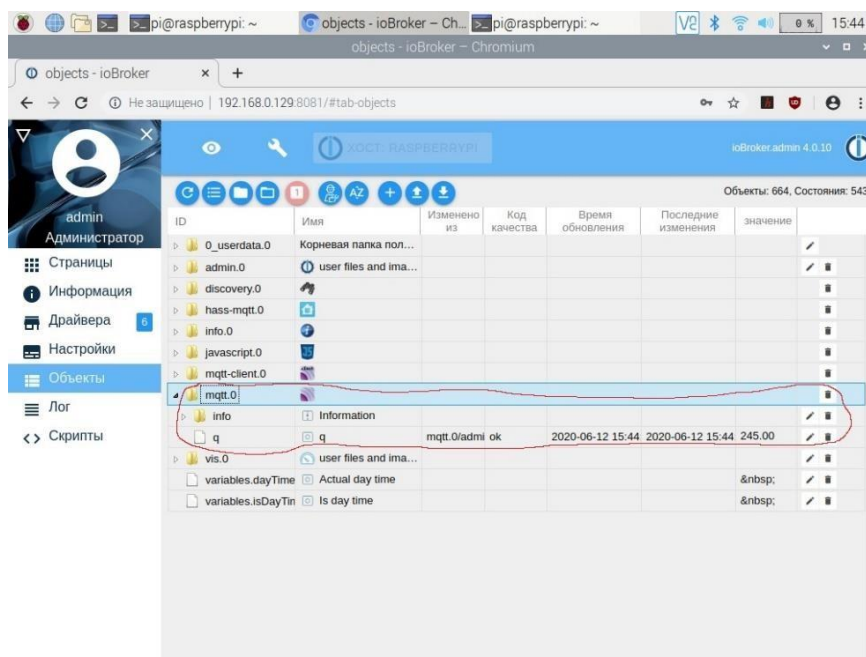


Рисунок 3.16 - Дані з датчиків

3.3 Застосування і розвиток системи

Розроблена система "розумна полиця" може бути використана в ритейлі для контролю роботи товарознавців. Фактично з її допомогою можна впровадити систему KPI, і контролювати якість роботи по наповненню асортиментом магазину. Наприклад, якщо в якийсь момент товарна позиція скоротиться нижче певного рівня система дасть сигнал і розрахує тривалість часу, протягом якого для покупців був недоступний товар відповідно, також можна оцінити рівень втраченої виручки.

Крім цього система може бути доповнена аналітичним модулем, який буде оцінювати інтенсивність купівлі товару в залежності від часу доби і різних акцій мерчанта. Таким чином для постачальників товарів у велику мережу буде забезпечений додатковий аналітичний сервіс, що дозволяє найбільш ефективно налаштовувати товарну і цінову політику.

Що стосується конструкції полиці і клієнтського додатка, в майбутньому планується доопрацювати та вдосконалити систему, а саме розробити презентабельний вигляд для самої полиці, розробити зручний і красивий веб-додаток, за допомогою якого можна буде відстежувати товари за допомогою мобільних пристроїв, а також застосовувати машинне навчання.

ВИСНОВКИ

Зараз в світі йде трансформація економічних систем і великий попит приходить на рішення задач за допомогою Internet of Things. Дані технології допомагають автоматизувати і відстежувати багато речей в різних галузях, що веде, наприклад, до мінімізації втрат від продажів в ритейлі.

У даній роботі проведено огляд різних IoT-систем, які існують на сьогоднішній день, які використовуються в ритейлі, наводяться їх приклади і які завдання вони вирішують. Також наведено аналіз різних мікроконтролерів, бездротових типів мереж, тензодатчиків, платформ і протоколів, проаналізовано плюси і мінуси тих чи інших пристроїв і технологій.

В роботі наведено розробку своєї IoT-системи, а саме розробку IoT - системи контролю товарних запасів в ритейлі. Дана система - "розумна полиця", вона дозволяє управляти і стежити за динамікою товарів на полиці і визначати в яких полицях пора виставляти закінчені або відсутні товари, а також стежити за якістю роботи товарознавців.

За допомогою рішення «розумна полиця» можна зменшити ризики зараження, в даний час пов'язані з коронавірусною інфекцією, за рахунок зниження контакту працівників і покупців.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шваб К. Четвертая промышленная революция. М.: Изд-во Эксмо, 2016. 134 с.
2. Интернет вещей: эволюция или революция. www.aig.ru/content/dam/aig/emea/russia/documents/business/brochure.pdf (дата звернення 10.05.2021)
3. Розничные бизнес-решения для цифрового будущего физического магазина. www.business.panasonic.co.uk/solutions/retail (дата звернення 10.05.2021)
4. Визуальные технологии в ритейле. www.retailer.ru/vizualnye-tehnologii-vritejle (дата звернення 10.05.2021)
5. Технологии и инновации – Интернет вещей (Internet of Things). <https://www.it.ua/ru/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot> (дата звернення 10.05.2021)
6. ПО Arduino IDE. <https://www.arduino.cc/en/main/software> (дата звернення 10.05.2021)
7. Тензодатчик: принцип работы, устройство, типы, схемы подключения. <https://www.asutpp.ru/tenzodatchik.html> (дата звернення 10.05.2021)
8. Аппаратное обеспечение Arduino Nano. <http://arduino.ru/Hardware/ArduinoBoardNano> (дата звернення 10.05.2021)
9. АЦП для аналоговых весов HX711. <https://3d-diy.ru/wiki/arduino-moduli/acp-dlya-analogovykh-vesov-hx711/> (дата звернення 10.05.2021)
10. Что такое MQTT и для чего он нужен в IoT? Описание протокола MQTT. <https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/> (дата звернення 10.05.2021)
11. Куприяновский В.П., Намиот Д.Е., Дрожжинов В.Е., Куприяновская Интернет вещей на промышленных предприятиях. 2016. 352 с.

12. Буланча С.А., Куприяновский В.П., Черных К.Ю., Намиот Д.Е. Умные города как “столицы” цифровые экономики. 2015. 254 с.
13. Намиот Д.Е., Зубарева Е.В. Города управляемые данными 2016. 216 с.
14. Щербинина М.Ю., Стефанова Н.А. Концепция интернет вещей. 2015. 199 с.
15. Намиот Д.Е. Об учебных программах по Internet of Things. 2015. 147 с.
16. Бондарик В.Н., Кучерявый Е.А. Прогнозирование развития Интернета Вещей на горизонте планирования до 2030 года. 2015. 280 с.
17. Полукшин Н.А. Интернет Вещей: технология, способная изменить мир. 2015. 155 с.
18. Бородин В.А. Интернет вещей – следующий этап цифровой революции. 2013. 116 с.
19. Яненко М.Б., Яненко М.Е. Маркетинг взаимодействия в информационной экономике: проблемы и перспективы развития интернета вещей. 2014. 202 с.
20. Городнищева А.Н., Замятина Э.В. Интернет вещей и его место в информационном обществе. 2015 г. 312с.

ДОДАТОК А

Код для роботи тензодатчиків і мікросхеми HX711

Лістинг А1 - Код для роботи тензодатчиків і мікросхеми HX711

```

    #define VISUINO_ARDUINO_NANO

#include<OpenWire.h>
#include <Mitov.h>
#include
<Mitov_StandardSerial.h>
#include
<Mitov_HX711_Weight.h>

    namespace ComponentVariables
    {
        class
        {
        public:
            bool Value1: 1;
        }
    BitFie
    lds;
    class
    Variab
    le1
    {
    public:
        inline static bool GetValue () {return BitFields.Value1; }
        inline static void SetValue (bool AValue) {BitFields.Value1
= AValue; }
    };
    }

    namespace VisuinoConstants
    {
        class FloatValue1
        {

```

```

public:
    inline static constexpr float GetValue () {return -132699;
}

};

class FloatValue0
{
public:
    inline static constexpr float GetValue () {return 1; }
};
}

namespace PinCalls
{
class PinCallerReceive0
{
public:
    void Notify (void * _Data);
};
} // PinCalls

// Arduino Board Declarations namespace BoardDeclarations
{
    Mitov :: SerialPort <SERIAL_TYPE, // 0_T_TYPE
    Serial, // 1_C_TYPE

    Mitov    ::    ConstantProperty    <9,    uint32_t,    0>,    //
AfterSendingDelay Mitov :: ConstantProperty <2, uint32_t, 8>, //
DataBits Mitov :: ConstantProperty <8, bool, true>, // Enabled
Mitov :: ConstantProperty < 51, uint32_t, 0>, // FEndTime Mitov ::
ConstantProperty <50, bool, false>, // FSending Mitov ::
GenericPin_NoImplementation <5>, // OutputPin

    Mitov    ::    ConstantProperty    <1,    Mitov    ::
TArduinoStandardSerialParity, Mitov :: spNone>, // Parity Mitov ::
DigitalPin_NoImplementation <7>, // SendingOutputPin

    Mitov :: ConstantProperty <4, uint32_t, 9600>, // Speed Mitov
:: ConstantProperty <3, uint32_t, 1> // StopBits

    SerialPort0;

```

```

        Mitov    ::    ArduinoSerialInput    <Mitov    ::    SerialPort
<SERIAL_TYPE, // 0_T_TYPE

        Serial, // 1_C_TYPE

        Mitov    ::    ConstantProperty    <9,    uint32_t,    0>,    //
AfterSendingDelay Mitov :: ConstantProperty <2, uint32_t, 8>, //
DataBits Mitov :: ConstantProperty <8, bool, true>, // Enabled
Mitov :: ConstantProperty < 51, uint32_t, 0>, // FEndTime Mitov ::
ConstantProperty <50, bool, false>, // FSending Mitov ::
GenericPin_NoImplementation <5>, // OutputPin

        Mitov    ::    ConstantProperty    <1,    Mitov    ::
TArduinoStandardSerialParity, Mitov :: spNone>, // Parity Mitov ::
DigitalPin_NoImplementation <7>, // SendingOutputPin

        Mitov :: ConstantProperty <4, uint32_t, 9600>, // Speed Mitov
:: ConstantProperty <3, uint32_t, 1> // StopBits

        >, SerialPort0, float> SerialPort0_InputFloat_1; Mitov ::
ArduinoDigitalOutputChannel <Mitov :: ConstantProperty <22, bool,
false>, // IsAnalog Mitov :: ConstantProperty <24, bool, false>,
// IsCombinedInOut Mitov :: ConstantProperty <21, bool, false>, //
IsOpenDrain Mitov :: ConstantProperty <26, bool, false>, //
IsOutput Mitov :: ConstantProperty <20, bool, false>, // IsPullDown
Mitov :: ConstantProperty <4, bool, false>, // IsPullUp Mitov ::
ConstantProperty < 23, bool, false>, // IsRawInput Mitov ::
DigitalPin_NoImplementation <3>, // OutputPin

        3 // PIN

        ArduinoDigitalChannel_3;

        } // BoardDeclarations

        // Declarations namespace Declarations

        {

        Mitov :: HX711_Weight <Mitov :: ArduinoDigitalOutputChannel
<

        Mitov :: ConstantProperty <22, bool, false>, // IsAnalog
Mitov :: ConstantProperty <24, bool, false>, // IsCombinedInOut
Mitov :: ConstantProperty <21, bool, false>, // IsOpenDrain Mitov
:: ConstantProperty < 26, bool, false>, // IsOutput Mitov ::
ConstantProperty <20, bool, false>, // IsPullDown Mitov ::
ConstantProperty <4, bool, false>, // IsPullUp Mitov ::
ConstantProperty <23, bool, false >, // IsRawInput Mitov ::
DigitalPin_NoImplementation <3>, // OutputPin

        3 // PIN

        >, // 0_TYPE BoardDeclarations :: ArduinoDigitalChannel_3,
// 1_NAME Mitov :: ConstantProperty <12, int32_t, 1>, // Average

```

```

        Mitov    ::    ConstantProperty    <2,    bool,    false>,    //
ClockInputPin_o_IsConnected Mitov :: ConstantProperty <13, bool,
true>, // Enabled

        Mitov :: TypedVariable <0, bool, :: ComponentVariables ::
Variable1>, // FNeedsRead Mitov :: ConstantProperty <11, bool,
false>, // Gain128 Mitov :: ConstantPropertyFloat <9, float, ::
VisuinoConstants :: FloatValue1 >, // Offset

        Mitov :: AnalogPin_EmbeddedPinImplementation <14, :: PinCalls
:: PinCallerReceive0>, // OutputPin

        2, // PIN_CLOCK

        Mitov :: ConstantProperty <7, bool, true>, // Powered

        Mitov :: ConstantPropertyFloat <8, float, :: VisuinoConstants
:: FloatValue0>, // ScaleDivider Mitov :: ConstantProperty <10,
bool, false> // UseChannelB

        _o_WeightScale1;
    } // Declarations

    // Pin Call Implementations namespace PinCalls
    {
        void PinCallerReceive0 :: Notify (void * _Data)
        {
            BoardDeclarations ::
SerialPort0_InputFloat_1.InputPin_o_Receive (_Data);
        }

    } // PinCalls

    namespace ComponentsHardware
    {
        void SystemUpdateHardware ()
        {
        }

    } // ComponentsHardware

    // The setup function is called once at startup of the sketch
    void setup ()
    {

        BoardDeclarations :: ArduinoDigitalChannel_3.SystemInit ();
        BoardDeclarations :: SerialPort0.SystemInit (); Declarations :: _
o_WeightScale1.SystemInit (); BoardDeclarations ::

```

```

ArduinoDigitalChannel_3.SystemStart (); OpenWire :: SystemStarted
();

    }

    // The loop function is called in an endless loop void loop
    ()

    {

        BoardDeclarations :: ArduinoDigitalChannel_3.SystemLoopBegin
        ();BoardDeclarations      ::      SerialPort0.SystemLoopBegin      ();
        Declarations :: _ o_WeightScale1.SystemLoopBegin ();

    }

```

ДОДАТОК Б

Код на Python для передачі даних на платформу

Лістинг Б1 - Код на Python для передачі даних на платформу

```
import paho.mqtt.client as mqtt import paho.mqtt.publish as
publish import serial

import time

def get_weight ():

    ser = serial.Serial ( "/" dev / ttyUSB0", 9600) return
ser.readline ()

def pub (topic, val):

    client = mqtt.client () client.connect ( "192.168.0.129")
client.publish (topic, val)

while 1:

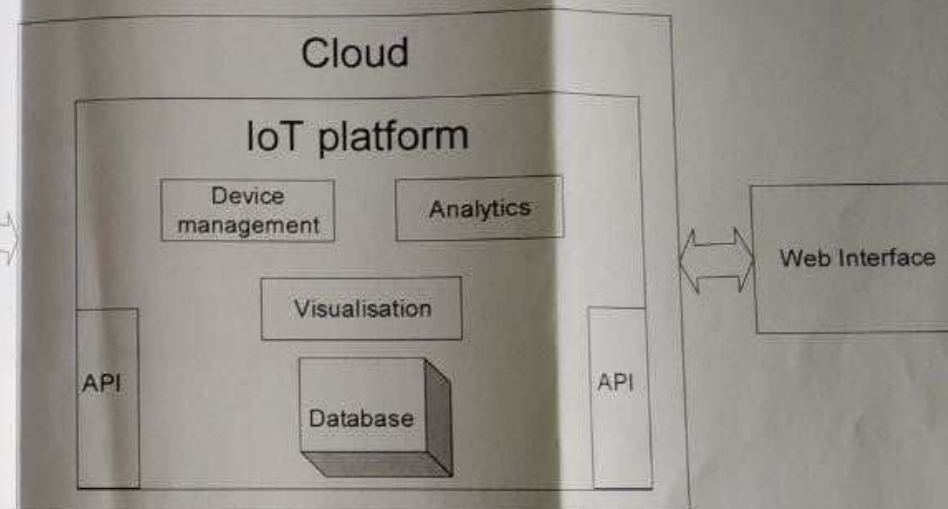
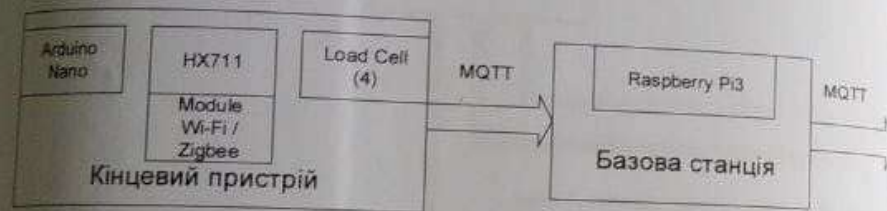
    pub ( "q", get_weight ()) time.sleep (60)
```

ХАРАКТЕРИСТИКА ПЛАТИ RASPBERRY PI

Процесор	64-бітний чотирьохядерний ARM Cortex- A53 з тактовою частотою 1,2 ГГц на однокристальному чипі Broadcom BCM2837
Оперативна пам'ять	1ГБ LPDDR2 SDRAM
Цифровий відеовихід	HDMI
Композитний вихід	3,5 мм (4 pin)
USB порти	USB 2.0 × 4
Мережа	WiFi 802.11n, 10/100 M6 RJ45 Ethernet
Bluetooth	Bluetooth 4.1, Bluetooth Low Energy
Роз'єм дисплея	Display Serial Interface (DSI)
Роз'єм відеокамери	MIPI Camera Serial Interface (CSI-2)
Карта пам'яті	MicroSD
Порти введення-виведення	40
Габарити	85x56x17 мм

					13.02070849.51705 ПЛ1			
Ім.	Лист.	№ докум.	Підп.	Дата	Розробка системи розумної поліції Характеристика плати Raspberry Pi	Лист	Маса	Масштаб
Розроб.	Власенко Я.А.			27.05				
Перев.	Гришко С.С.							
Т.контр.								
Н.контр.	Щербак Н.В.					Лист 1	Листів 1	
Затв.	Кудрявцев Р.К.					НУ «Запорізька політехніка» КНТ-517		

АРХІТЕКТУРА СИСТЕМИ



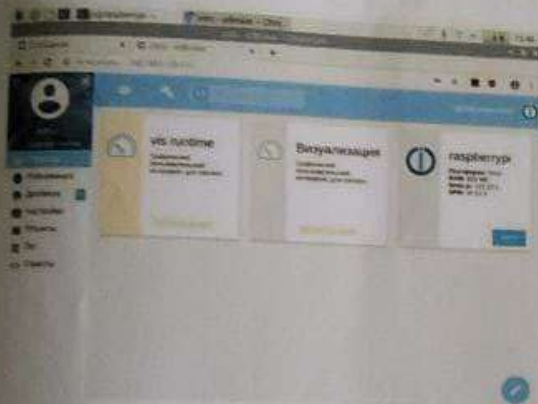
						13.02070849.51705 ПЛ2			
							Лист	Масш.	Масшт.
Зам.	Лист	№ докум.	Підп.	Дати	Розробка системи розумної полиці Архітектура системи				
Розроб.		Власенко Я.А.	<i>[Signature]</i>	21.05					
Перев.		Гришко С.С.	<i>[Signature]</i>	21.05					
Т.контр.						Лист 1	Листів 1		
Н.контр.		Щербань Н.В.	<i>[Signature]</i>			НУ «Запорізька політехніка»			
Затв.		Кудерничук Р.К.	<i>[Signature]</i>			КНТ-517			

ПІДКЛЮЧЕННЯ ТЕНЗОДАТЧИКІВ

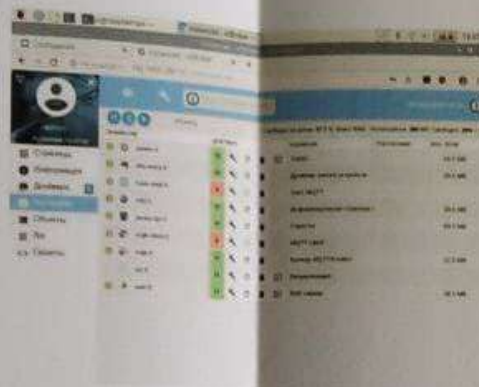


					13.02070849.51705 ПЛЗ			
Зам.	Лист	Ав. об'єкт	Проп.	Дата	Розробка системи розумної полиці Підключення тензодатчиків	Лист	Маса	Масштаб
Розроб.	Власенко Я.А.	Гулик С.С.	12.05	12.05				
Перев.								
Т.контр.						Лист 1	Листів 1	
Н.контр.	Щербак Н.В.					НУ «Запорізька політехніка» КНТ-517		
Затв.	Кудришова Р.К.							

РОБОТА З IOBROCER



Панель управління
адміністратора платформи
ioBroker



Встановлення і налаштування
драйвера



Дані з датчиків

						13.02070849.51705 ПЛ4			
							Лист	Міс.	Міс.
Зам.	Лист	Автори	Міс.	Дата	Розробка системи розумної пошти Робота з юBroker				
Розроб.		Власний Н.А.		21.03					
Перев.		Григорієв С.С.		21.03					
Т.контр.									
						Лист 1	Листа 1		
Н.контр.		Шербаков Н.В.				НУ «Запорізька політехніка»			
Затв.		Гудименко Р.К.				КНТ-517			