МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ВІДДАЛЕНОГО УПРАВЛІННЯ IP КАМЕРОЮ ЗА ДОПОМОГОЮ ESP8266»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1219

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

В.О. Ільченко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н., Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри прикладної математики та механіки, професор, д.т.н., Гищак В.З.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

Лісняк А.О.

(підпис)

« » 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Ільченко Владиславу Олеговичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка мобільного додатку управління IP камерою за допомогою ESP8266

керівник роботи (проекту) Мухін В.В., доцент кафедри програмної інженерії, к.т.н.

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

затверджені наказом ЗНУ від ««20» травня 2020 року № 576-с

2. Строк подання студентом роботи 30.11.2020 року

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Програмування плати ESP8266, розробка мобільного додатку для віддаленого управління IP камерою

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20.05.2020 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	06.09.2020	
2.	Збір вихідних даних.	17.09.2020	
3.	Обробка методичних та теоретичних	15.10.2020	
	джерел.		
4.	Розробка першого розділу.	25.10.2020	
5.	Розробка другого розділу.	19.11.2020	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	30.11.2020	
7.	Захист кваліфікаційної роботи.	15.12.2020	

Студент _____ В.О. Ільченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного додатку віддаленого управління IP камерою за допомогою ESP8266»: 71 с., 10 рис., 1 табл., 10 джерел, 3 додатки.

IP КаМеРа, ESP8266, МОБІЛЬНА РОЗРОБКА, ARDUINO, PYTHON, ESP32-CAM, FLASK, REACT NATIVE, JAVASCRIPT.

Об’єкт дослідження – плата ESP32-CAM, наступне покоління плати ESP8266.

Мета роботи: дослідити сучаний стан галузі Інтернету речей, дослідити плату ESP32-CAM, розробити проект камери спостереження з датчиком руху на базі плати ESP32-CAM.

Метод дослідження – аналітичний, методи програмної інженерії.

Плата ESP8266 є дуже популярна серед розробників через свою дешеву ціну і доволі багатий функціонал. Плата ESP32 є наступницею плати ESP8266 і має ряд покращення за майже такою ж ціною. В цій роботі буде досліджено плату, її програмування за допомогою Arduino. Як допоміжна плата буде використано китайську копію плати Arduino uno. Розроблений дуже простий сервер на Flask для отримання та передачі інформації з датчика руху, а також простий мобільний додаток написаний на React Native, що буде показувати статус з датчика руху і за бажанням трансляцію відео з камери. Усі ці компоненти є зменшеною копією основних компонентів сучасних модулів систем розумного будинку, галузі яка дуже швидко розвивається і проникає у багато галузей бізнесу. Тому тема є надзвичайно актуальною і цікавою.

SUMMARY

Master’s Qualification Thesis "Development of the Remote IP-camera Control Mobile Application using ESP8266": 71 pages, 10 figures, 1 tables, 10 sources, 3 applications.

IP CAMERA, ESP8266, MOBILE DEVELOPMENT, ARDUINO, PYTHON, ESP32 CAM, FLASK, REACT NATIVE, JAVASCRIPT.

The object of research is the ESP32-CAM board, the next generation of the ESP8266 board.

The aim of the study is to explore modern state of Internet of things industry, to explore the ESP32-CAM board, develop a surveillance camera project with a motion sensor based on ESP32-CAM board.

The methods of research are analytical, methods of program engineering.

The esp8266 board is very popular among developers due to its cheap price and quite rich functionality. The esp32 board is a successor to the esp8266 board and has a number of improvements at almost the same price. In this work the board and methods how to program it will be explored. A Chinese copy of the Arduino Uno board will be used as an auxiliary board. Developed a very simple server using Flask for receiving and transmitting information from the motion sensor, as well as a simple mobile application written in React Native, which will show the status of the motion sensor and optionally broadcast video from the camera. All these components are a common components of modern smart home systems modules, an industry that is developing very fast and is expanding on many industries. Therefore, the topic is extremely relevant and interesting.

ЗМІСТ

ВСТУП

Інтернет речей (англ. Internet of Things, IoT) — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку. Окрім датчики, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Через еволюційний розвиток технологій обладнання яке використовується для прикладних застосувань за галузі Інтернету речей стають все доступнішими, так плата що буде використовуватись у цій роботі зараз коштує приблизно 9$. Що відкриває можливість для бізнесу розробляти і впроваджувати різні проекти у галузі. Через це ринок Інтернету речей поступово зростає і наразі є дуже перспективним у світі. Окрім традиційних цілей датчики дозволяють збирати велику кількість інформації, що зараз можно обробляти методами і алгоритмами з науки про дані (Data science) та використовувати для передбачення та оптимізацію виробничих і навіть бізнес процесів.

Таким чином метою кваліфікаційної роботи являється розробка мобільного додатку віддаленого управління IP камерою за допомогою ESP8266. Для виконання цілі поставлені наступні задачі:

1. Дослідити основні компоненти системи Інтернету Речей.
2. Розглянути плати ESP32-CAM.
3. Розробити мобільний додаток для керування камерою.

Структурно робота складається дослідження сучасного стану галузі Інтернету речей, дослідження плати ESP8266 та її наступниці — ESP32-CAM, вибір технології для програмування плати, сервер ної частини, мобільного додатку, сбірка і тестування приладу і програмного забезпечення.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Інтернет речей

Основною концепцією Інтернету речей є можливість підключення всіляких об'єктів (речей), які людина може використовувати в повсякденному житті, наприклад, холодильник, кондиціонер, автомобіль, велосипед і навіть кросівки. Всі ці об'єкти (речі) повинні бути оснащені вбудованими давачами або сенсорами, які мають можливість обробляти інформацію, що надходить з навколишнього середовища, обмінюватися нею і виконувати різні дії в залежності від отриманої інформації. Прикладом впровадження такої концепції є система «розумний будинок» або «розумна ферма» [1]. Ця система аналізує дані навколишнього середовища і в залежності від показників регулює температуру в приміщенні. У зимовий період регулюються інтенсивність опалення, а в разі спекотної погоди будинок має механізми відкривання і закривання вікон, завдяки чому провітрюється будинок, і все це відбувається без втручання людини.

Для ідентифікації кожного об'єкту потрібна проста, компактна технологія. Тільки при наявності системи унікальної ідентифікації можна збирати та накопичувати інформацію про певний предмет. Такий функціонал можна забезпечити за допомогою мікросхем RFID (Radio-Frequency IDentification). Вони здатні без власного джерела струму передавати інформацію приладам зчитування. Кожна мікросхема має індивідуальний номер. Як альтернатива до даної технології для ідентифікації об'єктів можуть використовуватись QR-коди. Для визначення точного місця знаходження речі підійде технологія GPS, яка ефективно використовується вже сьогодні у смартфонах та навігаторах.

Для відслідковування змін у стані елементу чи оточуючого середовища об'єкти повинні оснащуватися сенсорами.

Для обробки та накопичення даних з сенсорів повинен використовуватися вбудований комп'ютер (наприклад Raspberry Pi, Intel Edison).

Для обміну інформацією між пристроями можуть бути використані технології бездротових мереж (Wi-Fi, Bluetooth, ZigBee, 6LoWPAN).

Інтеграція з Інтернетом має на увазі, що пристрої будуть використовувати IP-адресу як унікальний ідентифікатор. Проте, через обмежені адресні простори в IPv4 (що дозволяє використовувати 4,3 мільярда унікальних адрес), об'єктам IP доведеться використовувати IPv6, який забезпечує унікальними адресами мережевого рівня не менше 300 млн пристроїв на одного жителя Землі. Об'єктами в IP будуть не тільки пристрої із сенсорними можливостями, але також пристрої, які виконують дії (наприклад, лампочки або замки, якими керують через Інтернет). Значною мірою, майбутнє інтернету речей не буде можливим без підтримки IPv6, отже, глобальне впровадження IPv6 у найближчі роки буде мати вирішальне значення для успішного розвитку IP в майбутньому.

Для бездротової передачі даних особливо важливу роль в побудові інтернету речей відіграють такі характеристики, як ефективність, відмовостійкість, адаптивність, можливість самоорганізації. Основне зацікавлення в цьому сенсі представляє стандарт IEEE 802.15.4, що управляє доступом для організації енергоефективних персональних мереж, і є основою для таких протоколів, як ZigBee, WiFi, Bluetoot, 6LoWPAN.

ZigBee — це комунікаційна технологія, заснована на протоколі IEEE 802.15.4 для реалізації низькошвидкісних бездротових приватних мереж. ZigBee володіє такими характеристиками, як низьке енергоспоживання, низька швидкість передачі даних, низька вартість і висока пропускна здатність. В даний час ZigBee використовується в основному при передаванні

інформації між різними речами електронного обладнання, які знаходяться в межах короткої відстані і швидкості передачі даних не дуже висока. Це, в основному периферійні пристрої (миша, клавіатура) і побутова електроніка (TV, DVD), а також пристрої промислового управління (монітори, давачі і засоби автоматизації).

WiFi — це локальна бездротова технологія, яка використовує 2,4 ГГц надвисокої частоти або 5 ГГц супер-високочастотної радіохвилі. Ця технологія дуже добре підходить для передавання великих обсягів даних по бездротовій мережі між пристроями, але це також вимагає багато енергії для роботи і має невеликий рівень пропускної здатності даних. При використанні цієї технології потрібно буде замінювати батареї у всіх пристроях на регулярній основі.

Bluetooth — це бездротова технологія, яка використовується для передачі даних в персональних мережах. Він передає дані по смузі частот від 2,4 до 2,485 ГГц і працює на коротших відстанях, ніж Wi-Fi. Ви можете синхронізувати пару пристроїв, таких як телефони, навушники, колонки, комп'ютери і багато іншого. З розвитком Bluetooth v4.0 з'явилася можливість реалізувати функцію низького енергоспоживання і збільшений радіус дії до декількох десятків метрів.

Серед провідних технологій важливу роль у розповсюдженні інтернету речей відіграють рішення PLC — технології побудови мереж передачі даних по лініях електропередач, оскільки у багатьох додатках присутній доступ до електромереж (наприклад, торгові автомати, банкомати, інтелектуальні лічильники, контролери освітлення спочатку підключені до мережі електропостачання). 6LoWPAN, який реалізує шар IPv6 як над IEEE 802.15.4, так і над PLC, будучи відкритим протоколом, стандартизованих IETF, відзначається як особливо важливий для розвитку інтернету речей.

Вже зараз інтернету речей приділяється увага на найвищому рівні, зокрема починаючи з 2009 року у Брюселі при підтримці Єврокомісії проходять конференції Annual Internet of Things, на який виступають з доповідями єврокомісари, науковці та керівники провідних IT-компаній. За прогнозами аналітиків у найближчі роки очікується справжній бум інтернету речей. Так, за прогнозами Gartner, до 2020 року кількість підключених до всесвітньої мережі пристроїв становитиме 26 мільярдів, а дохід від продажу устаткування, програмного забезпечення та послуг становитиме 1,9 трлн дол [1]. Деякі інші аналітичні агентства висловлюють ще більш оптимістичні прогнози. Найбільші світові IT компанії вже почали перегони за лідерство на цьому ринку. Так корпорація Intel у 2014 році після випуску «SoC Edison» оголосила конкурс «Make it Wearable» з призовим фондом $1,3 млн на найкраще застосування своєї системи для концепції IoT та створила власний підрозділ «Internet of Things Solutions Group» для розвитку цього напрямку [1]. Компанія «Google» на початку 2014 року за 3,2 млрд доларів купила невелику фірму «Nest Labs», яка займається випуском інтелектуальних термостатів. Спеціалісти цієї компанії займались впровадженням на американському ринку технологій IoT. Виробники побутової техніки також працюють у цьому напрямку. Так на виставці CES 2014 у Лас-Вегасі була представлена велика кількість побутової техніки (холодильники, телевізори, пральні машини) з можливістю підключення до інтернет. Значення на ринку прогнозується на рівні 80 мільярдів доларів [1].

Лідерами у розробці та впровадженні інтернету речей є країни, в який розвинена індустрія виробництва мікропроцесорів та вбудованих комп'ютерів — це США, Китай, Південна Корея. Також значний прогрес у цій галузі демонструють європейські країни та Японія.

Розумний дім (розумний будинок/ smart home, digital house) — система домашніх пристроїв, здатних виконувати дії і вирішувати певні повсякденні

завдання без участі людини. Функціонально пов'язуються між собою усі електроприлади будівлі, якими можна керувати централізовано — з пульта-дисплею. Прилади можуть бути під'єднані до комп'ютерної мережі, що дозволяє керувати ними за допомогою ПК та надає віддалений доступ до них через Інтернет. Завдяки інтеграції інформаційних технологій у домашні умови, усі системи та прилади узгоджують виконання функцій між собою, порівнюючи задані програми та зовнішні показники (обстановки).

Для визначення високо-технологічних особливостей приміщення також вживають терміни: intelligent building, smart-house, digital home.

Розумний дім створюється за допомогою професійного проектування та програмування компаніями, що займаються розробкою проектів smart-home. Програми, що вводяться до алгоритмів multi-room розумного дому, розраховані на певні потреби мешканців та ситуації, пов'язані із зміною середовища або безпекою. Особливістю smart-home є керування з пульта, на котрому людина може натиснути одну-єдину клавішу з метою створення певної обстановки. При цьому, сама система мульти-рум аналізує навколишню ситуацію та параметри усередині приміщення, та, керуючись власними висновками, виконує задані користувачем команди із відповідними налаштуваннями [2]. Окрім того, електронні побутові прилади, встановлені у розумному будинку, можуть бути об'єднані у домашню Universal Plug'n'Play — мережу із виходом до інтернету.

Розумні будинки, як і більшість досягнень сучасної техніки, початково з'явилися на сторінках фантастичних оповідань [2]. Але матеріалізовуватись ідея почала лише у XX-му сторіччі після широкого введення електрики у будівлях і розвитку інформаційних технологій. Перше повідомлення про віддалені прилади контролю можна віднести до розробки Ніколою Тесла дистанційного керування судами та транспортними засобами у 1898 році [2].

електричні побутові прилади почали з'являтись у 1915-1920 роках і продемонстрували готовність суспільства замінити роботу домашнього персоналу дешевими механічними пристроями. Правда на той час, проблема енергозбереження при використанні нових технологій ще вирішена не була. Тому, певний час, новітні технологій були доступні лише дуже заможним людям.

Ідеї більш розвинені до понять сучасних систем автоматизації будинку були продемонстровані на ярмарках у Чикаго (1934) та Нью-Йорку. У «великому яблуці» трохи пізніше (1964-1965), представили плани електрофікованих та автоматизованих приміщень . У решті-решт перший серйозний аналог розумного дому з'явився у 1966 році [2]. Це була експериментальна система домашньої автоматизації — «домашній комп'ютер Эхо IV». Його винахідник — Джим Сазерленд, інженер компанії Westinghouse Electric. Його технологія була приватним, некомерційним проектом. Перші «дротові будинки» були зведені американськими винахідниками-любителями у 1960-х, але вони були суттєво обмежені можливостями тогочасних технологій.

Уперше термін «розумний будинок» був вигаданий американською асоціацією Housebuilders у 1984 році [2]. Із винаходом мікроконтролерів, вартість на електроприлади швидко падала. Ця ж установа зазначила, що таке помешкання відмінне від звичайного своєю здатністю забезпечувати продуктивне та ефективне використання робочого та житлового середовища. За цим, віддалені інтелектуальні технології керування були прийняті будівельною промисловістю, яка поступово почала вводити їх не лише у бізнес установах, але і у домашніх помешканнях. Під час активної домашньої автоматизації 90-х років інформатика та телевізійні системи були поєднані для підтримки інтелектуальних можливостей приміщень. У 1995 році

винахідники технологій Java оголосили одним із основних призначень даної технології — «збільшення інтелекту побутових приладів» [2].

Сьогодні технології дозволяють збирати домашню автоматику покомпонентно: обирати лише ті функції розумного будинку, які дійсно потрібні користувачу. Тепер новітні технології керування приміщенням з'являються щодня. Навіть речі, котрі раніше розглядалися лише як красиві предмети інтер'єру тепер можуть виконувати ряд мультимедійних або побутових функцій.

Нині вже йдеться про цілі «розумні» міста, в яких дані накопичуються і опрацьовуються в реальному часі, аби, приміром, регулювати транспортний рух, щоб не було заторів, керувати ефективніше водопостачанням і загалом економніше використовувати ресурси. Над фізичним світом буде прокладена віртуальна мережа, яка аналізує обсяг інформації, що надходить від «розумних» речей та сенсорів.

Якщо на Youtube задати пошук «Internet of Things», то на екрані з’являється відео від IBM. Там у найпривабливіших фарбах описується, як планета Земля розвине собі справжню нервову систему завдяки «інтернету речей». Уже роками IBM форсує дослідження в цьому напрямку. В одному проекті у бразильському Ріо-де-Жанейро ідеться, приміром, про програму запобігання катастрофам. Сенсори на землі та в повітрі мають бути пов'язані із системою штучного розуму, аби передбачати сильні зливи та зсуви ґрунтів за 48 годин до самої стихії. Таким чином залишатиметься ще достатньо часу для евакуації. Лише минулого року в Ріо внаслідок зсувів загинуло 70 людей [2].

У Німеччині також тривають інтенсивні дослідження. Серед іншого ці проекти підтримує міністерство економіки – загалом 11 проектів. А на нещодавньому галузевому саміті в Мюнхені було представлено «розумне» авто, яке може пересуватися без водія і достатньо «smart», аби тримати

відстань до інших машин, контролювати швидкість та у випадку заторів порекомендувати пересісти на потяг.

Але те, що для розробників видається чудовою ідеєю, захисники даних сприймають як страшний сон. Приміром, програма «smart meter». У ній ідеться про «розумні» кабелі енергопостачання, які здатні вирівнювати надмірне споживання електроенергії, регенеруючи енергію вітру та сонця. Ідея гарна, однак вимірюючи споживання енергії, програма «бачить» також, що коїться в квартирі. Приміром, які фільми або телепрограми переглядають. Тому захисники особистих даних громадян, зокрема члени неурядової організації «Європейські цифрові права», закликають науковців уже від самого початку подбати про захист даних, аби це питання ще в процесі розробки посідало одне з головних місць. Активісти вимагають укласти перелік правил інформаційного суспільства, аби цифрові ноу-хау потім не обернулися проти користувачів [11].

Система «розумний дім» допомагає більш результативно використовувати комерційні пересування, автоматизувати певні побутові процеси, урізноманітнити дозвілля. Попри те, що smart-home — дорога технологія, яка вимагає планування із самого початку зведення будинку та якісного устаткування, існують альтернативні рішення. Найпростіший за проектом дім можна доповнити певним прогресивним обладнанням, яке розширить функціональні можливості житлової площі та усучаснить пересування.

Наприклад вже тепер, за допомогою технологій інтелектуального будинку, піч може повідомити хазяїв, коли вона потребує чистки. А коли холодильнику стане необхідний техогляд, він «скаже» про це. Сигналізація може одночасно подзвонити на номери служби безпеки та хазяїна будинку, якщо у домі з'явився незваний гість. За допомогою налаштувань мульти-рум, будинок може визначити, хто із членів родини пересувається по

помешканню, і включити таке освітлення (температуру/музику тощо), яке влаштовує саме цю людину. Або наприклад «розумний замок» використовує з'єднання Bluetooth, щоб зафіксувати, коли людина та її смартфон залишаєте приміщення. Користувач може надати право доступу друзям та членам сім'ї за допомогою спеціально згенерованого ключа. Кожного разу, коли хто-небудь відкриває двері, власник буде отримувати повідомлення на телефон [2].

Більш складні системи можуть вести облік продукції у комерційних закладах, облік її використання через зчитування штрих-кодів або RFID-тег. А у домашньому використанні: готувати список покупок.

Сучасні мобільні пристрої, забезпечені акселерометрами, мікрофонами, камерами, всілякими датчиками, які можуть забезпечити потік даних, що однозначно і чітко описує все, що відбувається в навколишньому середовищі. І залишається тільки розробити досить складні і потужні універсальні програмні алгоритми, які здатні інтерпретувати цей потік даних, зробити висновки, прийняти відповідні рішення і виконати необхідні дії.

Різновиди IoT-систем.

Загалом, усі екосистеми бувають двох типів: масові та критичні.

Масові — це такі, де один сервер приймає багато запитів від великої кількості пристроїв, обробляє їх та аналізує. Масовий IoT вже навколо нас — система Eway відстежує рух громадського транспорту в містах України, Нова пошта моніторить посилки і дає вам інформацію про доставку, а MiBand аналізує ваш сон і порівнює його з результатами інших користувачів. Масові системи зазвичай дешеві та не споживають багато електроенергії.

Не менш важливу роль у нашому житті відіграють критичні екосистеми IoT. Їхнє завдання — оперативно передавати інформацію через надійну стійку мережу, бо від цього часто залежить людське життя. Наприклад — автопілот Tesla. Він працює завдяки чотирьом елементам, які обробляє

центральний комп’ютер. Фронтальний радар дозволяє системі «бачити» крізь дощ чи туман. 12 невеличких ультразвукових сенсорів, розташованих по всій машині, допомагають автопілоту розуміти, де і на якій відстані перебувають машини навколо [2]. Фронтальна відеокамера розпізнає та фіксує дорожні позначки, а надточний GPS-трекер, який водночас є й контролером, відстежує та аналізує дії всієї системи.

Проблема безпеки в Інтернеті речей.

Найбільшою проблемою IoT є захищеність системи. У інтернету речей дуже низький рівень безпеки. Пристрої не мають жодних антивірусів або навіть систем ідентифікації користувача. А коли вони ще й під’єднуються до інтернету, хакери спокійно можуть викрасти будь-яку інформацію. Дослідники з Массачусетського технологічного інституту (MIT) навіть писали про «ботнет речей».

Ботнет (англ. botnet від robot і network) — це комп'ютерна мережа, що складається з деякої кількості хостів, із запущеними ботами — автономним програмним забезпеченням. Найчастіше бот у складі ботнета є програмою, яка приховано встановлюється на комп'ютері жертви і дозволяє зловмисникові виконувати певні дії з використанням ресурсів інфікованого комп'ютера. Зазвичай використовуються для протиправної діяльності — розсилки спаму, перебору паролів на віддаленій системі, атак на відмову в обслуговуванні, отримання персональної інформації про користувачів, крадіжка номерів кредитних карток та паролів доступу. Кожен комп'ютер в мережі діє як «бот» і управляється шахраєм для передачі шкідливих програм або шкідливого контенту для запуску атаки. Ботнет деколи називають «армією зомбі», так як комп'ютери контролюються кимось іншим, крім їх власника.

Залучення комп'ютерів до ботнету.

Комп'ютер може потрапити в мережу ботнету через встановлення певного програмного забезпечення, без відома користувача. Трапляється це зазвичай через:

1. Інфікування комп'ютера вірусом через вразливість в ПЗ (помилки в браузерах, поштових клієнтах, програмах перегляду документів, зображень, відео).
2. Недосвідченість або неуважність користувача — шкідливе ПЗ маскується під «корисне програмне забезпечення».
3. Використання санкціонованого доступу до комп'ютера (рідко).
4. Підбір адміністративного пароля до мережевих ресурсів зі спільним доступом (наприклад, до $ADMIN, що дозволяє віддалено виконати програму) — переважно в локальних мережах.

Механізм маскування.

Механізм захисту від видалення аналогічний більшості вірусів та руткітів, зокрема:

1) маскування під системний процес;
2) підміна системних файлів для самомаскування;
3) інжекція коду безпосередньо в адресний простір системного процесу або процесу користувача;
4) перехоплення системних викликів для маскування наявності в системі файлів ботнету та посилань на нього;
5) перехоплення системних процедур роботи з мережею для маскування трафіку ботнету під трафік користувача або системних утиліт;
6) використання поліморфного коду, що ускладнює сигнатурний аналіз;
7) маскування під корисне ПЗ (прискорювачі Інтернет, програми для завантаження на диск онлайн-відео та -аудіо та ін.).

Механізм самозахисту.

До механізмів самозахисту відносять:

1) створення перешкод нормальній роботі антивірусного ПЗ;
2) перезавантаження комп'ютера та інші порушення нормальної роботи при спробі доступу до виконуваних файлів або ключів автозапуска, в яких прописані файли програмного забезпечення ботнету.

Механізм автозапуску.

Для автозапуску найчастіше використовуються наступні технології:

1) використання нестандартних методів запуску (використовуються шляхи автозапуску від старого програмного забезпечення, підміна налагоджувальника процесів);
2) використання двох процесів які перезапускають один одного, у випадку зняття одного з цих процесів інший процес знову його запустить;
3) підміна системних файлів, що автоматично завантажуються операційною системою;
4) Реєстрація в ключах автозапуску або в списку модулів розширення функціональності системи.

Механізм керування ботнетом.

Раніше керування передбачало «очікування» певних команд по певному порту, або участь в IRC-чаті. При відсутності команд програма «спить» очікуючи на команду власника, можливо намагається саморозмножуватись. При отриманні команди від «власника» ботнету, починає виконувати вказану команду. В ряді випадків за командою завантажується виконуваний файл (таким чином, є можливість «оновлювати» програму і завантажувати модулі які додають функціональність).

Наразі отримали поширення ботнети які керуються через веб-сайт або по принципу p2p-мереж.

1.2 Плати ESP8266, ESP32, ESP32-CAM

ESP8266 — мікроконтролер китайського виробника Espressif з інтерфейсом Wi-Fi. Окрім Wi-Fi, мікроконтролер здатен виконувати програми з зовнішньої флеш-пам'яті з інтерфейсом SPI. Мікроконтролер привернув увагу в 2014 році у зв'язку з виходом перших продуктів на його базі за неочікувано низькою ціною. Навесні 2016 року почалося виробництво ESP8285, що об'єднує ESP8266 та флеш пам'ять на 1 Мбайт. Восени 2015 року Espressif запропонувала вдосконалену модель лінійки — мікросхему ESP32 [3].

Мікроконтролер характеризується:

1) 80 MHz 32-bit процесор Tensilica Xtensa L106. Можливий негарантований розгін до 160 Мгц;
2) IEEE 802.11 b/g/n Wi-Fi. Підтримується WEP та WPA/WPA2;
3) 14 портів вводу-виводу (з них можливо використовувати 11), SPI, I²C, I²S, UART, 10-bit АЦП;
4) живлення 2,2…3,6 В. Споживання до 215 ма в режимі передачі, 100 ма в режимі прийому, 70 ма в режимі очікування. Підтримуються три режими зниженого живлення, все без зберігання з'єднання з точкою доступу: Modem sleep (15 ма), Light sleep (0.4 ма), Deep sleep (15 мка).

Мікроконтролер не має на кристалі енергонезалежної пам'яті для користувача [3]. Виконання програми ведеться із зовнішнього SPI ПЗП шляхом динамічного завантаження необхідних проміжків програми в КеШ інструкцій. Завантаження виконується апаратно, прозоро для програміста. Підтримується до 16 МБ зовнішньої пам'яті програм. Можливий Standard, Dual або Quad SPI інтерфейс. Виробник не надає документації на внутрішню периферію контролеру. Замість цього він надає набір бібліотек, через API

яких програміст отримує доступ до периферії. Так як ці бібліотеки інтенсивно використовують ОЗП контролера, то виробник у документах не вказує точну кількість оЗП на кристалі, а надає лише приблизну оцінку тої кількості пам'яті, що залишається після лінкування бібліотек — близько 50 кБ. Ентузіасти, що дослідили бібліотеки ESP8266, припускають, що він має 32 кБ кешу інструкцій та 80 кБ оЗП даних [3].

Електричні параметри, цокольовки, схеми включення можна знайти в документах «0A-ESP8266EX__Datasheet» та «0B-ESP8266__System_Description» з Espressif SDK.

ESP32 — це серія мікроконтролерів типу «система на кристалі», що мають інтегровані контролери Wi-Fi і Bluetooth (дворежимний, англ. dual-mode), низьке енергоспоживання і невисоку ціну. У серії ESP32 використовується мікропроцесор Tensilica Xtensa LX6 в двоядерних та одноядерних варіаціях та включає вбудовані антенні перемикачі, підсилювач потужності, приймач з низьким рівнем шумів, фільтри та модулі керування живленням [3]. ESP32 створений та розроблений компанією Espressif Systems, китайською компанією, розташованою у Шанхаї, а виробляється компанією TSMC. Він є наступником мікроконтролера ESP8266 [4].

особливості ESP32 включають в себе наступне [5]:

- процесор: Xtensa двоядерний (або одноядерної) 32-розрядний LX6 мікропроцесор, що працює на 160 або 240 МГц і виконує до 600 DMIPS;
- ультра низька потужність (оТП) співпроцесор;
- пам'ять: 520 Кб пам'яті SRAM;
- Wi-Fi: 802.11 b/g/N;
- Bluetooth: В4.2 БР/EDR і БЛЕ;
- 12-розрядний аЦП до 18 каналів;
- 2 × 8-біт Цапи;

- 10 × сенсорних датчиків (ємнісних датчиків і контролерів);
- Датчик температури;
- 4 × Сво;
- 2 × i2s для інтерфейсів;
- 2 × з I2C інтерфейси;
- 3 × UART;
- SD/SDIO/CE-ATA/MMC/eMMC хост-контролер;
- SDIO/SPI підпорядкований контролер;
- Ethernet Mac інтерфейс з виділеними DMA і стандарти IEEE 1588 точного часу за протоколом підтримки;
- CAN bus 2.0;
- інфрачервоний пульт дистанційного управління (передавач/приймач, до 8 каналів);
- можливість підключення двигунів та світлодіодів через ШІМ-вихід;
- ультра низька потужність аналоговий передпідсилювач;
- стандарт IEEE 802.11 підтримує всі функції безпеки, у тому числі АБФ, захист WPA/WPA2 і ВАПІ;
- безпечне завантаження;
- шифрування флеш;
- 1024-бітний ключ, до 768 біт для клієнтів;
- криптографічне апаратне прискорення: AES, SHA-2, RSA, криптографії на основі еліптичних кривих (eCC), генератор випадкових чисел (ГВЧ);
- управління живленням;
- внутрішній низький регулятор відключення;
- індивідуальний енергетичний домен для RTC;
- 5 мка струм режиму "глибокий сон";

- прокидання з переривання від GPIO, таймера, вимірювання аЦП, переривання ємнісного сенсорного датчика.

Мови програмування, платформи та середовища, що використовуються для програмування ESP32 [5]:

- Arduino IDE з ESP32 Arduino Core;
- Espressif IoT Development Framework — офіційна Espressif розробка для ESP32;
- Espruino — JavaScript SDK і прошивка майже замінює Node.js;
- Lua RTOS дляESP32;
- Mongoose OS — операційна система для підключених продуктів на мікроконтролерах;
- PlatformIO Ecosystem і IDE;
- Pymakr IDE — IDE призначений для використання з пристроями Pycom;
- Simba Embedded Programming Platform;
- Whitecat Ecosystem Blockly заснована на Web IDE;
- MicroPython;
- Zerynth — Python для IoT і мікроконтролерів, включаючи ESP32.

Серед явних переваг є невелика ціна, слот для SD карт, можливість досліджувати розпізнавання облич.

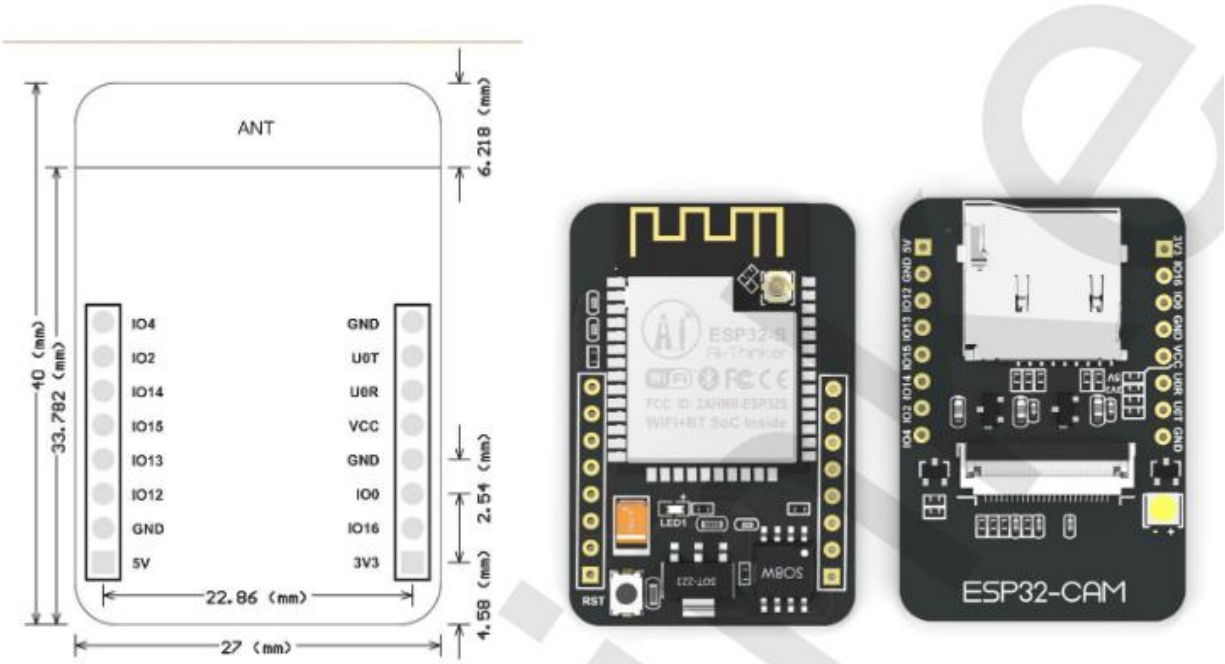

Рисунок 1.1 — Схема плати ESP32-CAM

Інформація про модуль [5]:

- найменший модуль SoC Wi-Fi BT 802.11b/g/n;
- 32-розрядний процесор низької потужності, може також обслуговувати процесор додатків;
- тактова частота до 160 МГц ' Підсумкова обчислювальна потужність до 600 DMIPS;
- вбудований 520 KB SRAM, зовнішня 4MPSRAM;
- підтримка UART / SPI / I2C / PWM / ADC / DAC;
- підтримка камер OV2640 і OV7670, вбудована лампа Flash;
- підтримка завантаження зображення WiFI;
- підтримка карт TF;
- підтримка декількох режимів сну;
- вбудовані Lwip і FreeRTOS;
- підтримка STA / AP / STA + Режим роботи точки доступу;
- підтримка технології Smart Config / AirKiss;

- підтримка локального та віддаленого оновлення мікропрограми послідовного порту (FOTA).

В цій роботі буде використано Arduino IDE з ESP32 Arduino Core через те що це дуже популярне рішення і має гарну документацію та базу прикладів у Інтернеті, а отже і перевірений багатьма людьми шлях. ESP32-CAM камера на базі ESP32 з можливістю програмування з середовища Arduino IDE.

Таблиця 1.1 — Таблиця пінів плати ESP32-CAM

Cam	ESP32	SD	ESP32
D0	PIN5	CLK	PIN14
D1	PIN18	CMD	PIN15
D2	PIN19	DATA0	PIN2
D3	PIN21	DATA1/Flash lamp	PIN4
D4	PIN36	DATA2	PIN12
D5	PIN39	DATA3	PIN13
D6	PIN34		
D7	PIN35		
XCLK	PIN0		
PCLK	PIN22		
VSYNC	PIN25		
HREF	PIN23		
SDA	PIN26		
SCL	PIN27		
POWER PIN	PIN32		

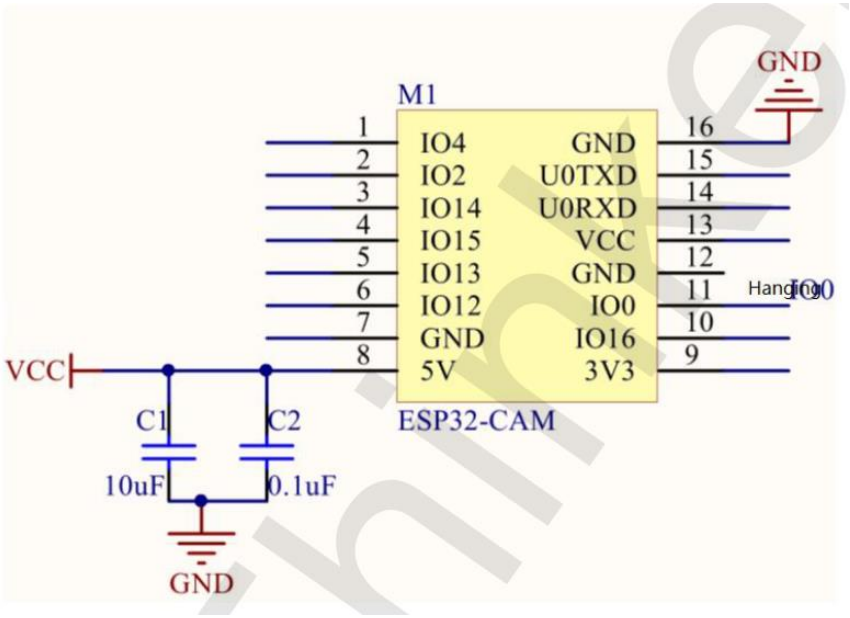

Рисунок 1.2 — Схема виводів плати ESP32-CAM

Окрім ESP32 CAM буде використана плата Arduino Uno, тому що ESP32 CAM не має USB інтерфейсу чи іншого гарного інтерфейсу для програмування за замовчуванням, інша популярна опція працюючи з ESP32 CAM є FTDI програматор, він дуже дешевий і простий.

Рисунок 1.3 — Плата Arduino UNO

1.3 Огляд методів програмування плати

Існує багато способів програмування плати ESP32:

- Arduino IDE з ESP32 Arduino Core;
- Espressif IoT Framework - офіційна розробка Espressif для ESP32;
- Espruino - JavaScript SDK, емулятор Node.js;
- RTV Lua;
- Mongoose OS - операційна система для несучих електроніків, рекомендована Espressif Systems, AWS IoT, та Google Cloud IoT;
- mruby для ESP32;
- екосистема PlatformIO і IDE;
- Pymakr IDE - IDE призначений для використання з пристроями Pycom;
- Вбудована платформа програмування Simba;
- екосистема Whitecat блоковано заснована на Web IDE;
- MicroPython;
- Zerynth - Python для IoT та мікроконтролерів, включаючи ESP32;
- OWLOS - сетева операційна система з відкритим вихідним кодом для управління пристроями IoT.

Найбільш популярними є програмування на C та мові ассемблеру, використовуючи SDK від фірми виробника Espressif та Arduino [4]. Для розробки цього проекту було обрано Arduino тому що воно має готові шаблони коду для керування основними функціями плати, що робить програмування значно простішим а також потребує значно менше налаштувань.

Arduino (ардуіно) — апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки

Processing/Wiring на мові програмування, що є спрощеною підмножиною C/C++. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Processing, Adobe Flash, Max/MSP, Pure Data, SuperCollider) [4]. Інформація про плату (рисунок друкованої плати, специфікації елементів, програмне забезпечення) знаходяться у відкритому доступі і можуть бути використані тими, хто воліє створювати плати власноруч.

Інтегроване середовище розробки Arduino це багатоплатформовий додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату. Середовище розробки засноване на мові програмування Processing та спроектоване для програмування новачками, не знайомими близько з розробкою програмного забезпечення. Мова програмування аналогічна мові Wiring. Загалом, це C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC [6].

Програми Arduino пишуться на мові програмування C або C++. Середовище розробки Arduino поставляється разом із бібліотекою програм «Wiring» (бере початок від проекту Wiring, який дозволяє робити багато стандартних операцій вводу/виводу набагато простіше). Користувачам необхідно визначити лише дві функції для того, щоб створити програму, яка буде працювати за принципом циклічного виконання:

- setup(): функція виконується лише раз при старті програми і дозволяє задати початкові параметри;
- loop(): функція виконується періодично, доки плата не буде вимкнена.

1.4 Вибір технологій для серверної частини проекту

На даний момент існує дуже багато як відкритих так і закритих засобів для програмування веб серверів. При виборі технології було поставлено такі вимоги: технологія має буди безкоштовною і простою в застосуванні.
На даний момент популярною мовою веб розробки є Python. Мова Python є досить проста і не перевантажена додатковими модулями (за замовченням) а також є безкоштовною і має сервер що можна швидко запустити і використовувати — Flask. Тож вона повністю задовольнила нашим вимогам.

Python (найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [7].

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Розробка мови Python була розпочата в кінці 1980-х років співробітником голландського інституту CWI Гвідо ван Россумом [7]. Для розподіленої оС Amoeba потрібна була розширювана скриптова мова, і Гвідо почав писати Python на дозвіллі, запозичивши деякі напрацювання для мови ABC (Гвідо брав участь у розробці цієї мови, орієнтованої на навчання програмування). У лютому 1991 року Гвідо опублікував вихідний текст в

групі новин alt.sources. Мова почала вільно поширюватися через Інтернет і сподобалася іншим програмістам. З 1991 року Python є цілком об'єктно-орієнтованим. Python також запозичив багато рис таких мов, як C, C++, Modula-3 і Icon, й окремі риси функціонального програмування з Ліспу [7].
Назва мови виникла зовсім не від виду плазунів. Автор назвав мову на честь популярного британського комедійного серіалу 70-х років «Повітряний цирк Монті Пайтона». Втім, все одно назву мови частіше асоціюють саме зі змією, ніж з фільмом — піктограми файлів в KDE або в Windows, і навіть емблема на сайті python.org зображують зміїну голову [7].

Наявність дружньої спільноти користувачів, поряд з дизайнерською інтуїцією Гвідо, вважається одним з головних факторів успіху Python. Розвиток мови відбувається згідно з чітко регламентованими процесами створення, обговорення, відбору та реалізації документів PEP (Python Enhancement Proposal) — пропозицій щодо розвитку Python.

3 грудня 2008 року, після тривалого тестування, вийшла перша версія Python 3000 (або Python 3.0, також використовується скорочена Py3k). У Python 3000 усунено багато недоліків архітектури з максимально можливим (але не повним) збереженням сумісності зі старішими версіями. На сьогодні підтримуються Python версії 3.

Python портований і працює майже на всіх відомих платформах — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android.

У міру старіння платформи її підтримка в основній гілці мови припиняється. Наприклад, з версії 2.6 припинена підтримка Windows 95, Windows 98 та Windows ME. Однак на цих платформах можна

використовувати попередні версії Python — спільнота активно підтримує версії Python починаючи від 2.3 (для них виходять виправлення).

При цьому, на відміну від багатьох портованих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Навіть більше, існує спеціальна версія Python для віртуальної машини Java — Jython [7], що дозволяє інтерпретатору виконуватися на будь-якій системі, яка підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на ньому. Також кілька проектів забезпечують інтеграцію з платформою Microsoft.NET, основні з яких — IronPython та Python.Net.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді. З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

Інтерактивний режим

Подібно Ліспу та Прологу в режимі відлагодження, інтерпретатор Python має інтерактивний режим роботи, при якому введені з клавіатури вирази відразу ж виконуються, а результат виводиться на екран. Цей режим цікавий не тільки новачкам, але й досвідченим програмістам, які можуть протестувати в інтерактивному режимі будь-який фрагмент коду, перш ніж використовувати його в основній програмі, або просто використовувати як калькулятор з великим набором функцій.

Так виглядає діалог роботи з Python в інтерактивному режимі:

```
>>> 2 ** 100 # піднесення 2 до 100-го степеня
1267650600228229401496703205376L
>>> from math import * # імпорт математичних функцій
>>> sin (pi * 0.5) # обчислення синуса від половини пі
1.0
>>> help (sorted) # допомогу по функції sorted
Help on built-in function sorted in module __builtin__:
sorted (…)
  sorted (iterable, cmp=none, key=none, reverse=false) -> new sorted list
```

В інтерактивному режимі доступний дебагер pdb та система довідки (викликається за help()). Система допомоги працює для модулів, класів і функцій, тільки якщо ті були забезпечені рядками документації.

Крім вбудованої, існує й покращена інтерактивна оболонка IPython.

Об'єктно-орієнтоване програмування.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами [7].

Можливості та особливості:

- класи є одночасно об'єктами з усіма нижче наведеними можливостями;
- успадкування, в тому числі множинне;
- поліморфізм (всі функції віртуальні);
- інкапсуляція (два рівні — загальнодоступні та приховані методи і поля). Особливість — приховані члени доступні для використання та помічені як приховані лише особливими іменами;

- спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті;
- перевантаження операторів (усіх, крім is, '.', '=' і символьних логічних);
- властивості (імітація поля за допомогою функцій);
- управління доступу до полів (емуляція полів і методів, частковий доступ тощо);
- методи для управління найпоширенішими операціями (істиннісне значення, len(), глибоке копіювання, серіалізація, ітерація по об'єкту, …);
- метапрограмування (управління створенням класів, тригери на створення класів, та ін);
- повна інтроспекція;
- класові та статичні методи, класові поля;
- класи, вкладені у функції та інші класи.

Функціональне програмування.

Python підтримує парадигму функціонального програмування, зокрема:

- функція є об'єктом;
- функції вищих порядків;
- рекурсія;
- розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори);
- аналог замикань (closures);
- часткове застосування функції;
- можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Модулі та пакети.

Програмне забезпечення (застосунок або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакунки. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на Ci: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет — у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором import. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. У ході виконання програми модуль можна перезавантажити функцією reload().

Інтроспекція.

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Застосування інтроспекції (метапрограмування) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і фреймворках Python, таких як PyRO, Pyro, PLY, CherryPy, Django та інших, заощаджуючи час програміста, що ними користується.

Стандартна бібліотека.

Python поставляється «з батарейками в комплекті». Багата стандартна бібліотека є однією з привабливостей мови Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційною системою дозволяє писати крос-платформні застосунки. Існують модулі для роботи з регулярними виразами, текстовими

кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, юніт-тестуванням та ін.

Модулі розширення та програмні інтерфейси

Крім стандартної бібліотеки існує багато інших, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Існує велика кількість прикладних бібліотек для Python у різноманітних галузях: веб-розробка, бази даних, обробка зображень, обробка тексту, чисельні методи, програми операційної системи тощо.

Для Python прийнята специфікація програмного інтерфейсу до баз даних DB-API 2 та розроблено відповідні цій специфікації пакети для доступу до різних СУБД: PostgreSQL, Oracle, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server, MySQL та sqlite. На платформі Microsoft Windows доступ до БД можливий через ADO (ADOdb). Комерційний пакет mxODBC для доступу до СУБД через ODBC для платформ Windows і UNIX розроблений eGenix. Для Python написано багато ORM: (SQLObject, SQLAlchemy, Dejavu, Django), виконані програмні каркаси для розробки веб-застосунків (Django, Pylons).

Бібліотека NumPy для роботи з багатовимірними масивами дозволяє досягти продуктивності наукових розрахунків, порівнянної зі спеціалізованими пакетами. SciPy використовує NumPy і надає доступ до великого спектра математичних алгоритмів (матрична алгебра — BLAS, level 1-3 і LAPACK; ШПФ).

Бібліотека WSGI — інтерфейс шлюзу з веб-сервером (Python Web Server Gateway Interface).

Python надає простий і зручний програмний інтерфейс C API для написання власних модулів на мовах C та C++. Інструмент SWIG дозволяє

майже автоматично отримувати прив'язки для використання C/C++ бібліотек у коді на Python. Можливості цього та інших інструментів варіюються від автоматичної генерації (C/C++/Fortran)-Python інтерфейсів за спеціальними файлами (SWIG, pyste, SIP, pyfort) до надання зручніших API (boost::python, CXX та ін.) Інструмент стандартної бібліотеки ctypes дозволяє програмам Python безпосередньо викликати функції з динамічних бібліотек/DLL, написаних на C. Існують модулі, що дозволяють вбудовувати код на C/C++ прямо у вихідні файли Python, створюючи розширення «на льоту» (pyinline, weave). Для підключення математичних функцій, особливо із застосуванням NumPy, наразі офіційно рекомендованим є Cython.

Інший підхід полягає у вбудовуванні інтерпретатора Python у застосунки. Python легко вбудовується в програми на Java, C/C++, Ocaml. Взаємодія Python-застосунків з іншими системами можлива також за допомогою CORBA, XML-RPC, SOAP, COM.

За допомогою Pyrex можлива компіляція Python-подібної мови (додано можливість типізації) в еквівалентний Сі-код і зв'язування із зовнішніми модулями.

Експериментальний проект shed skin передбачає створення компілятора для трансформації неявно типізованих Python програм в оптимізований C++ код. Починаючи з версії 0.22 shed skin дозволяє компілювати окремі функції в модулі розширень. Повна компіляція (станом на 1 липня 2007) далека від завершення.

Python та переважна більшість бібліотек до нього безкоштовні й поставляються у вихідних кодах. Навіть більше, на відміну від багатьох відкритих систем, ліцензія ніяк не обмежує використання Python у комерційних розробках та не накладає ніяких зобов'язань, крім зазначення авторських прав.

Flask — мікрофреймворк для веб-додатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD.

Станом на грудень 2016 року стабільна версія Flask має номер 0.12. Flask використовується для розробки таких проектів як Pinterest, LinkedIn, а також сторінка спільноти Flask.

Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які забезпечують додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. Розширення оновлюються частіше аніж базовий код [8].

Основні властивості Flask [8]:

- містить сервер для розробки та відлагоджувач;
- вбудована підтримка юніт-тестів;
- управління запитами RESTful;
- використовує шаблони Jinja2;
- має підтримку безпечних куків (сесії на стороні клієнта);
- 100% відповідність WSGI 1.0;
- підтримка Unicode;
- докладна документація;
- сумісність з Google App Engine;
- наявність розширень для забезпечення бажаної поведінки.

1.5 Вибір технології для мобільної розробки

Для моніторингу камери одразу було вирішено розробити мобільний додаток, можна булоб розробити веб сайт, але в сучасному світі дуже популярним є мобільні додатки для керуванням пристроями розумного будинку, і зрозуміло чому — це дуже зручно, вони дійсно завжди з користувачем, не потрібно виконувати зайві дії по переходу на сайт у браузері.

Так як мобільний додаток задумувався з мінімумом функціоналу, без використання нативних бібліотек що є залежними від платформи — одразу з’явилась ідея використовувати фреймворк для кросс платформної розробки, щоб написавши один раз код можна було б запускати на телефонах Android та iOS.

Дуже популярними фреймворками є Ionic та React Native. Обидва використовують JavaScript як мову програмування. Тож вибір пав на них.
Ionic - це повний SDK з відкритим кодом для розробки гібридних мобільних додатків, створений Максом Лінчем, Беном Сперрі та адамом Бредлі з Drifty Co. у 2013 році. Оригінальна версія була випущена в 2013 році і побудована на базі AngularJS та Apache Cordova [9]. Однак останній випуск був перероблений як набір веб-компонентів, що дозволило користувачеві вибрати будь-яку структуру інтерфейсу користувача, наприклад Angular, React або Vue.js [9]. Це також дозволяє використовувати іонічні компоненти, які взагалі не мають інтерфейсу користувача. Ionic надає інструменти та послуги для розробки гібридних мобільних, настільних та прогресивних веб-програм, заснованих на сучасних технологіях та практиці веб-розробки, використовуючи веб-технології, такі як CSS, HTML5 та Sass. Зокрема, мобільні програми можна створювати за допомогою цих веб-технологій, а

потім розповсюджувати їх у власних магазинах додатків, щоб встановлювати на пристрої за допомогою Cordova або конденсатора [9].

Ionic використовує плагіни Cordova та нещодавно Capacitor, щоб отримати доступ до таких функцій операційної системи, як камера, GPS, ліхтарик тощо Користувачі можуть створювати свої програми, а потім їх можна налаштувати для Android, iOS, Windows, Desktop (з Electron) або сучасних браузерів. Ionic дозволяє створювати та розгортати програми, обгортаючи інструмент побудови Cordova або Capacitor за допомогою спрощеного інструменту командного рядка "ionic".

Ionic включає мобільні компоненти, типографіку, інтерактивні парадигми та розширювану базову тему.

Використовуючи веб-компоненти, Ionic надає власні компоненти та методи взаємодії з ними. Один з таких компонентів, віртуальна прокрутка, дозволяє користувачам прокручувати список тисяч елементів без будь-яких звернень продуктивності. Інший компонент, вкладки, створює інтерфейс із вкладками з підтримкою навігації у стилі рідного стилю та управління станом історії.

Окрім SDK, Ionic також надає послуги, які розробники можуть використовувати для включення таких функцій, як розгортання коду, автоматизовані збірки. Ionic також пропонує власну IDE, відому як Ionic Studio.

Ionic також надає інтерфейс командного рядка (CLI) для створення проектів. CLI також дозволяє розробникам додавати плагіни Cordova та додаткові інтерфейсні пакети, увімкнути push-сповіщення, генерувати піктограми програм та заставки та створювати власні двійкові файли.

React Native - це фреймворк мобільних додатків з відкритим кодом, створений Facebook, Inc. Він використовується для розробки програм для Android, Android TV, iOS, macOS, tvOS, Web, Windows та UWP, дозволяючи

розробникам використовувати React's фреймворк разом із можливостями власної платформи

Принципи роботи React Native практично ідентичні React, за винятком того, що React Native не маніпулює DOM через віртуальний DOM. Він працює у фоновому процесі (який інтерпретує написаний розробниками JavaScript) безпосередньо на кінцевому пристрої та взаємодіє з власною платформою через серіалізаційний, асинхронний та пакетний міст.

Компоненти React обгортають наявний власний код та взаємодіють із власними API через декларативну парадигму інтерфейсу користувача React та JavaScript. Це дозволяє розробляти власні програми для цілих нових команд розробників і може дозволити існуючим власним командам працювати набагато швидше [10].

React Native не використовує HTML або CSS. Натомість повідомлення з потоку JavaScript використовуються для маніпулювання власними поданнями. React Native також дозволяє розробникам писати власний код такими мовами, як Java для Android та Objective-C або Swift для iOS, що робить його ще більш гнучким.

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація,

автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ооП. Крім того, JavaScript має кілька властивостей, притаманних функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою С має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні функції.

JavaScript містить декілька вбудованих об'єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, грубо кажучи, не обов'язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від С), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї.

1995 року компанія Netscape поставила завдання вбудувати мову програмування Scheme чи «якусь схожу» в браузер Netscape. Для цього був запрошений Брендан айк, американський розробник, що спеціалізувався на системному програмуванні. Також, для прискорення розробки, Netscape почали співробітництво з компанією Sun Microsystems.

З часом, концепція розроблюваної мови програмування була розширена до можливості використання безпосередньо в HTML-коді сторінки. Компанії мали на меті створити мову, що могла зв'язати різні частини вебсайтів: зображень, Java-аплетів, об'єктної моделі документа. Ця мова повинна була стати зручною для вебдизайнерів та некваліфікованих програмістів. Робочою назвою нової мови була Mocha, яка була змінена на LiveScript в перших двох бета-версіях браузера Netscape 2.0. А дещо пізніше, користуючись популярністю бренду Java, LiveScript був перейменований на JavaScript і третя бета-версія (2.0B3) Netscape 2.0 вже вийшла з сучасною назвою. Для цього була придбана відповідна ліцензія у компанії Sun Microsystems, що володіла брендом Java.

1992 року компанією Nombas була розроблена скриптова мова програмування Cmm (англ. C-minus-minus, гра слів навколо мови C++), яка пізніше була перейменована на ScriptEase та могла вбудовуватися в вебсторінки. Існує хибна думка, що JavaScript створено під впливом Cmm. Насправді, Брендан айк ніколи не чув про Cmm до того, як він створив LiveScript. Пізніше, Nombas зупинили розробку Cmm та почали використовувати JavaScript, а згодом брали участь у групі зі стандартизації JavaScript.

У листопаді 1996 року Netscape заявила, що відправила JavaScript в організацію Ecma International для розгляду мови як промислового стандарту. В результаті подальшої роботи з'явилась стандартизована мова з назвою ECMAScript. У червні 1997 року, Ecma International опублікувала першу

редакцію специфікації ECMA-262. Рік по тому, у червні 1998 року, щоб адаптувати специфікацію до стандарту ISO/IEC-16262, були внесені деякі зміни і випущена друга редакція. Третя редакція побачила світ в грудні 1999 року.

Четверта версія стандарту ECMAScript так і не була закінчена і четверта редакція не вийшла. Тим не менш, п'ята редакція з'явилася в грудні 2009 року.

У червні 2015 року вийшла шоста версія, починаючи з якої комітет ECMAScript прийняв рішення перейти на щорічні оновлення і нова версія отримала назву ES2015. Вона отримала цілу низку нововведень, серед яких: об'єкт Promise для зручного асинхронного виконування коду, деструктуруюче присвоювання, стрілочні функції, функції-генератори, шаблонні рядки, оператори оголошення змінних let та const тощо.

Версія ES2016 вийшла у червні 2016 року, серед нововведень оператор піднесення до степеня ** та метод Array.prototype.includes, який перевіряє, чи міститься переданий аргумент в масиві.

Версія ES2017, що вийшла в червні 2017 року і на сьогодні є актуальною версією стандарту додала можливість використання асихронних функцій, «висячих» ком в параметрах функцій, об'єкт Atomics, декількох нових методів для роботи з рядками.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови, а подальші модифікації мови за стандартами ES2015 та ES2017 внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та

налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

Спеціальні атрибути async і defer використовуються для того, щоб поки вантажиться зовнішній скрипт — браузер показав іншу (наступну за ним) частину сторінки. Без них цього не відбувається.

атрибут async підтримується всіма браузерами, крім IE9-. Скрипт виконується повністю асинхронно. Тобто, при виявленні <script async src = "..."> браузер не зупиняє обробку сторінки, а спокійно працює далі. Коли скрипт буде завантажений — він виконається.

атрибут defer підтримується всіма браузерами, включаючи самі старі IE. Скрипт також виконується асинхронно, не змушує чекати сторінку, але є дві відмінності від async. Перше — браузер гарантує, що відносний порядок скриптів з defer буде збережений. Тобто, в такому коді (з async) першим працюватиме той скрипт, котрий швидше завантажиться.

<script src="1.js" async></script>

<script src="2.js" async></script>

А в такому коді (з defer) першим спрацює завжди 1.js, а скрипт 2.js, навіть якщо завантажився раніше, буде його чекати.

<script src = "1.js" defer> </ script> <script src = "2.js" defer> </ script>

Тому атрибут deferвикористовують в тих випадках, коли другий скрипт 2.js залежить від першого 1.js, наприклад — використовує щось, описане першим скриптом. Друга відмінність — скрипт з defer спрацює, коли весь HTML-документ буде оброблений браузером. Наприклад, якщо документ досить великий.

<script src = "async.js" async> </ script> <script src = "defer.js" defer> </ script>

Той скрипт async.js виконається, як тільки завантажиться — можливо, до того, як весь документ готовий. А defer.js почекає готовності всього

документа. Це буває зручно, коли ми в скрипті хочемо працювати з документом, і повинні бути впевнені, що він цілком отриманий.

Обидва фреймворки дуже схожі за ідеєю. Але на початку було обрано Ionic, бо за прикладами коду в Інтернеті він виглядав більш зрілим, в ньому було менше підготовчого коду. Це була помилка. При розробці додатку було виявлено що нещодавно в Ionic було інтегровано JavaScript фреймворк Vue, він використовувася за замовчуванням. При цьому документація залишилась старою, а сам фреймворк розділився на два різні модулі. Це значною мірою вповільнило розробку. Сплутанність модулів і відсутність доброї документації підштовхнули мене спробувати React Native.

Ситуація з React Native була зовсім інша, він виявився більш стабільним. Документація була актуальна, а також з’явились дуже цікаві і зручні інструменти розробки і встановлення додатків на телефони як Expo. Тож фінальне рішення було — використовувати React Native.

2 ПРАКТИЧНА ЧАСТИНА

2.1 Підключення плати

Для проекту було використано плати ESP32-CAM, Arduino Uno (китайська копія) та датчик руху HC-SR501. Плата Arduino Uno використовувалась насамперед для програмування ESP32-CAM, датчик руху підєднувався до ESP32-CAM. Підключення відбувалося за наступною схемою, зображеною на рисунку 2.1.

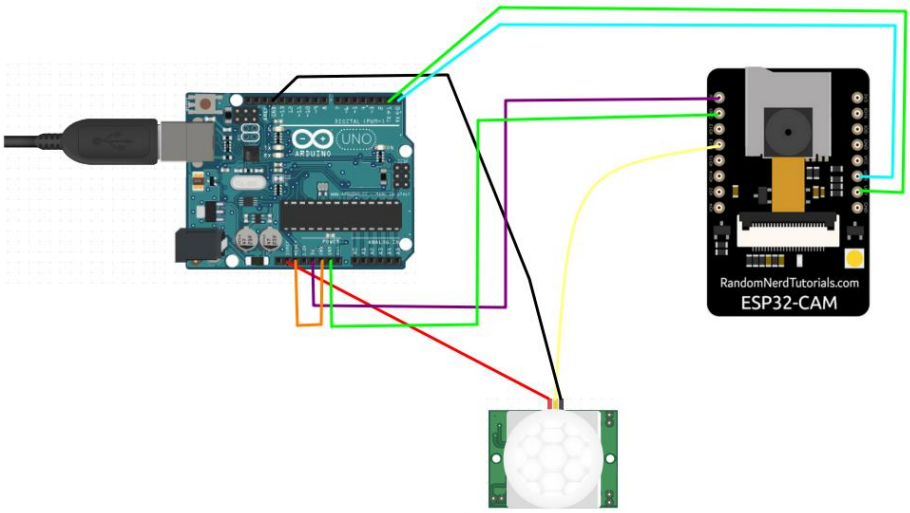

Рисунок 2.1 — Схема підключення плат Arduino Uno, ESP32-CAM і датчика HC-SR501

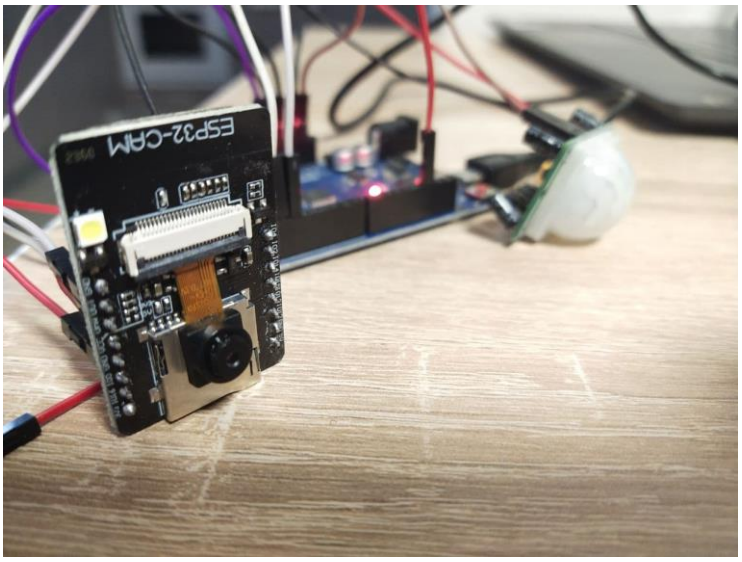

Рисунок 2.2 — Вид плати ESP32-CAM

Для сборки прототипу не використовувалась пайка. Замість цього плати просто з’єднуємо проводами і підключаємо до компьютера через USB порт.

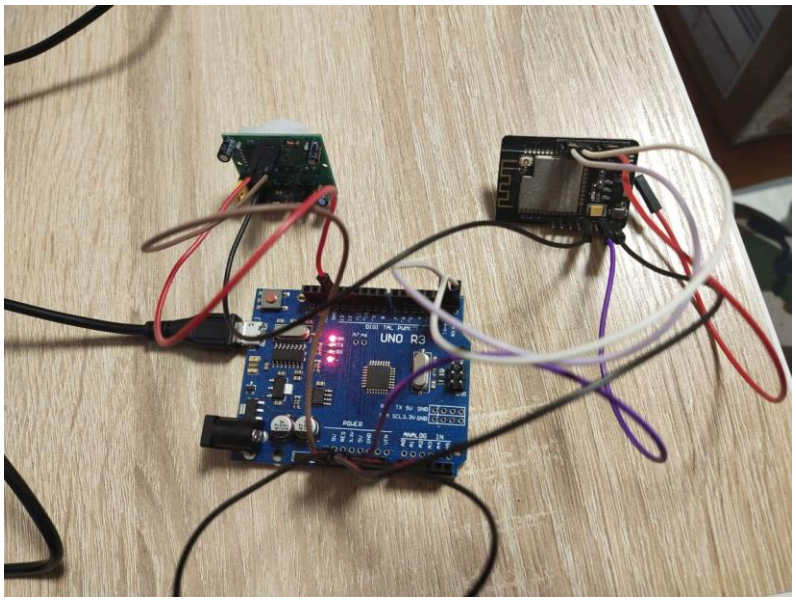

Рисунок 2.3 — Зібраний прототип приладу

2.2 Програмування плати

Програмування проводилось лише для плати ESP32-CAM. Для цього була використана Arduino IDE, і стандартні бібліотеки за пакету ESP32 Camera Web Server, що були мінімально відредаговані для підключення до локальної мережі і відправлення на сервер даних з датчика. У функції setup відбувається канфігурування серверу для трансляції відео, налаштування деяких параметрів сенсору камери. В якості серверу камери використовується стандартний код з бібліотеки Camera Web Server. Повний код буде доступний в додатку. Нижче представлено частину коду для датчика руху, а саме головна функція loop:

```
void loop() {
  // put your main code here, to run repeatedly:
  int level;
  level = digitalRead(13);
  if(level == HIGH){
     Serial.print("I see you");
     if ((WiFi.status() == WL_CONNECTED)) { // Check the current
connection status
HTTPClient http;
http.begin(host); // Specify the URL
int httpCode = http.GET(); // Make the request
if (httpCode > 0) { //Check for the returning code
  String payload = http.getString();
  Serial.println(httpCode);
  Serial.println(payload);
  Serial.print("Report successfully sent !\n");
}
```

```
        else {
          Serial.println("Error on HTTP request");
          Serial.println(httpCode);
        }
        http.end(); //Free the resources
        } else {
        Serial.print("Can't send report, not connected to Wi-Fi\n");
        }
        }
  delay(3000);
}
```

Як видно з коду, програма зчитує сигнал з входу 13 у циклі. Якщо двоїчний сигнал на вході “Високий” — робиться спроба відправити на сервер дані про рух. Також обробляється можлива помилка при відправленні.

Змінити налаштування камери досить просто. Просто потрібно використовувати рядки коду після ініціалізації камери. Після цього можно використовувати звичайні функції та код для управління камерою. Щоб змінити налаштування зображення, після ініціалізації камери треба змінити параметри сенсора.

```
sensor_t * s = esp_camera_sensor_get()
```

Щоби змінити яскравість:

```
s->set_brightness(s, 0); // -2 to 2
```

Для зміни контрасності:

```
s->set_contrast(s, 0); // -2 to 2
```

Для зміни насиченості кольору:

```
s->set_saturation(s, 0); // -2 to 2
```

Для встановлення спеціального фільтру:

```
s->set_special_effect(s, 0); // 0 to 6 (0 – No Effect, 1 – Negative, 2 –
Grayscale, 3 – Red Tint, 4 – Green Tint, 5 – Blue Tint, 6 – Sepia)
```

Для встановлення балансу білого:

s->set_whitebal(s, 1); // 0 = disable , 1 = enable

Та інші властивості:

s->set_awb_gain(s, 1); // 0 = disable , 1 = enable

s->set_wb_mode(s, 0); // 0 to 4 – if awb_gain enabled (0 – Auto, 1 – Sunny, 2 – Cloudy, 3 – Office, 4 – Home)

s->set_exposure_ctrl(s, 1); // 0 = disable , 1 = enable

s->set_aec2(s, 0); // 0 = disable , 1 = enable

s->set_ae_level(s, 0); // -2 to 2

s->set_aec_value(s, 300); // 0 to 1200

s->set_gain_ctrl(s, 1); // 0 = disable , 1 = enable

s->set_agc_gain(s, 0); // 0 to 30

s->set_gainceiling(s, (gainceiling_t)0); // 0 to 6

s->set_bpc(s, 0); // 0 = disable , 1 = enable

s->set_wpc(s, 1); // 0 = disable , 1 = enable

s->set_raw_gma(s, 1); // 0 = disable , 1 = enable

s->set_hmirror(s, 0); // 0 = disable , 1 = enable

s->set_colorbar(s, 0); // 0 = disable , 1 = enable

Також це можна зробити у графічному інтерфейсі (див. рис. 2.4).

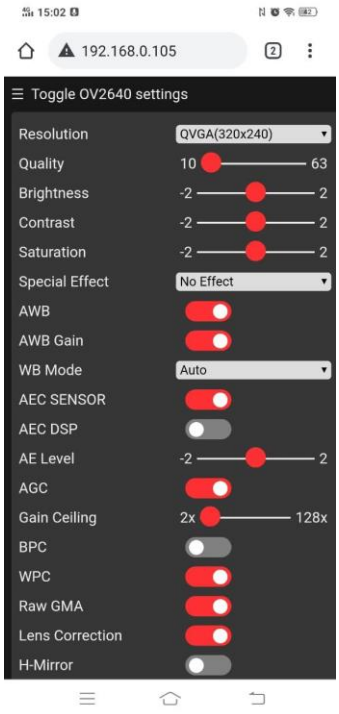

Рисунок 2.4 — Налаштування відео доступні в ESP32-CAM

2.3 Програмування серверу

Серверна частина написана за допомогою пакету Flask і мови програмування Python. Сервер має 2 логічних адреси для запитів. Перша повертає статус датчика руху, за замовчуванням — “Руху не має” і слугує точкою входу для запитів з мобільного додатку (приклад роботи серверу зображено на рисунку 2.5). Друга служить точкою входу для запитів з плати ESP32-CAM і обновлює значення змінної якщо було виявлено рух датчиком. При спрацюванні датчика сервер буде 10 секунд вертати статут “Виявлено рух”. Також обробляється ситуація коли за 10 секунд рух було виявлено кілька разів поспіль (у цьому разі плата зробить кілька запитів і сервер буде збільшувати час повертання сигналу про рух відповідно до часу останнього запиту). Код серверної частини буде наведено у додатку Б.

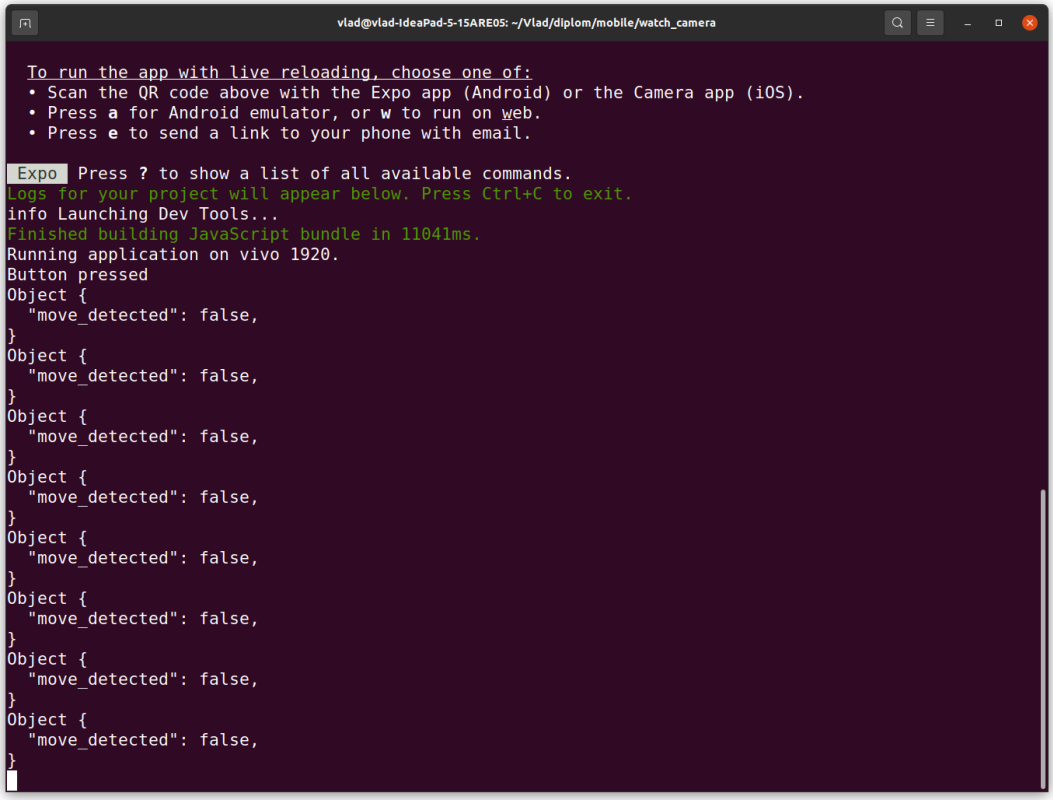

Рисунок 2.5 — Логи робочого серверу на Flask

2.4 Програмування мобільного додатку

Мобільний додаток (див. рис. 2.6) буде представляти собою інтерфейс для перегляду статусу датчика руху і перегляду відео з камери за бажанням користувача.

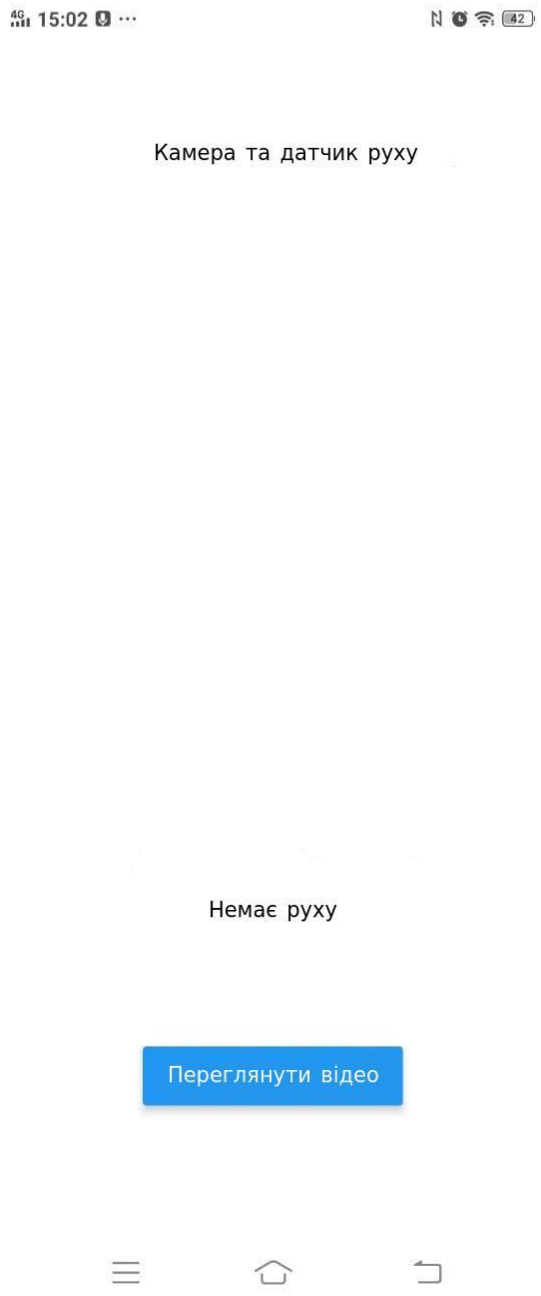

Рисунок 2.6 — Інтерфейс мобільного додатку

Для програмування мобільного додатку було використано покет React Native та мову програмування JavaScript. Цей вибір дозволив використовувати спільний код для iOS та Android додатків. Також відносна простота програми (незалежність від нативних компонентів операційних систем) сильно вплинули на вибір.

Мобільний додаток показує статус датчика руху у реальному часі, зчитуючи інформацію з сервера. А також має кнопку для перегляду відео з камери (див. рис. 2.7).

Рисунок 2.7 — Повідомлення про знаходження руху

Повний код буде доступний в додатку. Для прикладу наведемо основні функції програми.

```
render() {
        return (
        <View style={styles.container}>
        <Text style={{marginBottom: 400, fontSize: 20}}>Камера и датчик движения</Text>
                <Text style={{ marginBottom: 100, fontSize:14 }}>{this.state.move_msg}</Text>
                <Button
        onPress={
                () => this.videoRequest()
        }
        title = "Посмотреть видео"
        />
        </View>
        )
 }
```

Фунція render відображає основні елементи програми: текст за статусом з датчика, кнопку для перегляду відео, назву додатку.

```
videoRequest() {
        //On button press - show video
        console.log("Button pressed");
   Linking.openURL("http://192.168.0.105")
 }
```

Функція videoRequest відкриває веб сторінку у системному браузері в додатку на якій можно переглядати відео з камери (див. рис. 2.8).

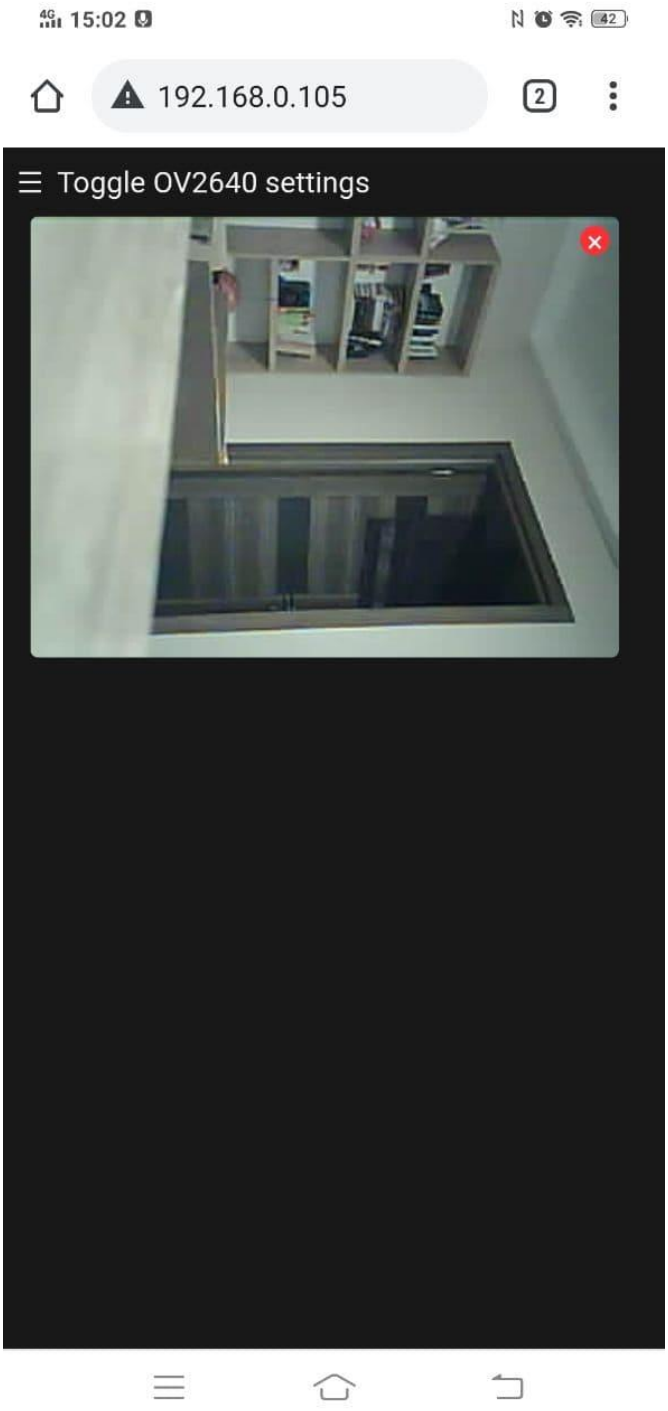

Рисунок 2.8 — Перегляд відео в мобільному додатку

```
checkMove() {
    resp = fetch('http://192.168.0.103:5000/ask')
            .then((response) => response.json())
                .then((responseJson) => {
```

```
            console.log(responseJson)
            return responseJson;
      }).then((rObj) => {
            if(rObj.move_detected) {
               this.setState({move_msg : "Движение обнаружено !!! Посмотрите
видео"})
            } else {
              this.setState({move_msg : "Движение не обнаружено"})
            }})
 }
```

Функція checkMove робить запит на сервер для перевірку статусу датчика руху. У разі повідомлення про рух — функція змінює текст елементу додатку та інформую користувача що було виявлено рух.

2.5 Тестування

Для тестування було зібрано по схемі плати і датчика. Встановлено мобільний додаток та запущено сервер у локальній мережі. Потім було проведено серію тестів використання додатку в умовах наближених до реального застосування. Датчик досить впевнено розпізнавав рух на відстані 0,5 — 3м та посилав коректні дані на сервер. Мобільний додаток одразу ж відображав їх в інтерфейсі користувача. Загалом результати тестування були задовільні. Як наступний логічний крок систему можно легко перенести на віддалений хостинг з доступом в Internet і вона буде доступна на великій відстані від камери. Для покращення системи надалі також необхідно перевести сервер на шифрований канал зв’язку HTTPS, а також упаковати плати і датчик у якийсь каракас. Також для здешевшення і спрощення

системи можно замінити плату Arduino Uno на будь яке джерело живлення для плати ESP32-CAM і датчика руху. Робити програмування плати за допомогою FTDI программатора або плати Aruino Uno чи будь якого іншого способу, а для використання вже збирати із звичайним адаптером для живлення 5В.

ВИСНОВКИ

Дослідження, що були проведені в кваліфікаційній роботі, дозволили зробити низку висновків та узагальнень теоретичного і практичного характеру.

Передусім можна зробити висновок що галузь Інтернету речей є зрілою галузю науки. Дешевизна деталей та відкритість великої кількості засобів розробки дає можливість експерементувати витрачаючи мінімальні гроші.
В роботі було зроблено огляд сучасного стану галузі Інтернету речей, його основних областей застосування, небезпекта проблем галузі а також вірогідного майбутнього. На прикладі камери відоспостереження було зроблено огляд основних елементів сучасного продукту інтернету речей. Проаналізовані кілька конкурентних технологій для програмування плат, серверної частини, мобільного додатку.

В кваліфікаційній роботі було досліджено основні компоненти інтернету речей, розглянуто плату ESP32-CAM, розроблено мобільний додаток для віддаленого управління IP камерою. Результатом роботи є створення прототипу приладу — камери відеонагляду з датчиком руху і мобільним додатком.

ПЕРЕЛІК ПОСИЛАНЬ

1. Wikipedia. Інтернет речей [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D1%80%D0%B5%D1%87%D0%B5%D0%B9
2. Wikipedia. Розумний дім [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%83%D0%BC%D0%BD%D0%B8%D0%B9
3. Wikipedia. ESP8266 [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/ESP8266
4. Wikipedia. ESP32 [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/ESP32
5. Компанія Espressif. ESP32 Series datasheet [електронний ресурс]. Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
6. Wikipedia. Arduino [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Arduino
7. Wikipedia. Python [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Python
8. Wikipedia. Flask [електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Flask
9. Компанія Ionic. Ionic [електронний ресурс]. Режим доступу: https://ionicframework.com/
10. Компанія Facebook Inc. React Native [електронний ресурс]. Режим доступу: https://reactnative.dev/

11. dw.com. Еволюція 2.0: «інтернет речей». Режим доступу: https://www.dw.com/uk/%D0%B5%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D1%96%D1%8F-20-%D1%96%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%80%D0%B5%D1%87%D0%B5%D0%B9/a-15655337

ДОДАТОК А

Код плати ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include <HTTPClient.h>

//
// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
//            or another board which has PSRAM enabled
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "Mercedes";
const char* password = "0679611483";

const char* host = "http://192.168.0.103:5000/"; // My local IP ADRESS

void startCameraServer();
```

```
void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  pinMode(13, INPUT);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
```

```
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
```

```
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  int level;
  level = digitalRead(13);
  if(level == HIGH){
     Serial.print("I see you");
     if ((WiFi.status() == WL_CONNECTED)) { // Check the current connection status
       HTTPClient http;
       http.begin(host); // Specify the URL
       int httpCode = http.GET(); // Make the request
         if (httpCode > 0) { //Check for the returning code
           String payload = http.getString();
           Serial.println(httpCode);
           Serial.println(payload);
           Serial.print("Report successfully sent !\n");
         }
         else {
           Serial.println("Error on HTTP request");
           Serial.println(httpCode);
         }
         http.end(); //Free the resources
     } else {
         Serial.print("Can't send report, not connected to Wi-Fi\n");
     }
  }
  delay(3000);
}
```

ДОДАТОК Б

Код серверної частини проекту

```
#Web server for camera and phone communication
import time
import asyncio
from flask import jsonify
from flask import Flask

app = Flask(__name__)

is_move = False
wait_for = time.time() + 10

async def turn_off_after():
    #print("Entered turn off after function")
    global wait_for
    global is_move
    my_wait_for = wait_for
    secs = my_wait_for - time.time()
    #print("Going to wait for ", secs)
    await asyncio.sleep(secs)
    if (wait_for - my_wait_for) < 0.001:
        #print("my_wait_for ", my_wait_for)
        #print("wait_for ", wait_for)
        is_move = False
        #print("Set is_move to False")
```

```
@app.route('/')
def main():
    #print("Entered main function")
    global is_move
    global wait_for
    if not is_move:
        is_move = True
        #print("Set is_move  = true")
    wait_for = time.time() + 10
    asyncio.run(turn_off_after())
    return 'Updated: movement await time'

@app.route('/ask')
def ask():
    #print("Triggered ask function")
    global is_move
    #print(is_move)
    ret = {"move_detected": is_move}
    return jsonify(ret)
```

ДОДАТОК В

Код мобільного додатку

```
import { StatusBar } from 'expo-status-bar';
import React, {Component} from 'react';
import { StyleSheet, Text, View, Button, Linking } from 'react-native';

export default class ButtonBasics extends Component{

 constructor(props){
  super(props);
  this.state = { move_msg: "Движение не обнаружено" };
 }


 sleep(ms) {
      return new Promise(resolve => setTimeout(resolve, ms));
 }



 render() {
      return (
      <View style={styles.container}>
      <Text style={{marginBottom: 400, fontSize: 20}}>Камера и датчик движения</Text>
            <Text style={{ marginBottom: 100, fontSize:14 }}>{this.state.move_msg}</Text>
```

```
                <Button
        onPress={
                () => this.videoRequest()
        }
        title = "Посмотреть видео"
        />
        </View>
        )
 }

 videoRequest() {
        //On button press - show video
        console.log("Button pressed");
   Linking.openURL("http://192.168.0.105")
 }

checkMove() {

   resp = fetch('http://192.168.0.103:5000/ask')
          .then((response) => response.json())
              .then((responseJson) => {
               console.log(responseJson)
               return responseJson;
      }).then((rObj) => {
                if(rObj.move_detected) {
                  this.setState({move_msg : "Движение обнаружено !!! Посмотрите
видео"})
                } else {
                  this.setState({move_msg : "Движение не обнаружено"})
```

```
            }})
 }

componentDidMount() {
  this.interval = setInterval(() => this.checkMove(), 4000);
}
componentWillUnmount() {
  clearInterval(this.interval);
}

}
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```