Національний університет «Полтавська політехніка імені Юрія Кондратюка» (повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки (повна назва факультету)

Кафедра комп’ютерних та інформаційних технологій і систем

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проекту (роботи)**

бакалавра

(рівень вищої освіти)

на тему

Розробка системи керування навчальної установки «Мостовий кран» на базі

мікроконтролера

Виконав: студент 4 курсу, групи 402-ТК спеціальності

|  |  |  |  |
| --- | --- | --- | --- |
| 123 | | | Комп’ютерна інженерія |
|  |  |  | (шифр і назва напряму) |
|  |  |  | Руденко В.В. |
|  |  |  | (прізвище та ініціали) |
| Керівник | | | Руденко О.А. |
|  |  |  | (прізвище та ініціали) |
| Рецензент | | |  |
|  |  |  | (прізвище та ініціали) |

Полтава – 2021

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ«ПОЛТАВСЬКА ПОЛІТЕХНІКА**

**ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**Навчально-науковий інститут інформаційних технологій та РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛавра**

**спеціальність 123 «Комп’ютерна інженерія»**

**на тему:**

**«Розробка системи керування навчальної установки «Мостовий кран» на базі мікроконтролера»**

**Студента групи 402-ТК Руденка Віталія Віталійовича**

Керівник роботи

кандидат технічних наук, доцент Руденко О.А.

Консультант

старший викладач

Демиденко М.І.

Завідувач кафедри

кандидат технічних наук,

доцент Головко Г.В.

Полтава – 2021

**РЕФЕРАТ**

Кваліфікаційна робота бакалавра: 53с.,14рис., 2 додатки, 12 джерел.

**Актуальність роботи.** Наразі є велика кількість вантажопідіймальної техніки, механічна складова якої не є застарілою, а програмно-апаратна складова або застаріла, або повністю відсутня. Модернізація навчальної установки дозволить підтягнути її рівень до сучасності. Також, програмно-апаратний комплекс для навчальної установки, фактично, буде придатний до аналогічних робочих машин.

**Об'єкт дослідження:** сучасні пристрої та програмно-апаратні комплекси для вантажопідіймальних машин.

**Предмет дослідження:** програмно-апаратний комплекс для керування навчальною установкою мостового крану.

**Метою роботи** є модернізація навчальної установки мостового крану шляхом встановлення програмно-апаратного комплексу, що дозволить віддалено керувати установкою через смартфон чи комп’ютер.

**Хронологічні межі.** Розробка проекту розпочалася з проектування в січні 2021 року і завершилась реалізацією в червні 2021 року.

**Наукова новизна і теоретичне значення:** проект показує можливість і доцільність модернізації застарілої навчальної техніки, а також розвиває напрям дистанційного керування вантажопідіймальними машинами через WiFi інтерфейс.

**Практичне значення:** створений програмно-апаратний для модернізації начальної крановою установки може застосовуватися і на робочих машинах.

**Ключові слова:** прошивка, програмно-апаратний комплекс, мікроконтролер, мостовий кран, навчальна установка

**ANNOTATION**

Bachelor's qualification work: 53 p., 14images, 2addition, 12 sources.

**Relevance of work.** This work is relevant because there are currently a large number of lifting equipment, the mechanical component of which is not obsolete, and the software and hardware component is either obsolete or completely absent. Modernization of the educational crane will allow to bring its level to the present. Also, the software and hardware complex for the training crane, in fact, will be suitable for similar machines.

**Object of research:** modern devices and software and hardware complexes for lifting machines.

**Subject of research**: software and hardware complex for control of the training bridge crane.

**The purpose of the work** is to modernize the training bridge crane by installing a software and hardware complex that will allow you to remotely control the crane via a smartphone or computer.

**Chronological boundaries.** Project development began with the design in January 2021 and ended with the implementation in June 2021.

**Scientific novelty and theoretical significance**: the project shows the possibility and expediency of modernization of obsolete educational equipment, as well as develops the direction of remote control of lifting machines via WiFi interface.

**Practical significance**: the created software and hardware for the modernization of the initial crane can be used on working machines.

**Keywords**: firmware, software and hardware complex, microcontroller, bridge crane, training installation.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 4](#_Toc74141530)

[РОЗДІЛ 1 ЗАГАЛЬНИЙ ОГЛЯД ВАНТАЖОПІДІЙМАЛЬНИХ МЕХАНІЗМІВ ТА ТРЕНАЖЕРІВ 7](#_Toc74141532)

[1.1 Загальних опис підіймальних кранів 7](#_Toc74141533)

[1.2 Огляд системикерування мостового крану 9](#_Toc74141534)

[1.3 Огляд існуючих пультів керування 11](#_Toc74141535)

[1.4 Специфікація вимог 13](#_Toc74141536)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ПРОГРАМНО–АПАРАТНОГО КОМПЛЕКСУ 15](#_Toc74141537)

[2.1 Постановка задачі проектування 15](#_Toc74141538)

[2.2 Проектування апаратної частини 17](#_Toc74141539)

[2.3 Проектування прошивки 19](#_Toc74141540)

[РОЗДІЛ 3 СТВОРЕННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ 20](#_Toc74141541)

[3.1 Принцип і алгоритм роботи пристрою 20](#_Toc74141542)

[3.2 Апаратна складова програмно-апаратного комплексу 20](#_Toc74141543)

[3.3 Основні технічні характеристики приладу 22](#_Toc74141544)

[3.4 Реалізація прошивки 26](#_Toc74141545)

[РОЗДІЛ4 ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ КІНЦЕВОГО ПРОДУКТУ 37](#_Toc74141546)

[4.1 Тестування програмно-апаратного комплексу 37](#_Toc74141547)

[4.2 Демонстрація кінцевого продукту 40](#_Toc74141548)

[ВИСНОВКИ 44](#_Toc74141549)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 46](#_Toc74141550)

[ДОДАТОК А ЛІСТИНГ КОДУ ПРОШИВКИ 48](#_Toc74141551)

[ДОДАТОК Б ТАБЛИЦЯ ЗАПИТІВ 52](#_Toc74141553)

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ,** **СКОРОЧЕНЬ І ТЕРМІНІВ**

**ПК** – персональний комп’ютер

**SSID** (англ. Service Set Identifier) – унiкальний ідентифікатор WiFi мережі

**IP** (англ. Internet Protocol address) – ідентифікатор пристрою в мережі

**HTML** (англ. Hyper Text Markup Language) – мова розмітки гіпертексту

**CSS** (англ. Cascading Style Sheets) – каскадні таблиці стилів

**JS** (англ.JavaScript) – мова програмування

**АЦП** – аналогово-цифровий перетворювач

**I2C**– послідовна шина даних для зв’язку інтегральних схем

**HTTP** (англ. Hyper Text Transfer Protocol) – протокол передачі гіпертекстових документів

**SPI** (англ. Serial Peripheral Interface) – послідовний синхронний повнодуплексний стандарт передачі даних

**SPIFFS** (англ. SPI Flash File System) – це файлова система flash пам’яті .

**ВСТУП**

**Актуальність роботи.** Кінець 20-го та початок 21 століть виділяються стрімким розвитком комп’ютерної техніки, розвитком та створенням нових методів програмування, роботизацією та автоматизацію в будь-яких сферах людської діяльності. Також, до вказаного періоду створено багато механізмів, які вже не відповідають сучасним тенденціям в виробництві, але механічно не є застарілими. В цій роботі показується модернізація навчальної установки мостового крану. Також продемонстровано, що дистанційне керування можливе не лише по кабелю чи радіоканалу, а й з використання сучасних мережевих технологій, в даному випадку технології WiFi.

**Метою роботи** є модернізація навчальної установки мостового крану шляхом встановлення і під’єднання до силової частини програмно-апаратного комплексу на базі ESP32. Модернізація полягає в створенні можливості оператору дистанційно керувати крановою установкою по WiFi використовуючи Web-інтерфейс на своєму смартфоні чи ПК.

Поставлено такі задачі:

1. Дослідження підіймальних установок, для визначення особливостей керування ними.
2. Дослідження сучасних систем дистанційного керування крановими установками та виділення основних вимог до них.
3. Розробка та впровадження програмно-апаратного комплексу на базі мікроконтролера, що дозволить модернізувати застарілу підіймальну техніку.

**Об’єкт дослідження:** сучасні пристрої та програмно-апаратні комплекси для вантажопідіймальних машин.

**Предмет дослідження:** програмно-апаратний комплекс для керування навчальною установкою мостового крану.

**Хронологічні межі.** Розробка проекту розпочалася з проектування в січні 2021 року і завершилась реалізацією в червні 2021 року.

**Практичне значення:** створений програмно-апаратний для модернізації начальної крановою установки може застосовуватися і на робочих машинах. Він дозволяє дистанційно користуватися застарілою вантажопідіймальною технікою використовуючи новітні технології бездротового зв’язку.

**Особистий внесок автора.** Автором цієї роботи було розроблено та виготовлено програмно-апаратний комплекс дистанційного керування навчальною установкою мостового крану на базі мікроконтролера ESP32.

# **РОЗДІЛ 1**

# **ЗАГАЛЬНИЙ ОГЛЯД ВАНТАЖОПІДІЙМАЛЬНИХ МЕХАНІЗМІВ ТА ТРЕНАЖЕРІВ**

Можливо виділити велику кількість технічних завдань, для реалізації яких застосування виключно фізичної сили однієї людини, чи групи людей є недостатнім чи нераціональним. Так, для зрушення з місця тяжкого каміння треба 10 людей, тоді як доступними для виконання задачі є менша кількість осіб. Тому людство протягом свого існування перебуває в пошуках способів оптимізувати обсяги застосування фізичної праці при виконанні подібних робіт. Наприклад, при вирішенні вищезазначеної задачі люди зрозуміли, що застосування довгого твердого тіла (важелю) дозволяє підсилити можливості людини.

З часом, були винайдені багато інших механізмів для полегшення виконання завдань, що традиційно полягали у підійманні і переміщенні; їх ускладнення та поєднання. Сучасні крани, фактично, є не лише механізмами, а цілим комплексом, що може включати шасі, систему керування, систему запобігання критичних ситуацій тощо.

## **1.1 Загальних опис підіймальних кранів**

Чинний державний стандарт України 2986-95 містить визначення та класифікацію кранових механізмів. Відповідно до зазначеної нормативної бази, вантажопідіймальний кран – машина циклічної дії, призначена для підіймання та переміщення в просторі вантажу, підвішеного за допомогою гака чи утримуваного іншим вантажозахоплювальним органом.

Крани різняться по конструкції, основними з яких є такі типи.

1. Мостовий кран – кран, несівні елементи конструкції якого опираються безпосередньо на підкранову колію.
2. Козловий кран – кран, несівні елементи конструкції якого опираються на колію опорними стояками.
3. Кабельний кран – кран, несівними елементами якого є канати, закріплені у верхній частині опорних щогл.
4. Кран стрілового типу – кран, вантажозахоплювальний орган якого підвішений до стріли чи візка, що переміщуються по стрілі.
5. Портальний кран – пересувний поворотний кран на порталі, призначений для пропускання залізничного чи автомобільного транспорту.
6. Баштовий кран – поворотний кран з стрілою, закріпленою у верхній частині вертикально розташованої башти.
7. Залізничний кран – кран, змонтований на платформі, що пересувається по залізничній колії.
8. Плавучий кран – кран на самохідному чи несамохідному понтоні, призначений для його встановлення на пересування.
9. Стріловий судновий кран – поворотний кран стрілового типу, встановлений на борту судна і призначений для його навантаження та розвантаження.
10. Щогловий кран – поворотний кран з стрілою, закріпленою шарнірно на щоглі, що має нижню та верхню опори.

Визначені основні поняття, що характеризують можливості навантаження крану: вантажний момент, перекидний вантажний момент, конструктивна та загальна маса, тиск колеса.

Важливо згадати лінійні параметри: виліт, підхід, задній габарит, висота підіймання, глибина опускання, діапазон підіймання, висота підкранової колії.

Чи не найважливішими є параметри стійкості: власна стійкість крана, вантажна стійкість [1].

Слід додати, що наявність тих чи інших параметрів залежить від типу крану, наявності і типу шасі, підвіски, можливості повороту, підіймання тощо. Наприклад, основними параметрами для баштового крану є довжина стріли, виліт каретки, висота спуску, максимальне навантаження, поворот башти. Тоді як для крану стрілового типу основними є виліт стріли, кут підіймання стріли, кут повороту крану.

## 

## **1.2 Огляд системи керування мостового крану**

Крани можуть керуватися двома принципово різними способами: ззовні та з кабіни керування. Варіант ззовні, за допомогою радіо керування чи проводового пульта видається менш складним, але, перебуваючи в кабіні, машиніст може спостерігати за роботою вантажопідйомного механізму з зручного положення [2].

Мостові крани встановлюються, як правило, в закритих цехах і виробничих приміщеннях, їх розташування під стелею зручно, оскільки дозволяє економити місце і робочий простір, а так само забезпечує більшу маневреність. Розміри такого обладнання завжди досить значні, тому управління таким складним механізмом завжди проводиться професійними кранівниками.

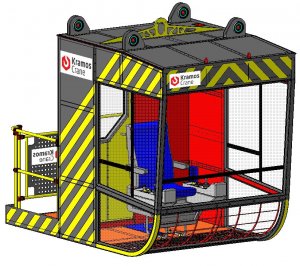


Рисунок 1.1 – Модель кабіни оператора крану [3]

В кабіні оператора крановою установкою завжди є система керування, яка, зазвичай, представлена важелями чи джойстиками, що дозволяють рухати кареткою, спускати чи підіймати гак тощо. Так як оператор знаходиться безпосередньо на крані, під час критичних ситуації, таких як перенавантаження крану, або з’їзд каретки з колії, існує ризик отримання травм.

В кабіні оператора можуть встановлюватися дисплей, світлові та звукові індикатори, що будуть показувати параметри крану. Світлові індикатори показують, чи кран рухається, чи рух крану заблокований. Блокування може бути супроводжене звуком. На дисплеї показуються фізичні параметри крану:довжина стріли, виїзд каретки, кут повороту башти тощо.

Дистанційне керування ще недавно представляло собою виносної пульт, пов’язаний багатожильним кабелем з електронними блоками, наповненими реле управління. Ці електронні блоки передають команду від пульта до виконавчих органів засобу. Таке управління застосовують і сьогодні, але необхідно зважати на ризик обриву кабелю; при цьому оператор хоча і відчуває себе більш вільно, ніж в кабіні, все ж обмежений у діях.

Останнім часом бездротові пристрої зв’язку завзято витісняють проводовий зв’язок. Це пов’язано в першу чергу з постійним зниженням вартості радіочастотних компонентів. У той же час зростає вартість матеріалів для дротового зв’язку. Навіть на близькій відстані вартість прокладки і монтажу кабелю, ціна роз’ємів і корпусів з’єднань може сумарно виявитися вище, ніж використання передавача і приймача.

Процес створення сучасних систем дистанційного бездротового керування налічує не один десяток років. З багатьох пропонованих методів передачі сигналу в наш час частіше застосовують ультразвукові, інфрачервоні і радіочастотні системи.

## **1.3 Огляд кранових тренажерів**

Тренажер кранових операцій призначений для ознайомлення з системою керування крановою установкою та отримання навичок керування в безпечних умовах. Кранові тренажери повторюють конструкцію в менших розмірах, імітують роботу найбільш поширених типів сучасних кранів і є універсальним інструментом підготовки операторів для вироблення рефлексної моторики при управлінні обраною моделлю крана як в штатних, так і в критичних ситуаціях. На тренажері можлива відпрацювання основних операцій з вантажами при різній видимості і перешкодах у вантажних операціях, включаючи:

1. Безпечне переміщення вантажу в контейнерах різних типів.
2. Забезпечення цілісності контейнерів і берегових об’єктів;
3. Оптимальну траєкторію руху і ефективні прийоми і послідовність операцій з перевантаження контейнерів; способи мінімізації навантажень на конструкцію, кількості керуючих операцій і забезпечення допустимої динаміки переміщення вантажу.
4. Дії в критичних ситуаціях: аварійна зупинка, аварійне опускання вантажу, втрата вантажу, втрата харчування, перевантаження, сильний вітер.

Кранові тренажери діляться на два основних види: реальні навчальні установки та комп’ютерні тренажери-симулятори.

## **1.4 Огляд існуючих пультів керування**

Розглянемо пульти керування краном TELECRANE F24-6D, Micron FST 516 , EMASPV9E30B4444 та виділимо спільні риси.

TELECRANE модель F24-6D – це радіопульт з шістьма 2-х ступінчастими кнопками натискання. Фактично, це не лише пульт, а комплекс з власне пульта, та приймача, що встановлюється на кран і підключається до його системи керування. Ця система може встановлюватися на будь-які крани, на інші механізми, що потребують дистанційного керування.

Пульт має 8 кнопок: START, UP , DOWN,WEST, EAST, NORTH, SOUTH та кнопку аварійного вимкнення. Також він обладнаний поворотним ключем, що переключає швидкість руху. Рух здійснюється кроками за таким алгоритмом:керування вмикається кнопкою START, обирається швидкість (1 чи 2 кроки за натискання) ключем, потім натискається і утримується кнопка напрямку. Після відпускання, кран робить останні кроки і припиняє рух.

У комплект поставки системи радіокерування FST 516 micron 5 входить ручний передавач (пульт) FSSmicron 5 Ex и приймачFSE 516. Приймач встановлюється на кран, отримує і виконує команди з пульта [6].

Сам пульт має вимикач “СТОП”, 10 двоступінчастих кнопок, поворотній перемикач на 3 положення – 1, 2, 2+1. Також є 3 двоступінчастого приводу: переміщення крану , переміщення вантажної каретки, лебідка [7].

Пульти керування EMAS серії PV призначені для управління кранами, кран-балками, тельферами, ланцюговими і канатними талями, мостовими і козловими кранами. Також пульт серії PV може мати кілька варіантів виконання: двох, трьох, чотирьох, п’яти, шести, семи, восьми, дев’яти, десяти, одинадцяти та дванадцяти кнопкові, і включати аварійна зупинка, аварійний сигнал, сигнал, ключ марка, кнопка (перемикач) вкл / викл живлення, світловим фільтром, клавішами збільшення або зменшення швидкості роботи двигуна, двох або чотирьох позиційними джойстиками.

EMASPV9E30B4444 Пульт, що складається з 8 кнопок плюс додаткова аварійна червона кнопка.8 кнопок відповідають за рух в різних напрямках. Такий пульт можна використовувати під час дощу, в вологу погоду, в цехах з підвищеним складом пилу[8].

Видно, що у всіх розглянутих пультах є кнопки, що відповідають за рух, але відсутній будь-який інтерфейс, що буде давати інформацію про стан крану.

## **1.5 Специфікація вимог**

Навчальні посібники, або ж тренажери, повторюють кранову установку в менших масштабах, або імітують роботу крану в комп’ютерній симуляції. У всіх випадках всі інформаційно-безпекові системи на кранових тренажерах, як на баштових, так і на автокранах чи мостових має інтерфейс для виводу інформації, систему для запобігання критичних ситуації, таких як перевантаження і перекидання крану. Керування зазвичай є суто механічним, тому окремої комп’ютеризованої системи не має.

Інтерфейс зазвичай є набором світлових, звукових індикаторів, або ж одним чи більше дисплеями, та знаходиться в кабіні оператора. На інтерфейсі оператор бачить фізичні характеристики кранової установки для автокранів: довжина стріли, її кут відносно шасі, іноді кут повороту, а також фізичне навантаження на кран. На баштових замість довжини стріли вказує виїзд каретки. На мостових вказується фізичне навантаження, іноді положення каретки по осях X, Y.

Для виводу інформації на інтерфейс комп’ютеризована система зчитує стан датчиків, обробляє її та виводить результат до оператора. Також, якщо стан датчиків є незадовільним, то система безпеки заблокує або обмежить роботу крану для запобігання пошкодження чи перекидання крану. В автокранах зазвичай обмежується збільшення довжини стріли та зменшення її кута. В баштових кранах блокується збільшення виїзду каретки. В мостових стоять крайні положення по осях X,Y для запобігання виїзду за обмежені межі.

Як видно, система керування та моніторингу стоїть в кабіні оператора, тому немає можливості бачити стан крану, чи керувати ним ззовні. Наразі фактично кожне підприємство має комп’ютер, інтернет мережу, майже в кожної людини є смартфон,тому інтеграція комп’ютерів та смартфонів з крановими установками є логічним кроком.

Це можна зробити через підключення комп’ютера чи смартфона безпосередньо до інформаційно-безпекової системи крану через шину USB, UART, CAN, COM тощо. Також можна використати бездротові технології передачі даних,такі як WiFi чи Bluetooth.

Отже, для модернізації навчальної установки мостового крану треба комп’ютеризовану систему, що повинна відповідати таким вимогам:

1. Можливість підключити смартфон чи комп’ютер до системи крану.
2. Мати дистанційний інтерфейс для виводу інформації до оператора.
3. Дистанційне керування установкою.
4. Мати постійний зв’язок між пристроєм та системою крана.
5. Швидкий вивід інформації.
6. Можливість виведення інформації про стан підключення до мережі.
7. Можливість ведення відеоспостереження за установкою.

# **РОЗДІЛ 2**

# **ПРОЕКТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ**

## **2.1 Постановка задачі проектування**

Як вже зазначалось, метою дипломної роботи є модернізація навчальної установки мостового крану, для досягнення чого було виділено ряд вимог до проекту зазначених вище. Розглянемо ці вимоги та як можна їх задовольнити. На рис. 2.1 зображено саму навчальну установку «Мостовий кран».



Рисунок 2.1 – Навчальна установка «Мостовий кран».

Для налагодження зв’язку між системою крана та смартфоном, чи стаціонарним комп’ютером треба визначити, як саме буде здійснюватися зв’язок. Складність процесу під’єднання смартфона чи ПК до моторів крану напряму не співмірна його ефективності, тому слід зробити «посередника», що буде керувати моторами, та буде під’єднаний до пристрою керування. Для цієї задачі доцільно буде використання мікроконтролера, виходи якого будуть керувати моторами, а вбудовані інтерфейси такі як UART, WiFi, COM-порт тощо будуть здійснювати зв’язок з комп’ютером.

Тепер треба вирішити чи зв’язок буде дротовий, чи бездротовий. Дротовий зв’язок вимагає використання дротів, що обмежує мобільність. Застосування бездротового зв’язку вирішує питання рухомості, а сучасні технології забезпечують цілком надійний рівень зв’язку. Отже, було вирішено використати бездротові технології для забезпечення зв’язку між контролером та смартфоном чи стаціонарним комп’ютером.

Керувати краном можна буде через Web-інтерфейс – сайт в браузері смартфона чи ПК, що матиме кнопки, натискаючи на які, користувач буду посилати команду на контролер по WiFi мережі. Web-інтерфейс є зручним рішенням, бо він буде працювати незалежно від конфігурації системи, виробника смартфона чи ПК, єдине що треба – це браузер, та підключення до мережі. Також, використання WiFi, як інтерфейсу зв’язку, дозволить, по-перше, різним пристроям підключатися до системи крану, по-друге, забезпечить надійний на безперервний зв’язок між краном та комп’ютером.

Для виводу інформації про підключення до мережі можна використати світлодіодний індикатор, що буде показувати стан підключення до мережі. Ще можна використати дисплей, що позуватиме не лише стан підключення, а й IP-контролера в мережі. Для цього проекту було вирішено використати дисплей.

Для ведення відеоспостереження слід використати WiFI камери, бо вони зможуть підключитися до мережі. Зображення з камер буде виводитись на сайті.

Нижче наведено загальну структурну схему приладу.

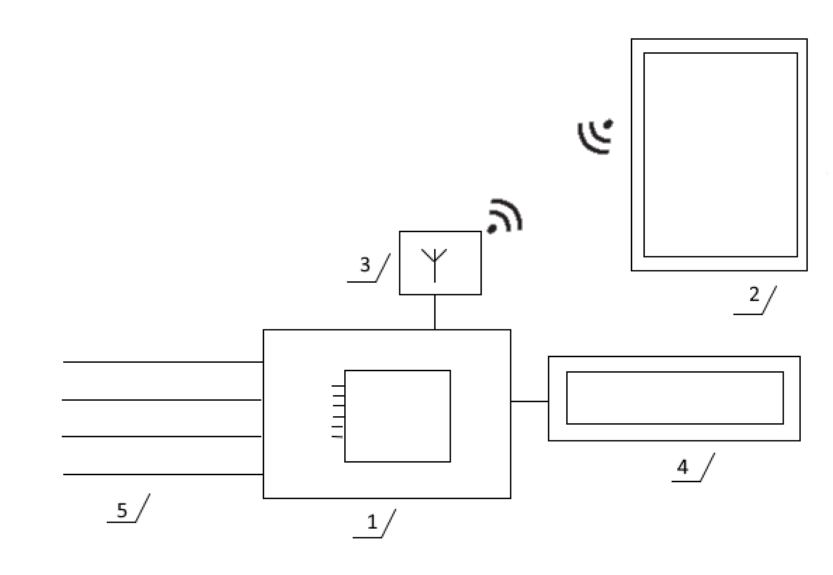


Рисунок 2.2 – Загальна схема приладу,

де: 1 – блок мікроконтролер, 2 – блок смартфон (або ПК), 3 – блок бездротового зв’язку, 4 – блок дисплея, 5 – керуючі виходи на кран.

## **2.2 Проектування апаратної частини**

Основою програмно-апаратного комплексу є мікроконтролер, тому слід максимально відповідально поставитися до вибору того контролера, що піддійте тдо нашого проекту. Наразі, є багато різних виробників, основними з яких є: STM ST Microelectronics, Espressif Systems, Atmel, Raspberry PI. У кожної з цих компаніє є велика кількість дочірніх підприємств, що випускають плати під іншими назвами. Вони мають великий вибір різноманітних плат, що мають різну обчислювальну потужність, кількість терміналів, розрядність, наявність тих або інших вбудованих функцій тощо. Серед наведеного вище обладнання, треба обрати саме ту плату, що задовольнить наші вимоги: можливість буди частиною мережі, мати підтримку I2C, достатню кількість пам’яті, підтримку UART.

Почнемо розгляд с контролерів компанії STM. Вона виробляє сімейство 32-розрядних мікроконтролерів STM32 на основі процесора ARM Cortex-M. Ці мікроконтролери поєднують дуже високу продуктивність, можливості в режимі реального часу, цифрову обробку сигналу, роботу з низьким енергоспоживанням, низькою напругою, зберігаючи при цьому повну інтеграцію та простоту розробки. Асортимент мікроконтролерів STM32, заснований на стандартному ядрі, постачається з широким вибором інструментів та програмного забезпечення для підтримки розвитку проектів, що робить це сімейство продуктів ідеальними як для невеликих проектів, так і для наскрізних платформ [9].

Вони мають дві серії мікроконтролерів, що підтримують бездротовий зв’язок: STM32WB, STM32WL. Вони підтримують стандарт IEEE 802.11, мають i2cінтерфейс, але в цих серіях недостатній для цього проекту розмір flash пам’яті.

Далі розглянемо Espressif Systems. Ця компанія спеціалізується на мікроконтролерах з підтримкою бездротових технологіях, тому їхні вироби мають вбудовані WiFi та Bluetooth модулі. Пропонує фірма такі моделі: ESP8266 та ESP32. ESP32 є логічним продовженням ESP8266 та має кращі параметри ніж попередник. Програмувати на цих контролерах можна як на С, так і на Mycro Python. Платформа Arduino має підтримку цих мікроконтролерів.

Ще слід розглянути Raspberry PI. Мікроконтролери цієї компанії є досить потужними, підтримують різні технології, та, фактично, є повноцінними комп’ютерами. Ціна на ці вироби, відповідно, висока. Зважаючи на те, що програмно-апаратне забезпечення навчальної установки не повинне бути мультифункціональним та універсальним комп’ютером, мікроконтролери цієї фірми не є оптимальними для цього проекту.

З оглянутих вище варіантів найбільш підходящим є ESP32, бо він має вбудовану підтримку бездротового зв’язку і достатній об’єм пам’яті, тому зупинимося на цьому мікроконтролері.

Щодо дисплея, є два основних варіанти: використання LCD-індикатора, чи LED дисплея. Для вирішення, що саме треба використати, слід визначити яку інформацію дисплей буде доносити. В цьому проекті дисплей повинен показувати лише стан підключення по мережі, та IP мікроконтролера в цій мережі. Отже, велике розширення дисплея не потрібне, бо виводитись будуть лише цифри. Тому доречним буде використання індикаторів.

## **2.3 Проектування прошивки**

Особливістю програмування мікроконтролерів є те, що прошивка, вихідний код, цілком залежить від технічних характеристик контролера, кількості пінів, частоти таймера, наявності АЦП тощо. Було обрано мікроконтролер ESP32, отже при розробці прошивки слід опиратися від його можливостей, та використовувати програмне забезпечення, що підтримує цей контролер.

Для розробки програмного забезпечення Arduino IDE, ця платформа має підтримку ESP32 та має вбудований компілятор, вбудований засіб для завантаження прошивки в мікроконтролер, та має швидких та зручний пошук бібліотек для використання в проекті. Після завантаження та інсталяції на комп’ютер даної IDE автоматично отримуємо велику кількість навчальних скетчів, які використовуються для демонстрації можливостей навчальної установки.

Ще слід розглянути WiFi-камеру, яка теж працює на базі мікроконтролера, отже в неї можна записати програму. В цьому проекті основне завдання камери – це запис та трансляція роботи крану до оператора. Встановлена камера дозволяє бачити процес роботи крану та керувати ним, не знаходячись в кабіні оператора.

# **РОЗДІЛ 3**

# **РЕАЛІЗАЦІЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ**

## **3.1 Алгоритм роботи пристрою**

Алгоритм роботи пристрою є досить простим. Пристрій в свої пам’яті має інформацію, SSID та пароль, про WiFi мережу до якої слід підключитися, до цієї ж мережі підключається і пристрій керування, смартфон чи ПК. Користувач, якщо підключення вдалося, бачить на дисплеї мікроконтролера його IP-адресу, яку він вводить в браузері. Після цього браузер надсилає запит на цю адресу до маршрутизатора, який перенаправляє запит на мікроконтролер. Так як всі ці пристрої функціонують в одній мережі, мікроконтролер отримує запит. В прошивці написано, що при отримані запитана своя адреса, ESP32 відсилає відповідь на адресу відправника, в якій міститься Web-інтерфейс, що є html, css, js файлами. Користувач завантажує Web-інтерфейс та розпочинає керування краном.

Слід додати, що в цій системі присутня WiFi-камера, яка в свою чергу є окремим пристроєм на мікроконтролері. Вона теж підключається до мережі, отримує свою IP-адресу, та стає повноцінним учасником мережі. Програма прошивки камери працює так, що при зверненні за адресою камери на неї, вона починає на адресу відправника транслювати відео, яке сама записує і обробляє.

## **3.2 Апаратна складова програмно-апаратного комплексу**

Зважаючи на вимог та особливості проекту, на обраний мікроконтролер ESP32, треба розробити архітектуру, що задовольнить поставлені вимоги. Розробка ведеться фактично навколо ESP32, враховуючи особливості його архітектури, наявні піни, наявність і розрядність АЦП. Для того щоб пристрій працював безвідмовно, рівні напруги повинні бути узгоджені. Нижче наведено принципову схему мікроконтролера ESP32 (рис. 3.1):

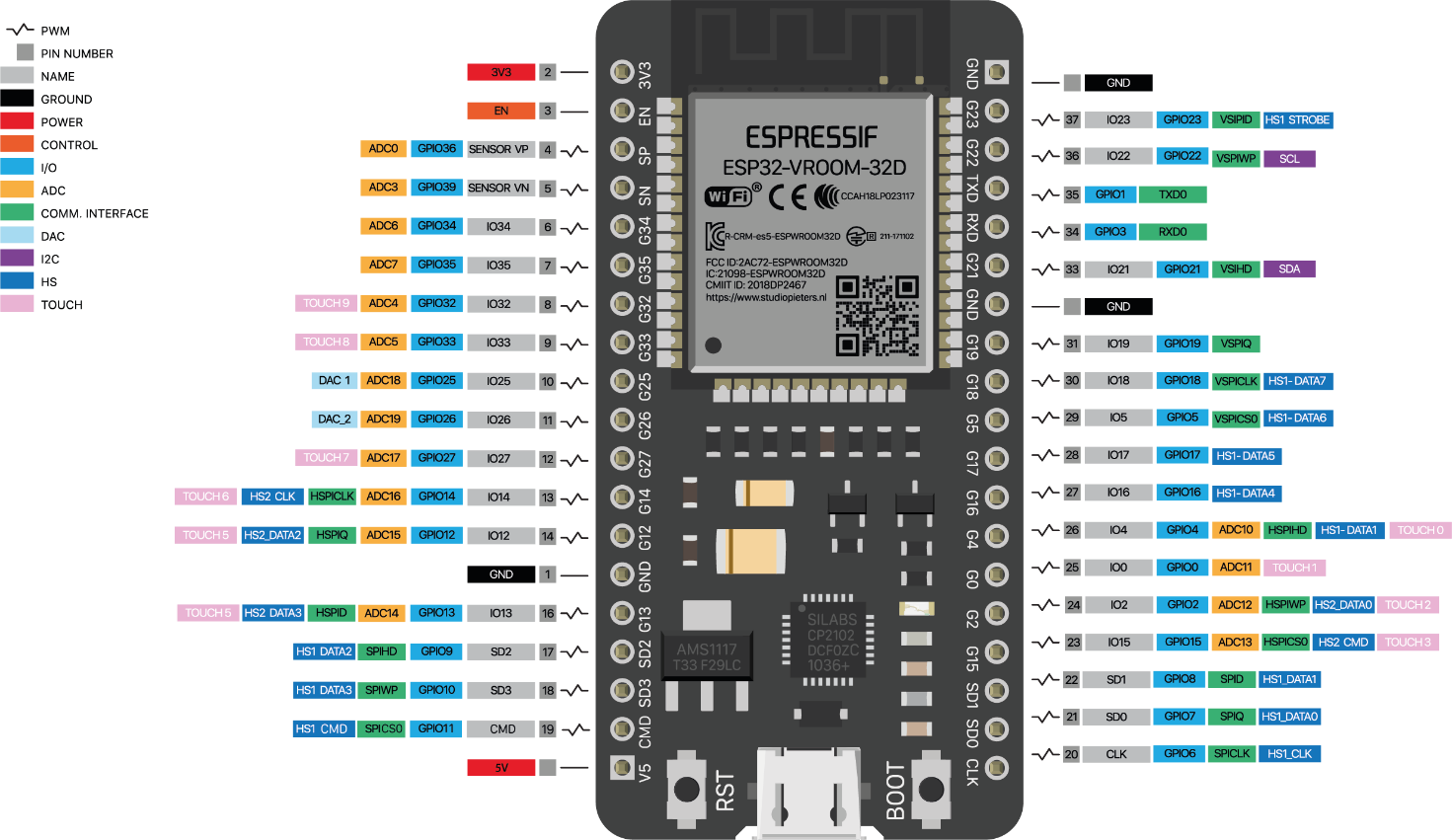


Рисунок 3.1 – Принципова схема мікроконтролера

ESP32 здатний надійно функціонувати в промислових умовах із робочою температурою в діапазоні від –40 ° C до + 125 ° C.

Розроблений для мобільних пристроїв, електроніки та додатків інтернету речей, ESP32 забезпечує наднизьке енергоспоживання за допомогою комбінації декількох типів фірмового програмного забезпечення. ESP32 також включає найсучасніші функції, такі як дрібнозернистий годинник, різні режими живлення та динамічне масштабування потужності.

ESP32 високо інтегрований із вбудованими антенними вимикачами, радіочастотним сигналом, підсилювачем потужності, підсилювачем прийому з низьким рівнем шуму, фільтрами та модулями управління потужністю. ESP32 додає безцінну функціональність та багатофункціональність вашим програмам з мінімальними вимогами до друкованих плат (PCB).

ESP32 може працювати як повна автономна система або як ведений пристрій для головного MCU, зменшуючи накладні витрати на стек зв’язку на основному процесорі програми. ESP32 може взаємодіяти з іншими системами для забезпечення функціональних можливостей WiFi та Bluetooth через свої інтерфейси SPI / SDIO або I2C / UART [10].

Широкий функціонал мікроконтролера ESP32 дозволяє розробнику на його базі створювати найрізноманітніші пристрої. Контролер може виступати в ролі маршрутизатора WiFi мережі, він може забезпечувати дистанційне керування будь якими приладами та механізмами. Використовуючи цей контролер створюють метеостанції, WiFi-радіо, Bluthoose-колонки тощо.

## **3.3 Основні технічні характеристики приладу**

Для виконання проекту з дистанційного керування обов’язково повинен бути модуль дистанційного зв’язку, в цьому випадку WiFi. В ESP32 є вбудований WiFi модуль з антенною, та частотою 2.4 ГГц, отже немає сенсу встановлювати додаткові засоби бездротового зв’язку.

В мікроконтролері є певні цифрові піні, які можуть виводи лише один рівень напруги 3.3В. Ці піни будуть керуючими – вони будуть замикати реле, яке в свою чергу підключається до контактору, що стоїть в силовій частині крану, та керує моторами РД-09. Один двигун відповідає одній осі руху каретки на крані. Всього осей руху 3X,Y, Z. Тому потрібно 3 керуючи реле, що будуть вмикати двигун. Також, для визначення напрямку руху каретки по осі треба ще по керуючому реле на кожен двигун, що у ввімкненому стані буде змінювати рух каретки по осі на протилежний.



Рисунок 3.2 – Старий пульт керування, де видно контактор

З вище зазначеного зрозуміло, що для повноцінного керування рухом треба 6 реле: 3 реле для самого руху, 3 реле для визначення його напрямку. В продажу є релейні модулі 1, 2, 4, 8 , 16 – канальні. Нам треба 6 реле, тому обираємо 8-канальний модуль реле. Реле живляться від 5 В, здатні комутувати навантаження до 10 А (при напрузі до 250 В). Споживання струму кожним реле: 15-20 мА.

Раніше було визначено, що доцільно буде використання індикаторів для виводу інформації про підключення до мережі. Так як інформація, що виводиться – це IP-адреса, то слід використати найменші варіанти. В цій роботі було вирішено використати LCD індикатор 16/2 з вбудованим інтерфейсом I2C. Було обрано саме цей інтерфейс, бо він вимагає лише 4 піни для підключення, тоді як SPI-інтерфейс 6 пінів.

LCD-дисплей, це електронний пристрій візуального відображення, в нашому випадку він має 2 рядки по 16 комірок. Кожна комірка може показувати 1 символ. принцип дії якого ґрунтується на явищі електричного переходу Фредеріка в рідких кристалах. Дисплей складається з довільної кількості кольорових або монохромних точок (пікселів) і джерела світла або відбивача (рефлектора). Такі дисплеї зараз використовуються майже всюди, як в великих пристроях, таких як монітори, телевізори, так і в малих: годинниках, смартфонах тощо. Це зумовлено тим, що вони мають, в порівняні з іншими видами дисплеїв, низьке енергоспоживання.

Шина I2C – це двопровідна шина приймання і передачі, яка передає інформацію послідовно по байтах. Підтримує можливість двонаправленої передачі. Працює по принципу master / slave, або ведучий / ведений, де master – ведучий пристрій, що відсилає команди до slave, веденого, що отримує команди, та виконує їх. Сам ведений якісь самостійні дії виконувати не може, назувати він теж не може. Також слід додати, що і ведучий, і ведений можуть як передавати, так приймати інформацію, але давати команди може лише ведучий.

Одною з основних особливостей цієї шини є те, що один ведучий може мати багато ведених, керування якими здійснюється про принципу адресації. Ведений має 7-бітну (128адрес), або 10-бітну адресу (1024 адрес), ведучий повинен знати ці адреси, щоб давати команди за ними [12]. В нашому випадку ESP32 – це ведучий (master), тоді як LCD-індикатор ведений (slave), тому задіяна буде лише одна адреса.

WiFI-камера в цьому проекті є окремим модулем і фактично не є частиною архітектури приладу. Вона також не є необхідним елементом виробу – і сайт, і система керування будуть працювати без проблем, лише не буде дистанційної візуалізації процесу роботи мостового крану. Тому, було вирішено, що камеру недоречно додавати до загальної архітектури.

Живлення програмно-апаратного комплексу здійснюється блоком живлення 220V-5V 2A. Живлення блока реле здійснюється цім же блоком живлення.

На рис. 3.2 наведено схему апаратної частини виробу.

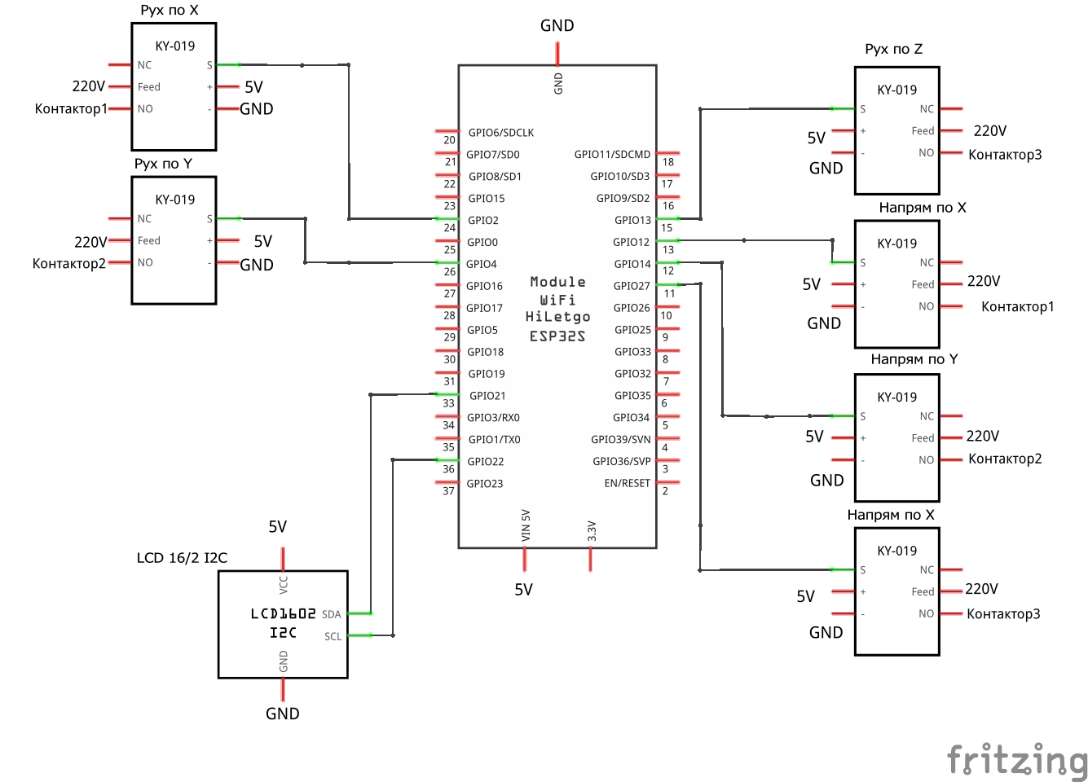


Рисунок 3.4 – Схема апаратної частини

На схемі можна побачити такі елементи: мікроконтролер ESP32, 6 реле KY-019, LCDдисплей 16/2 з підключенням по I2C. ESP32 є основою приладу, що здійснює керування дисплеєм та реле. Також, мікроконтролер підключається до мережі, отримує команди, виконує їх. Такі піни задіяні: GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO27, GPIO21, GPIO22. З них GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO27 є сигнальними, або керуючими пінами, тобто вони перемикають стан реле і тим самим керують моторами крану. GPIO2, GPIO4, GPIO13, GPIO12,GPIO14, GPIO27 керують рухом по осі. GPIO21, GPIO22 піні керують дисплеєм, командуючи, що саме дисплей повинен відображати, в нашому випадку – це IP-адреса пристрою в локальній мережі.

Отже, ми має мікроконтролер ESP32, що підключається до мережі, отримує команди, виконує їх, чи забезпечує рук каретки крана. Інформацію про підключення до мережі показує на дисплеї. Керування рухом каретки здійснюється завдяки керуючим пінам, що подають сигнал на керуючі входи реле.

## 

## **3.4 Реалізація прошивки**

Прошивка – це програма, набір команд, які виконує мікроконтролер. В ній записується з якими входами, виходами працює мікроконтролер, які його функціональні елементи задіяні в роботі. В основному програма на базі платформи Arduino умовно поділяються на 3 частини: ініціалізація, запуск, виконання.

Ініціалізація – це та частина коду прошивки що починає виконуватися, фактично, під час компіляції, перетворення тексту, або коду, на набір машинних бітових команд. В цю частину коду зазвичай записуються бібліотеки, які треба підгрузити до прошивки, препроцесорі макроси, ініціалізуються і записуються змінні і константи. Бібліотека – це набір функцій, що написані іншими програмістами. Використовуються вони для того, щоб пришвидшити процес програмування і уникнути необхідності циклічно вирішувати одні і ті самі задачі.

Запуск, або старт – це частина коду, яка виконується один раз під час вмикання системи, і більше не виконується до перезавантаження. Також, сюди записуються переривання – дії, що виконуються при настанні певних умов, наприклад, зміна рівня напруги на вході з низького на високий. Зазвичай під час запуску ініціалізуються класи, запускаються підсистеми, визначається стан пінів (вхід чи вихід), їх початковий рівень, а також записуються початкові значення в змінні. Ще на цьому етапі можуть зчитувати файл конфігурації з пам’яті.

Етап виконання виконується безперервно, циклічно. У більшості проектів це найбільша частина коду. Тут записуються команди, які треба виконувати безперервно, одна за одною, або з певним періодом. Це і зчитування датчиків, обробка інформації з них, і керування станами пінів при певних умовах.

В цьому проекті програма працює по іншому принципу, а саме за перериваннями. Фактично є лише два етапи: ініціалізація, та запуск. В прошивці записано, що мікроконтролер на етапі запуску вмикається, налаштовує піни, намагається під’єднатися до мережі WiFi. Також на цьому етапі записуються команди-переривання, які виконуються коли певна подія відбувається. Тут події – це запит від сайту. Реакцією на ці запити є відсилання html, css, js , image файлів, рух каретки крана, запинка руху, та відправка інформації на сайт про стан крану, чи він рухається і в якому напрямку.

Розглянемо код, що використаний в прошивці і почнемо з передпроцесорних директив підключення бібліотек. Нижче наведено код:

#include "WiFi.h"

#include "ESPAsyncWebServer.h"

#include "SPIFFS.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

З коду видно, що були підключені бібліотеки WiFi.h, ESPAsyncWebServer.h, SPIFFS.h, Wire.h, LiquidCrystal\_I2C.h. Вони містять набір функцій, що знадобились під час написання програми. Кожна з цих бібліотек має спеціалізацію, а саме:

* WiFi.h має в собі набір функцій, що дозволяють мікроконтролеру бути частиною WiFiмережі: отримувати IP-адресу, приймати, відправляти запити тощо;
* ESPAsyncWebServer.h має набір функцій, що дозволяє ESP32 працювати, як асинхронний Web-сервер, тобто відправляти та отримувати запити відправляти їх на сайт, виконувати запити отримання з сайту. В нашому випадку вона потрібна для обробки запитів на рух каретки крану;
* SPIFFS.h дозволяє контролеру мати файлову систему SPIFFS: завантажувати, зчитувати, вивантажувати файли різних розширень. В нашому випаду вона потрібна для зберігання html, css, js файлів і вивантаженні їх до користувача за запитом;
* Wire.h – ця бібліотека дозволяє контролеру взаємодіяти з пристроями по шині I2C;
* LiquidCrystal\_I2C.h – ця бібліотека має набір вже готових команд, для взаємодії з LCD-дисплеєм по I2C. Для роботи вона вимагає підключення Wire.h.

Далі розглянемо використання передпроцесорних директив для визначання констант:

#define axisX 13

#define axisY 4

#define axisZ 2

#define dirX 12

#define dirY 14

#define dirZ 27

В цьому відрізку коду задаються передпроцесорні константи. Вони відрізняється від звичайних констант тим, що не створюють константу в комірці пам’яті, а створюють ідентифікатор за яким приписується якесь значення, або символ, число, слово тощо. Під час процесу перетворення коду в машинні команди, компілятор помічає цей ідентифікатор в коді, і заміняє його значенням, що йому приписане. Це дозволяє економити пам’ять і не робити зайві змінні.

В вище зазначеному коді створюються ідентифікатори для номері пінів, що задіяні для руху #define axisX 13, #define axisY 4, #define axisZ 2 для пінів руху осей Х, Y, Z, тоді як #define dirX 12, #define dirY 14, #define dirZ 27 для їх напрямку.

Далі є ініціалізація змінних і констант:

const char\* ssid = "\*\*\*\*\*\*\*\* ";

const char\* password = "\*\*\*\*\*\*\*\*\* ";

//байт в який записується стан крану: рух і напрям по осям

byte state = 0;

Тут створено дві константи типу const char\*ssid та password. Константа const char\* зберігає в собі адресу масиву символі. Сам масив може мати будь який розмір. Фактично, const char\* не зберігає масив в собі, а лише посилання на нього, його нульовий байт. В ssid записаний ідентифікатор мережі, тоді як в password пароль цієї мережі.

Також створено зміну типу byte, що і займає в пам’яті 1 байт. Створена вона для того, щоб містити в собі інформацію про стан крану: чи рухається каретка, і в якому напрямі. 6 бітів байта state можуть приймати лише два значення: 0 та 1. Це означає, що перші 3 біта показують чи рухається каретка, якщо біт 1 то рухається, якщо 0 – то ні. Схожий принцип і з другою трійкою бітів: вони показують напрям руху: 0 – рух іде в одному напрямі, 1 – в протилежному.

Наприклад, після виконання запиту користувача від Web-інтерфейсу, значення змінної state стало 9, тобто 0000 1001 в двійковій системі. Розглянемо значення бітів, 1 біт має значення 1, що означає що йде рух по осі X, 4 біт теж 1, що означає, що рух протилежний.

Далі є ініціалізація асинхронного серверу з портом 80:

AsyncWebServer server(80);

LiquidCrystal\_I2C lcd(0x27, 16, 2);

Ці дві функції взяті з додаткових бібліотек і запускають роботу модулів мікроконтролера: AsyncWebServer server(80) – обробку запитів поTCP порту 80, що відповідає HTTP запитам; LiquidCrystal\_I2C lcd(0x27, 16, 2) – модуль керування LCD-індикатором, його налаштуванням та виводом інформації.

Далі починається другий етап роботи програми, запуск, або setup. Як вже зазначалось, ці команди будуть здійснені лише раз. Почнемо розгляд цього етапу з наступного відрізку кода:

Serial.begin(9600);

if(!SPIFFS.begin(true))

{

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

Почнемо розгляд з функції Serial.begin(9600). Це стандартна функція класу Serialна платформі Arduino, тому пошук і завантаження додаткових бібліотек не знадобилось. Її робота полягає в тому, щоб увімкнути Serial (послідовний) порт для передачі якоїсь інформації по ньому. В неї записаний аргумент 9600, який визначає з якою швидкістю буде йти передача даних. В цьому випадку швидкість буде 9600 біт за секунду.

Послідовний порт в має широке застосування для мікроконтролерів. По ньому можна не лише передавати якусь інформацію, а й керувати контролером з ПК, робити налагодження та тестування. Далі буде видно, що послідовний порт буде використано для передачі інформації про запуск систем мікроконтролера.

Конструкція if(!SPIFFS.begin(true)) є умовою і дією одночасно. Робота полягає в тому, що спочатку робиться дія, а потім перевіряється умова. Умова if перевіряє чи значення виразу є логічною істиною. Виразом тут є !SPIFFS.begin(true), де SPIFFS.begin(true) функція запускає систему SPIFFS, і повертає значення true, істина. Знак оклику ”!” означає інверсію, тобто якщо вираз істинний, умова не виконується.

Дією при задоволенні умови є вивід помилки в послідовний порт, та завершення роботи контролера, бо без файлової системи, програмно-апаратний комплекс працювати не буде.

Продовжуємо розгляд прошивки з наступного коду:

pinMode(axisX,OUTPUT);

pinMode(axisY,OUTPUT);

pinMode(axisZ,OUTPUT);

pinMode(dirX,OUTPUT);

pinMode(dirY,OUTPUT);

pinMode(dirZ,OUTPUT);

digitalWrite(axisX, LOW);

digitalWrite(axisY, LOW);

digitalWrite(axisZ, LOW);

digitalWrite(dirX, LOW);

digitalWrite(dirY, LOW);

digitalWrite(dirZ, LOW);

Тут слід виділити дві функції, дії яких повторюються, але для різних пінів.

Це функції pinMode() та digitalWrite(), їхні дії доволі прості, але не менш важливі.

Функція pinMode() задає в якому режимі повинен робити пін: як вихід, чи як вхід. В аргументах записується номер піну, та режим роботи. Функція digitalWrite() теж працює з пінами, вона визначає, який рівень напруги буде видавати вихід: низький чи високий. Так як піни керують моторами, то ми їх задаєм як виходи, а для того, щоби на початку роботи програми каретка крану не рухалась ми спочатку примусово задаємо їм низький рівень на виході.

Далі йде вмикання WiFi модуля в мікроконтролері, це видно в наступному відрізку коду:

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Розглянемо функції більш детально. Почнемо з WiFi.begin(ssid, password) – ця функція починає роботу ESP32 як частини WiFi-мережі. Мікроконтролер при виконанні цієї команди намагається підключитися до мережі ідентифікатор та пароль якої записуються в аргументи функції. Після підключення ESP32 стає повноцінним членом мережі та отримує IP-адресу.

Підключення не завжди йде швидко, тому в програмі написано, що б контролер не робив ніяких дій і чекав поки не отримає відповіді під маршрутизатора про успішне підключення до мережі. Для цього використовується конструкція while(), яка виконує записані в неї дії, поки дійсна умов. У нас умовою виконання є вираз WiFi.status() != WL\_CONNECTED, що означає, що цикл буде виконуватися до дих пір, поки ESP32 не підключиться до мережі.

Тепер перейдемо до команд обробки запитів від користувача:

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/index.html");

});

server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/style.css");

});

server.on("/script.js", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/script.js");

});

Ці команди записують на сервер, тобто мікроконтролер, набір умов і дій, які контролер виконує при виконанні цих умов. Поглянемо на першу команду server.on() – це функція бібліотеки ESPAsyncWebServer.h. В параметри цієї функції записуються: запит, при надходженні якого виконується дія; тип запиту; тип відповіді; та дії, які виконуються, при надходженні запиту. Наприклад, в першому використанні функції server.on() маємо аргументи: "/", що є запитом; HTTP\_GET, що є типом запиту [8]; (AsyncWebServerRequest \*request), що є типом відповіді; request->send(SPIFFS, "/index.html") є дією при отримані запиту.

В цьому відрізку коду є три запити, на які реагує ESP32: "/","/style.css","/script.js". Запит "/" надходить на ESP32,коли користувач намагається зв’язатися з мікроконтролером по IP. Він це бачить, і реагує наступним чином – відсилає htmlфайл, що є складовою Web-інтерфейсу. В цьому файлі записано, що для коректної роботи йому потрібно ще два файли: css, та js. Користувач відправляє запит на ESP32 з проханням надати ці файли, і контролер їх відсилає.

Файли з розширенням HTM або HTML – це файли на мові розмітки гіпертексту (Hypertext Markup Language), що є стандартними веб-сторінками.

Оскільки HTM-файли – це текстові файли, вони містять тільки текст, а також текстові посилання на інші зовнішні файли. Наприклад, на картинки в цій статті.

HTM і HTML-файли можуть містити посилання на інші типи файлів – наприклад, відео, CSS або JS-файли.

CSS (Каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Далі йдуть запити керування рухом каретка крану. Код наведено нижче:

server.on("/up", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00001001;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, HIGH);

digitalWrite(dirZ, HIGH);

});

server.on("/down", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000001;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, HIGH);

digitalWrite(dirZ, LOW);

});

server.on("/left", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00010010;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, HIGH);

digitalWrite(dirY, HIGH);

});

server.on("/right", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000010;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, HIGH);

digitalWrite(dirY, LOW);

});

server.on("/forward", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00100100;

request->send(200, "text/plain",String(state));

digitalWrite(axisX, HIGH);

digitalWrite(dirX, HIGH);

});

server.on("/back", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000100;

request->send(200, "text/plain",String(state));

digitalWrite(axisX, HIGH);

digitalWrite(dirX, LOW);

});

Ці конструкції працюють аналогічно, до команд, що завантажують web-інтерфейс. Відмінністю є в типі даних і в періодичності виконання. Якщо файли Web-інтерфейсу користувач завантажує лише раз, поки не зачинить сторінку в браузері, то ці запити виконуються кожен раз, коли користувач натискає на кнопку на сайті, що керує рухом каретки крану.

Також, при отриманні запиту на рух, мікроконтролер не лише встановлює високий рівень на керуючих пінах, а і змінює відповідні біти в байті стану кранової установки. Потім відправляє відповідь до користувача, що стан керуючих виходів змінено.

Далі йдуть команди, що зупиняють рух каретки крану та обнулюють байт стану, що означає, що рух спинився, та відправляють цю інформацію до користувача. Код наведено нижче:

server.on("/stopz", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state = 0;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, LOW);

digitalWrite(dirZ, LOW);

});

server.on("/stopy", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state = 0;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, LOW);

digitalWrite(axisX, LOW);

digitalWrite(dirY, LOW);

digitalWrite(dirX, LOW);

});

Для узгодження стану крану Web-інтерфейс періодично надсилає запит на мікроконтролер, що би той надіслав байт стану. Приклад нижче:

server.on("/state", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(200, "text/plain",String(state));

});

В кінці програми, коли вже всі реакції на запити записані, можна вмикати серверну частину ESP32. Також, щоб користувач розумів, на який IP йому відсилати запит, на дисплей виводиться IP адреса мікроконтролера.

server.begin();

lcd.init();

lcd.clear();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print(WiFi.localIP());

Функція server.begin() вмикає ESP32 як сервер, що може реагувати на запити. Також написані функції налаштування дисплею: lcd.init() активує дисплей; lcd.clear() очищає його від старих зображень; lcd.backlight() вмикає підсвічування; lcd.setCursor(0, 0); встановлює початкове положення курсора, тобто комірку, з якою почнеться вивід символів; lcd.print(WiFi.localIP()) виводить IP-адресу контролера.

Отже, в написаній програмі реалізовано ESP32 як частину WiFi-мережі, а також як засіб керування навчальною крановою установкою мостового крану. При ввімкнені контролера в свої прошивці він має SSID та пароль мережі, до якої підключається. Після вдалого підключення ESP32 налаштовується як сервер, отримує директиви, в яких сказано, як реагувати на запити. Також реалізована файлова система SPIFFS, в якій зберігаються файли Web-інтерфейсу. Для того, щоб користувач розумів, по якій адресі підключатися, встановлено LCD-дисплей, де виводиться IP-адреса пристрою. Також в додатку Б наведена таблиця всіх запитів, що є в програмі.

Реалізація прошивки при розробці програмно-апаратного комплексу полягає не лише в написані програми, а й в компіляції програми, тобто перетворення словесних команд в бінарний машинний код. Після цього цей код завантажується в постійну пам’ять мікроконтролера.

Для написання коду, його компіляції і завантаження в мікроконтролер було використано програмну платформу Arduino IDE.

# **РОЗДІЛ 4**

# **ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ КІНЦЕВОГО ПРОДУКТУ**

## **4.1 Тестування програмно-апаратного комплексу**

Після проектування та власне розробки програмно-апаратного комплексу йде процес тестування приладу. Тестування робиться для того, щоб перевірити, чи справо працює прилад, чи відповідає його робота і функціонал поставленим вимогам проектування, а також, для визначення особливостей його роботи.

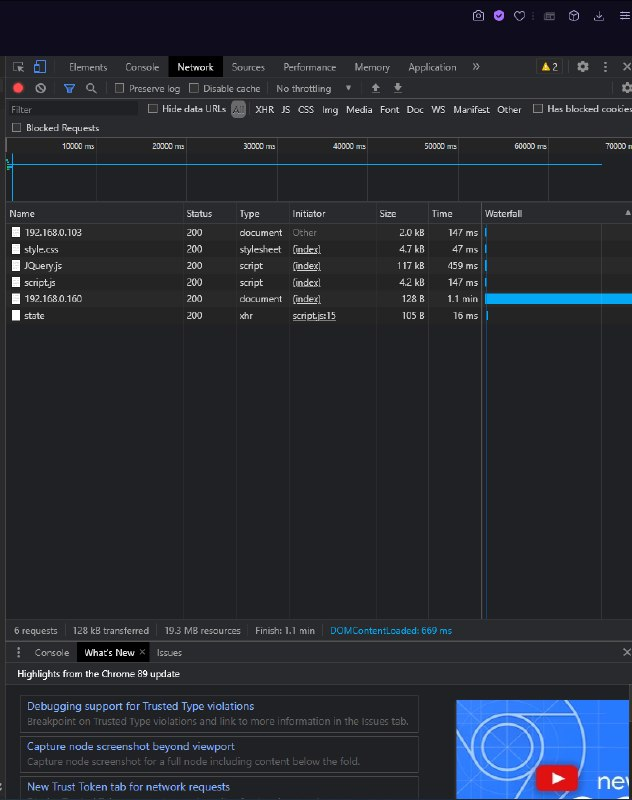


Рисунок 4.1 – Демонстрація швидкості завантаження інтерфейсу

Головним завданням тестування виробу на базі ESP32 є визначення швидкості і надійності WiFi-зв’язку між користувачем та самим мікроконтролером. Як вже зазначалось в цій роботі, користувач вводить IP-адресу пристрою в браузер. Браузер відсилає запит “/” на цю адресу в мережі. Мікроконтролер отримує цей запит і відсилає на IP-відправника, тобто пристрою, з якого зроблено запит, файл index.html, що є сайтом, що забезпечує Web-інтерфейс. Для функціонування інтерфейсу потрібно ще підвантажити ще файли: style.css, JQuery.js, script.js.

Демонстрацію завантаження цих файлів на комп’ютер, чи смартфон користувача подано на рис. 4.2. На ньому видно, що перший запит, що відсилає текст “/” під назвою “192.168.0.103” отримує відповідь і завантажується файл index.html за 147 мс. Аналогічно, style.css завантажується за 47 мс, JQuery.jsза 459 мс, script.jsза 147 мс. Також, на рисунку видно запит “192.168.0.160” – це адресація на WiFi-камеру. У випадку з камерою, відповідь на запит йде безперервно, на рис. 4.2 1,1 хв. Відповідь від камери є відеопотоком. Ще залишився запит “state”, відповідь на який прийшла за 16 мс. Як зазначено в додатку Б, це запит стану, ціль якого донести до Web-інтерфейсу стан системи, тобто чи рухається каретка крана і в якому напрямку.

Загалом, в демонстративному рисунку видно, що загальний час завантаження всіх файлів Web-інтерфейсу зайняло 800 мс, перш ніж користувач зміг приступити до роботи. Слід зазначити, що швидкість завантаження Web-інтерфейсу залежить не лише від мікроконтролера ESP32, а також і від якості WiFi-мережі, її завантаження. Було також проведено ще ряд тестувань швидкості завантаження Web-інтерфейсу на різних комп’ютерах, смартфонах, в різних мережах. Загальний час завантаження перебував в діапазоні 700-1000 мс.

Наступним кроком тестування є перевірка швидкості спрацьовування систем керування крану. Цей час також полягає в швидкості виконання запиту керування від Web-інтерфейсу самим мікроконтролером ESP32.

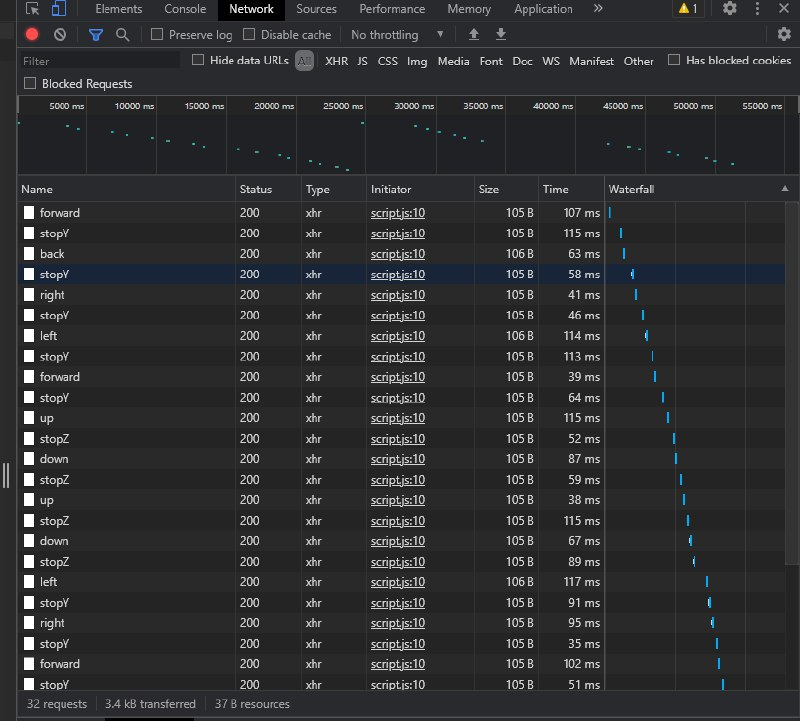


Рисунок 4.2 – Демонстрація швидкості виконання запитів керування



Рисунок 4.3 – Гістограма швидкості запитів

На рис. 4.2 надано демонстрацію швидкості виконання запитів керування. Було зроблено 48 запитів і записано скільки часу витрачається на їх виконання. Максимальний час виконання був 129 мс, в той час, як мінімальний 35 мс. Отже, діапазон часу виконання досить широкий. Також виміряно середній час виконання, який склав 78 мс.

На рис. 4.3 наведено гістограму, що показує який час виконання зустрічався найчастіше. На цьому рисунку видно, що більшість запитів виконувались до 120 мс, з яких 19 в проміжку від 35мс до 63мс, 11 в діапазоні 63мс-91мс, та 17 в проміжку 91-119мс. Отже, загалом, час виконання не перевищує 120 мс.

Також, слід зазначити, що, як і в ситуації з завантаження файлів Web-інтерфейсу, швидкість виконання запитів залежить в першу чергу від якості бездротового зв’язку та завантаженості Web-мережі.

## **4.2 Демонстрація кінцевого продукту**

Готовий виріб складається з самого мікроконтролера ESP32, під’єднаного до нього LCD дисплею, з корпусу, виготовленого на 3d-принтері спеціально для цього проекту, та з екрану з оргскла.

Для виготовлення корпусу було створено 3d-модель цього корпусу у комп’ютерному додатку 3DBuilder. Корпус має такі параметри: висота 32 мм, ширина 78 мм, довжина 98 мм, товщина стінок 2 мм, товщина днища 4 мм. Такі габарити було обрано, враховуючи розміри мікроконтролера та дисплея. Нижче на рис. 4.4 наведено 3d-модель корпусу. В внутрішній частині по кутах є стійки для встановлення апаратної частини. Також, в нижній стінці робиться отвір для проводів від мікроконтролера до реле керування, що в середині старого пульту керування.

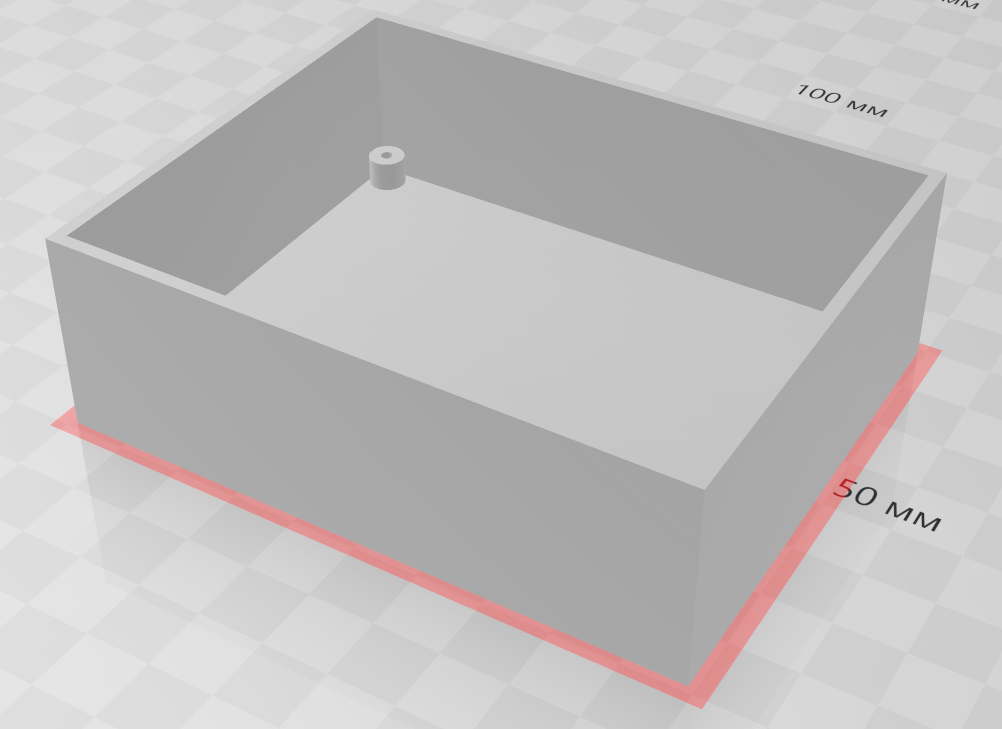


Рисунок 4.4 – 3d-модель корпусу

Програмно-апаратний комплекс встановлюється на бокову стіну старого пульту керування навчальною установкою мостового крану. На стінці пульта також просвердлюють отвори для проводів, що з’єднують керуючі виходи з реле керування. Сам же модуль приклеюється до старого пульта керування двосторонньою клейкою стрічкою.

Програмно апаратний комплекс без корпусу видно на рис. 4.4, встановлений на старий пульт керування видно на рис. 4.5.



Рисунок 4.5 – Програмно-апаратний комплекс без корпуса



Рисунок 4.6 – Встановлений програмно-апаратний комплекс на старому контролері навчального мостового крану

Для модуля WiFI-камери також було виготовлено корпус з пластику на 3d-принтері. Камеру було встановлено на кран, так, щоб було видно всю робочу поверхню. Напруга живлення камери 5V, по живиться по USB або від акумулятора, або від блока живлення. Нижче на рис. 4.7 видно встановлену камеру на крані.

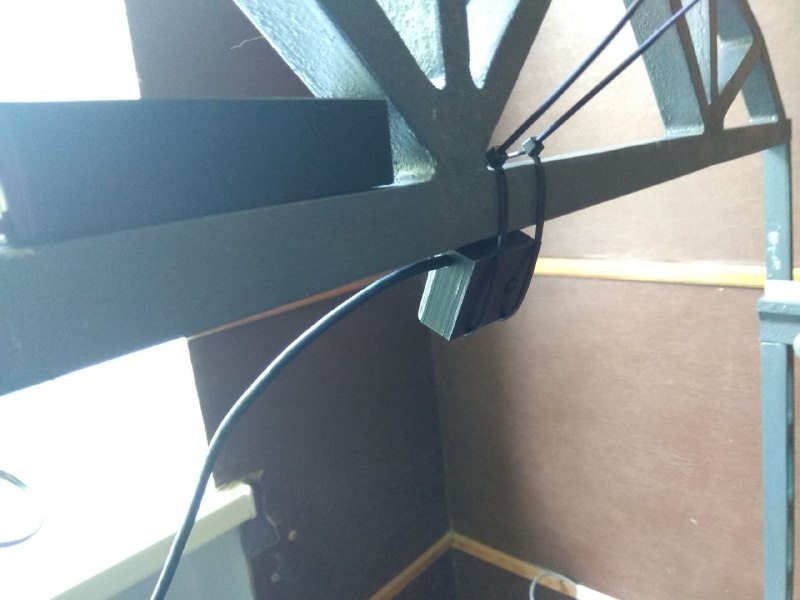


Рисунок 4.7 – Встановлена на крані камера

# **ВИСНОВКИ**

Результатом дипломного проектування є модернізація навчальної установки мостового крану завдяки створенню і встановленню програмно-апаратного комплексу на базі мікроконтролера ESP32. Цей комплекс підключається до WiFI мережі і дозволяє дистанційно, зручно і ефективно керувати навчальною установкою. В пам’яті мікроконтролера містяться файли Web-інтерфейсу користувача, які при підключенні до ESP32 завантажуються. Також, програмно-апаратний комплекс містить набір команд реагування на запити від користувача.

У першому розділі дипломної роботи наведено загальну інформацію про кранові установки, їх класифікацію. Описано особливості системи керування мостового крану, описано склад кабіни оператора. Розглянуто і проаналізовано деякі системи дистанційного керування кранами: дротові та бездротові. Нарешті, виділено ряд вимог, що мають бути в створеному програмно-апаратному комплексі.

У другому розділі описано процес проектування виробу, завдяки якому модернізується навчальна установка. Починається все з постановки задачі проектування, потім йде проектування апаратної частини комплексу, де вирішується що вона повинна робити та з яких функціональних модулів складатися. Далі йде проектування прошивки – програми, яка визначає алгоритм та принципи роботи апаратної частини.

В третьому розділі описано процес реалізації програмно-апаратного комплексу на базі створеного проекту. Спочатку визначається алгоритм, за яким система буде працювати. Сам процес реалізації починається зі створення архітектури, де визначаються особливості програмно-апаратного комплексу, як функціональні модулі будуть взаємодіяти. Після цього наводяться технічні особливості пристрою: використані протоколи, напруга живлення. Також наводяться особливості його складових частин. Нарешті, йде реалізація прошивки, тобто написання програми, що буде завантажено в апаратну частину комплексу, тобто в мікроконтролер ESP32. Також описується сам код, використані бібліотеки, змінні, функції та описується алгоритм роботи прошивки.

В результаті були зроблені висновки.

1. Модернізація застарілої техніки на базі нових технології є можливою та ефективною.
2. Використання бездротових технології передачі даних мікропроцесорною технікою дозволяє спростити використання, проектування та реалізацію, та розширити функціонал програмно-апаратного комплексу.
3. Розроблений комплекс дозволяє використовувати як смартфони, так і ПК для керування крановою установкою. Фактично керування можливе будь-яким пристроєм, що може бути частиною WiFi-мережі та має браузер.

Програмно-апаратний комплекс створено для модернізації навчальної установки, але це не заважає його адаптувати і до робочих машин. Саме цей комплекс створено для використання в навчальних установах.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. ДСТУ 2986-95 Крани вантажопідіймальні. Частина 1. Терміни та визначення основних понять. [Чинний від 1996-01-01] – Київ: Держстандарт України. – 71 с.
2. Кабины управления мостовых кранов [Електронний ресурс] – Режим доступу до ресурсу: https://kramos-crane.ru/kranovoe-oborudovanie/kabiny.
3. Протасов Н. Гидроманипуляторы [Електронний ресурс] / Николай Протасов – Режим доступу до ресурсу: https://os1.ru/article/6864-gidromanipulyatory-za-besprovodnym-upravleniem-budushchee.
4. Telecrane F21 Series and F24 Series Radio Remote Control Manual, 2006.
5. F24-6S specifications [Електронний ресурс] – Режим доступу до ресурсу: http://telecrane.cl/img/cms/descargaspdfs/f24-6Dspecifications%20EN.pdf.
6. Система радиоуправления FST 516 micron 5 Ex [Електронний ресурс] – Режим доступу до ресурсу: https://crancontrol.ru/upravlenie-vzi/fst-516-micron-5-ex/.
7. Technische Spezifikationen: PV9E30B4444 [Електронний ресурс] // EMAS – Режим доступу до ресурсу: https://asset.conrad.com/media10/add/160267/c1/-/de/001229443DS01/datasheet-1229443-emas-pv9e30b4444-pushbutton-1-pcs.pdf.
8. STM32 32-bit Arm Cortex MCUs [Електронний ресурс] // STmicroelectronics – Режим доступу до ресурсу: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html.
9. SoCs ESP32 [Електронний ресурс] // Espressif – Режим доступу до ресурсу: https://www.espressif.com/en/products/socs/esp32.
10. Baker S. Panel Technologies TN Film, MVA, PVA and IPS Explained / Simon Baker., 2011. Режим доступу до ресурсу: https://web.archive.org/web/20110717123858/http://www.tftcentral.co.uk/articles/content/panel\_technologies\_content.htm
11. Matoušek T. I2C bus Inter Integrated Circuits bus / Tomáš Matoušek., 2000. – (Philips Semiconductors).

# **ДОДАТОК А**

# **ЛІСТИНГ КОДУ ПРОШИВКИ**

//підключення бібліотек

#include "WiFi.h"

#include "ESPAsyncWebServer.h"

#include "SPIFFS.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

//визначення передпроцесорних макросві

#define axisX 13

#define axisY 4

#define axisZ 2

#define dirX 12

#define dirY 14

#define dirZ 27

//ініціальзіція констант з SSID та паролем WiFi мережі

const char\* ssid = "\*\*\*\*\*\*";

const char\* password = "\*\*\*\*\*\*\*\*";

//байт в який записується стан крану: рух і напрям по осям

byte state = 0;

//ініціалізація прийому/передачі по 80 порту

AsyncWebServer server(80);

//ініціалізація lcd-дисплея,

//виходи не задаються, бо овни апаратно вже визначені

//SCL - GPIO 22 SDA - GPIO 21

//адреса дисплея 0х27

LiquidCrystal\_I2C lcd(0x27, 16, 2);

//етап запуску

void setup()

{

//запуск послідовного порту

Serial.begin(9600);

//ініцілізація пам’яті, в яку записаний сайт: html, css, js

//якщо не вдалося, по виведеться помилка в послідовний порт

//і система не буде працюівати

if(!SPIFFS.begin(true))

{

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

//назначення керуючих пінів на вихід

pinMode(axisX,OUTPUT);

pinMode(axisY,OUTPUT);

pinMode(axisZ,OUTPUT);

pinMode(dirX,OUTPUT);

pinMode(dirY,OUTPUT);

pinMode(dirZ,OUTPUT);

//назначення низького початкого рівня

//напруги на виходах

digitalWrite(axisX, LOW);

digitalWrite(axisY, LOW);

digitalWrite(axisZ, LOW);

digitalWrite(dirX, LOW);

digitalWrite(dirY, LOW);

digitalWrite(dirZ, LOW);

//ініціалізація WiFI модуля та спроба

//підключення до мережі за SSID та паролем

//Поки не буде підключення, в послідовний

//буде виводитись символ ".", що показує

//процес підключення. Під час підключення

//більне ніяких дій не виконує

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

//нижче написані запити, які контролер буде

//сприймати, та дії, які він буде робити

//при отриманні запита

//при отриманні запита "/", що означає, що

//користувач звернувся за IP-адресою до ESP32,

//контролер відсилай index.html файл

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/index.html");

});

//при отриманні запиту "/style.css"

//відсилається файл style.css

server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/style.css");

});

//при отриманні запиту "/script.js"

//відсилається файл script.js та JQuery.js

server.on("/script.js", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/script.js");

});

server.on("/JQuery.js", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/JQuery.js");

});

server.on("/first\_letter.png", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/first\_letter.png");

});

server.on("/background.jpg", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/background.jpg");

});

server.on("/motionXY.png", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/motionXY.png");

});

server.on("/motionZ.jpg", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/motionZ.jpg");

});

//нижче написані дії є аналогічними,

//при отриманні запитів "/up","/down","/left","/rigth","/back","/forward"

//ESP32 подає високий рівен ва вихід

//та змінює байт стану state

server.on("/up", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00001001;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, HIGH);

digitalWrite(dirZ, HIGH);

});

server.on("/down", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000001;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, HIGH);

digitalWrite(dirZ, LOW);

});

server.on("/left", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00010010;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, HIGH);

digitalWrite(dirY, HIGH);

});

server.on("/right", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000010;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, HIGH);

digitalWrite(dirY, LOW);

});

server.on("/forward", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00100100;

request->send(200, "text/plain",String(state));

digitalWrite(axisX, HIGH);

digitalWrite(dirX, HIGH);

});

server.on("/back", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state |=0b00000100;

request->send(200, "text/plain",String(state));

digitalWrite(axisX, HIGH);

digitalWrite(dirX, LOW);

});

//при отриманні запиту "/stopZ"

//зупиняється рух по Z та облулюється стан

server.on("/stopZ", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state = 0;

request->send(200, "text/plain",String(state));

digitalWrite(axisZ, LOW);

digitalWrite(dirZ, LOW);

});

//при отриманні запиту "/stopy"

//зупиняється рух по X,Y та облулюється стан

server.on("/stopY", HTTP\_GET, [](AsyncWebServerRequest \*request)

{

state = 0;

request->send(200, "text/plain",String(state));

digitalWrite(axisY, LOW);

digitalWrite(axisX, LOW);

digitalWrite(dirY, LOW);

digitalWrite(dirX, LOW);

});

//при отримані запиту "/state"

server.on("/state", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(200, "text/plain",String(state));

});

//запуск сервера

server.begin();

//вмикання дисплею та його початковы налаштування

lcd.init();

lcd.clear();

lcd.backlight();

lcd.setCursor(0, 0);

//виведення на дисплей ip адреси, що видав сервер

lcd.print(WiFi.localIP());

}

void loop(){}

# **ДОДАТОК Б**

# **ТАБЛИЦЯ ЗАПИТІВ**

|  |  |
| --- | --- |
| Текст запиту | Дія |
| / | Esp32 відсилає на IP відправника index.html файл |
| /style.css | Esp32 відсилає на IP відправника файл style.css |
| /script.js | Esp32 відсилає на IP відправника файл script.js |
| /JQuery.js | Esp32 відсилає на IP відправника файл first\_letter.js |
| /first\_letter.png | Esp32 відсилає на IP відправника .png файл |
| /background.jpg | Esp32 відсилає на IP відправника файл background.jpg |
| /motionXY.png | Esp32 відсилає на IP відправника файл motionXY.png |
| /motionZ.jpg | Esp32 відсилає на IP відправника файл motionZ.jpg |
| /up | Починається рух каретки вверх |
| /down | Починається рух каретки вниз |
| /left | Починається рух каретки вліво |
| /right | Починається рух каретки вправо |
| /forward | Починається рух каретки вперед |
| /back | Починається рух каретки назад |
| /stopZ | Зупиняється рух по Z |
| /stopY | Зупиняється рух по X,Y |
| /state | Відсилається байт стану |