3. СТРУКТУРА ТА ОПИС РОБОТИ МОДУЛІВ ПРОГРАМНОГО І АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вибір апаратної бази пристрою

При виборі апаратної бази для обраної задачі магістерської дисертації, а саме, розробка цифрового оптичного спектрометра, було проаналізовано апаратні модулі за такими критеріями:

* ціна апаратної архітектури;
* потужність апаратної архітектури;
* програмна підтримка апаратного забезпечення (доступність програмування через офіційні IDE, підтримка плат операційними системами);
* складність реалізації (в залежності від вибору плати, змінюється складність реалізації — простим прикладом є мікроконтролер Atmega8 від компанії Atmel. Його можна напряму програмувати використовуючи Асемблерні команди. Це відкриває практично необмежені можливості, але паралельно з тим збільшує складність написання програм в десятки разів. Навіть проста програма моргання світлодіодом займає досить багато часу на реалізацію).

У виборі основної плати з додатковою периферією (ядро пристрою, яке збирає дані з сенсорів, робить математичні обрахунки, спілкується з комп’ютером пакуючи пакети і пересилаючи їх через протокол передачі даних) було розглянуто такі плати: Arduino Mega плата на базі мікроконтролера Atmega2560, Arduino Uno плата на базі мікроконтролера Atmega328, STM32 плата на базі мікроконтролера STM32F411CEU6(BlackPill edition).

Таблиця 3.1.1 — Порівняльна характеристика плат для ядра пристрою.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва плата | Ціна | Програмна підтримка | Складність реалізації | Потужність |
| Arduino Mega | 360 грн | Дуже хороша програмна підтримка. Є офіційне програмне забезпечення для прошивання, програмування та налагодження. Вагомим плюсом є наявність бібліотек, які можна завантажити і підключити для роботи з різними пристроями. Можливість розгорнути мікроопераційну систему FREERTOS для багатопоточного програмування. | відносно просто | Набагато потужніша якщо порівнювати із Arduino Uno, але поступається STM32. Набір периферій значно більший в порівнянні з Uno, але є суттєво менший від STM32. |
| Arduino Uno | 180 грн | Дуже хороша програмна підтримка. Є офіційне програмне забезпечення для прошивання, програмування та налагодження. Вагомим плюсом є наявність бібліотек, які можна завантажити і підключити для роботи з різними пристроями. | відносно просто | Є самою обмеженою в порівнянні з двома іншими представниками. Набір периферії також найменший. |
| STM32 | 200 грн | Хороша програмна підтримка. Є офіційне програмне забезпечення для прошивання, програмування та налагодження. Один із вагомих мінусів - потрібно вручну працювати із пристроєм, який підключаєш (немає набору бібліотек для роботи з пристроями), тай в загальному з платою. | дуже складно, 32 розрядна архітектура, тобто складність використання бібліотек для Arduino | Є самою потужною з представлених плат і має велику кількість периферій (UART, I2C, SPI, CAN тощо). |

Дивлячись на порівняльну таблицю, по співвідношенню параметрів один до одного, найкращим варіантом є STM32, але дуже складна реалізація виявилася блокером для вибору даної архітектури. Тому було обрано наступну по співвідношенню параметрів, а саме Arduino Mega. Вона сама дорожча по ціні серед представлених плат, але дозволяє реалізувати пристрій без будь-яких обмежень.

Серед RGBC модулів (сенсорів) вибір не є великим. Єдиний і найкращий вибір, який надає можливість виміряти світло різної довжини хвилі — APDS-9960. Даний сенсор є універсальним модулем, що покриває вимоги для вимірювання світла при досліджені поверхні напівпровідникових структур за допомогою оптичного спектрометра.

Кількість сенсорів, які входять до складу цифрового оптичного спектрометра, є дві одиниці. APDS-9960 під’єднуються до Arduino Mega за допомогою I2C шини (піни SDA та SCL відповідають за це). При під’єднані двох однакових модулів до I2C шини, виникає конфлікт комунікації, оскільки однакові девайси мають однакову адресу в шині. Для рішення цієї проблеми, було вирішено використовувати мультиплексор для I2C шини. Найпопулярніший I2C мультиплексором є TCA9548A. За своїми властивостями TCA9548A є восьмиканальним мультиплексором, що відповідно дозволяє під’єднати до восьми I2C девайсів з однаковими адресами. В майбутньому мультиплексор дозволе зробити модифікації пристрою із дисплеєм, кнопками і іншими доповненнями, які дадуть змогу зробити незалежний від комп'ютера пристрій.

Зв’язок із комп’ютером здійснюється за допомогою UART модуля, який вже вбудований в плату Arduino Mega. За своєю структурою UART девайс містить два протилежні канали — Tx і Rx, відповідно для передачі і отримання з UART девайса. Зі сторони комп’ютера, Arduino Mega приєднується за допомогою USB і відображається як COM порт. При програмуванні програми на стороні комп’ютера, для будь-якої мови програмування є можливість підключити бібліотеку для зчитування і передачі даних для COM порта.

3.2. Вибір програмної бази пристрою

Програмна база пристрою була вибрана, переважно, відповідно до обраної апаратної архітектури. Розробка програми відбулася на ОС Linux, але програма є кросплатформена, відповідно до чого, може з легкістю портуватися на ОС Windows.

Оскільки ядром пристрою було обрано плату Arduino Mega, програмне забезпечення для плати відбувалося в середовищі Arduino IDE. Це універсальне середовище розробки, яке надає можливість розробляти, прошивати та налагоджувати програмне забезпечення для всіх Arduino плат і не тільки (також можна використовувати для STM32, за умови завантаження додаткових бібліотек, які розраховані на архітектуру 8 біт). Мова програмування програмного середовища — С++. Відповідно ця високорівнева мова програмування надає можливість програмувати використовуючи ООП (сучасний підхід програмування). Arduino IDE дозволяє завантажити бібліотеку практично для кожного під’єдного пристрою, що дозволяє використовувати інтерфейси для управління пристроями, які були написані користувачами і додано до Arduino бібліотеки для загального користування. Для використання восьмиканального мультиплексора TCA9548A і RGBC сенсора APDS-9960, було завантажено і застосовано додаткові бібліотеки, які спрощують ініціалізацію пристроїв та їхнє керування.

Зі сторони комп'ютера програма, яка спілкується з Arduino Mega через COM порт, обраховує отримані результати, будує графіки реалізована на мові програмування C++ за допомогою Qt фреймворку. Qt фреймворк являє собою потужну систему, яка має бібліотеки для всіх можливих варіантів програмного забезпечення. Що і було використано для комунікації з COM портом, побудови графіків і обрахунку математичних даних.

3.3. Опис апаратної архітектури пристрою

Як було описано в пункті 1 цього розділу, основою пристрою було обрано плату Arduino Mega 2560 (ядро системи), для підключення двох і більше I2C девайсів було обрано мультиплексор TCA9548A, а для вимірювання світлових спектрів, було обрано RGBC сенсори — APDS-9960.

Arduino Mega 2560 являє собою пристрій на основі мікроконтролера ATmega2560. В склад пристрою входять: п’ятдесят чотири цифрових входів/виходів (з яких п’ятнадцять можуть бути ШІМ виходами), шістнадцять аналогових входів, чотири UART (апаратних приймально-передаючих модулів, які реалізують інтерфейси послідовної передачі даних), кварцовий резонатор на 16 МГц, USB роз’єм, роз’єм живлення, роз’єм ICSP для програмування внутрішньої схеми і кнопка скидання системи. Однією з особливостей Arduino Mega 2560 є те, що для забезпечення зв'язку з комп'ютером через інтерфейс USB-UART вона використовує мікроконтролер ATmega16U2. У попередніх версіях плати використовувалась мікросхема FTDI. Мікроконтролер ATmega16U2 відрізняється від мікросхеми FTDI тим, що дозволяє виконувати різноманітні операції, такі як перепрограмування інтерфейсу, який використовується для з'єднання з комп'ютером, а також може використовуватись для управління різними периферійними пристроями, такими як дисплеї, сенсори, інші мікроконтролери тощо.

Таблиця 3.3.1 — Характеристик Arduino Mega 2560

|  |  |
| --- | --- |
| **Характеристика** | **Значення** |
| Мікроконтролер | ATmega2560 |
| Робоча напруга | 5В |
| Напруга живлення (рекомендоване) | 7-12В |
| Напруга живлення (межа) | 6-20В |
| Цифрові входи/виходи або порти | 54 |
| Аналогові входи | 16 |
| Максимальний струм одного порту | 40 мА |
| Максимальний вихідний струм порту 3.3 В | 50 мА |
| Флеш пам’ять | 256 КБ, з яких 8 КБ використовується завантажувачем |
| SRAM | 8 КБ |
| EEPROM | 4 КБ |
| Тактова частота | 16 МГц |

I2C мультиплексор (комутатор, розширювач) TCA9548A призначений для вирішення проблеми підключення декількох I2C пристроїв з однаковими адресами або різними рівнями логічних сигналів до одного мікроконтролера. Цей пристрій дозволяє повністю контролювати процес доступу ведучого пристрою до введених на I2C шині пристроїв. Використання мультиплексора дозволило задіяти в програмній частині відомий паттерн програмування Фасад.

Мультиплексор підключається через I2C шину до мікроконтролера (ведучого пристрою), а датчики та модулі (введені пристрої) підключаються до портів мультиплексора. Таким чином, мультиплексор виступає посередником у взаємодії між ведучим та веденими пристроями. Для початку необхідно звернутися до адреси мультиплексора (за замовчуванням 0x70) та записати до регістру конфігурації мультиплексора номер порту, з яким необхідно працювати. Можна вказати одразу кілька портів. Номер активного порту визначається одиницею в відповідному розряді регістру. Наприклад, 00000100 означає, що активним стане третій порт. Далі робота здійснюється так само, як якщо ведене пристрій було б підключено безпосередньо до мікроконтролера. Тобто можна використовувати звичайні бібліотеки та звертатися за адресою веденого пристрою (дисплея, датчика тощо). Якщо необхідно переключитися на інший пристрій, для цього достатньо просто записати в регістр мультиплексора номер потрібного порту. Також варто звернути увагу, що якщо адреса якого-небудь периферійного пристрою збігається з адресою мультиплексора, то можна змінити адресу мультиплексора за допомогою портів А1, А2, А3 на платі мультиплексора.

Цифровий датчик для безконтактного виявлення жестів, наближення, RGB компонентів кольору та рівня освітленості навколишнього простору на чіпі APDS-9960. Фоточутлива частина датчика складається з одного потужного інфрачервоного світлодіода та чотирьох фотодіодів. Обмін даними з чіпом APDS-9960 реалізований через шину I2C.

Таблиця 3.3.2 — Характеристик APDS-9960

|  |  |
| --- | --- |
| **Характеристика** | **Значення** |
| Напруга живлення | 2.4-3.6В |
| Логічні рівні | 3.3 В |
| Інтерфейс | I2C (адреса 0x39) |
| Максимальна швидкість обміну | 400 КГц |
| Робочий діапазон дистанції | до 20 см |
| Робоча температура | -30-80 °C |
| Розмір плати | 15х20 мм |

Принцип з’єднання пристрою наступний: Arduino Mega 2560 з’єднана з комп'ютером за допомогою USB (комунікує і живиться), пін Arduino 5V з’єднаний з входами VIN мультиплексора TCA9548A та сенсорів APDS-9960 (живлення 5 В), пін Arduino GND з’єднаний з входами GND мультиплексора TCA9548A та сенсорів APDS-9960 (земля - нульовий потенціал), пін Arduino 20 (шина SDA протоколу I2C) з’єднаний з входом SDA мультиплексора TCA9548A, а пін Arduino 21 (шина SCL протоколу I2C) з’єднаний з входом SCL мультиплексора TCA9548A, виходи мультиплексора TCA9548A SD1 і SC1, SD2 і SC2 з’єднані відповідно з входами SDA та SCL двох сенсорів APDS-9960, виходи SD0 і SC0 зарезервовані на майбутнє для під’єднання I2C дисплея.

Схема з’єднання апаратної архітектури для цифрового оптичного спектрометра зображена на рисунку 3.3.1.

|  |
| --- |
|  |
| Рисунок 3.3.1 — Схема з’єднання апаратної архітектури для цифрового оптичного спектрометра |

3.4. Опис програмної архітектури пристрою

Початком програмної архітектури є програма розроблена за допомогою Qt фреймворку за допомогою якої комп’ютер спілкується з Arduino Mega приєднуючись до COM порту та пересилаючи шістнадцяти-байтні пакети через UART протокол. Вигляд Qt терміналу зображений на рисунку 3.4.1.

|  |
| --- |
|  |
| Рисунок 3.4.1 — Вигляд Qt терміналу. |

Відносно конфігурацій терміналу — майже всі з них належать до налаштування передачі через UART: Port — відповідний порт, через який відбувається передача; Baud Rate — швидкість передачі даних, мається на увазі, що число яке записане в цю конфігурацію відповідає максимальній передачі в бітах за секунду (біт/сек), межі Baud Rate від 50 до 4000000; Data Bits — число біт, які відведені на дані (допустимі значення: 5, 6, 7, 8); Stop Bit — число біт, які відведені на STOP frame (допустимі значення: 1, 1.5, 2); Parity — біт-паритет, який дозволяє перевірити правильність переданих даних на стороні отримувача (допустимі значення: No Parity — без цього біту, Even Parity — біти даних плюс Parity біт призводять до парної кількості бітів з одиницею, Mark Parity — біт завжди рівний одиниці, Odd Parity — біти даних плюс Parity біт призводять до непарної кількості бітів з одиницею, Space Parity — біт завжди рівний нулю); Flow Control — параметр, який визначає ким буде контролюватися кількість переданих даних (допустимі значення: No Flow Control, Hardware Flow Control, Software Flow Control). За допомогою кнопки Connect відбувається з’єднання з COM портом.

Після з’єднання з COM портом вся комунікація переходить в консоль терміналу (як це для прикладу в ОС Linux чи Command Prompt в ОС Windows). Команди для управління Arduino Mega, вивід інформації, обчислення інформації. Консоль виводить всю надіслану та отриману інформацію. На рисунку 3.4.2 показано приклад, як працює консоль терміналу.

|  |
| --- |
|  |
| Рисунок 3.4.2 — Приклад роботи консолі терміналу |

Приклад простої комунікації з Arduino Mega зображена на рисунку 3.4.3. Законектившись до COM порта (ttyUSB), встановивши Baud Rate 9600 (на стороні Arduino Mega це значення має бути таким же, в іншому випадку ми будемо отримувати невідомі символи), ми надсилаємо комаду “Hello”, яка відправляється на Arduino Mega через COM порт. На стороні Arduino Mega було додано простий скрипт, при якому, отримавши команду “Hello”, ми у відповідь відправляємо “Hello from Mega”.

|  |
| --- |
|  |
| Рисунок 3.4.3 — Приклад простої комунікації з Arduino Mega |

На боці Arduino Mega було розроблено програмну реалізацію за принципом ООП та стейт машини (Feature State Machine — FSM, відомий паттерн програмування Стейт). Стартує програма для Arduino Mega з типових для Arduino функцій setup() і loop(). Всередині setup() функції викликається метод класу megaMgr — initMega(). В якому відбувається ініціалізація всіх полів об’єкту, встановлюються початкові налаштування (Baud Rate тощо). Код реалізації функцій setup() і loop() зображено на рисунку 3.4.4.

|  |
| --- |
|  |
| Рисунок 3.4.4 — Код реалізації функцій setup() і loop() |

Всередині функції loop() відбувається перехід до нескінченного циклу, де відбувається обробка отриманих команд-івентів із Qt термінала, реагування на ці команди-івенти.

Для реагування Arduino Mega на команди-івенти було створено FSM, в якій відбувається зміна стейтів об’єкта класу megaMgr, виклик функцій після зміни стейту для обробки отриманих даних, конфігурування, збирання даних, пересилки отриманих даних через власний протокол через UART.

Принцип роботи FSM зображений на рисунку 3.4.5. Стартує система з стейту IDLE і очікує в ньому на будь-які івенти, які можуть прийти з Qt терміналу. Якщо приходить івент CONFIG\_EVENT, тоді ми переходимо в новий стейт CONFIG\_APDS\_MODULES, якщо приходи будь-який інший івент ми залишаємося чекати в IDLE стейті. Перейшовши в стейт CONFIG\_APDS\_MODULES ми починаємо конфігурувати APDS-ки викликом методу класу apdsConfiguration. Результатом даної функції є надсилання івента самому собі (для об’єкта класу megaMgr), якщо вдалося конфігурувати APDS-ки — CONFIG\_SUCCESS\_EVENT, якщо не вдалося конфігурувати APDS-ки — CONFIG\_FAILURE\_EVENT. При отримані CONFIG\_FAILURE\_EVENT івента ми переходимо в стейт IDLE і надсилаємо повідомлення до Qt терміналу про те, що не вдалося конфігурувати APDS-ки. При отриманні CONFIG\_SUCCESS\_EVENT, ми переходимо до нового стейту READY\_TO\_CAPTURE\_DATA і надсилаємо повідомлення про те, що вдалося конфігурувати APDS-ки. У стейті READY\_TO\_CAPTURE\_DATA, фітча очікує на івенти з Qt терміналу, якщо приходить CAPTURE\_DATA\_APDS1\_EVENT, збираються дані із APDS1, надсилається івент самому собі (для об’єкта класу megaMgr) SEND\_CAPTURED\_DATA\_EVENT і фітча переходе в новий стейт DATA\_CAPTURED, якщо приходить CAPTURE\_DATA\_APDS2\_EVENT, збираються дані із APDS2, надсилається івент самому собі (для об’єкта класу megaMgr) SEND\_CAPTURED\_DATA\_EVENT і фітча переходе в новий стейт DATA\_CAPTURED. У новому стейті DATA\_CAPTURED, якщо фітча отримує SEND\_CAPTURED\_DATA\_EVENT, фітча переходе в стартовий стейт IDLE і пересилає зібрані дані з APDS модуля до Qt терміналу.

|  |
| --- |
|  |
| Рисунок 3.4.5 — FSM для об’єкту класу megaMgr |

Передача будь-яких даних від Qt терміналу до Arduino Mega і навпаки — від Arduino Mega до Qt терміналу відбувається за допомогою власного шістнадцяти-байтного протоколу. Протокол дозволяє передавати беззнакові числа різних довжин (1, 2, 4 байти). Для цього створена конфігураційна таблиця, яка дозволяє задати максимальну довжину числа, яке передається, його положення в протоколі (offset), ім’я сигналу, яке відповідає числу та інші параметри (divider for ADC, float coefficient тощо). Конфігураційна таблиця була створена у відповідності до рекомендацій паттерна програмування Стратегія. На рисунку 3.4.6 зображена конфігураційна таблиця шістнадцяти-байтного протоколу для Arduino Mega.

|  |
| --- |
|  |
| Рисунок 3.4.6 — Конфігураційна таблиця шістнадцяти-байтного протоколу для Arduino Mega |

Відповідно, для того, щоб розпарсити отримані дані, конфігураційні таблиці передавача і отримувача повинні співпадати, в іншому випадку, отримані дані будуть розшифровані не вірно. Механіка пакування та розпакування організовані у відповідності до паттерну програмування Стратегія.

На рисунку 3.4.7 зображена структура пакета протокола. Відповідно до зображення, нульовий байт пакету — START\_FRAME (значення якого 0x3A), перший байт пакету — INSTRUCTION\_FRAME. Цей байт відповідає за тип інструкції, яка пересилається. Можна просто переслати інструкцію, як це робиться для прикладу для команди-івента CONFIG\_APDS\_EVENT, а можна задати тип інструкції і запакувати дані (як це робиться при пересиланні даних, які були зняті з APDS модуля). Тоді на стороні одержувача, коли він одержав цю інструкцію, він використовує конфігураційну таблицю, щоб розшифрувати отримані дані. Приклад можливих інструкцій: CONFIG\_INSTRUCTION (значення якого 0xF1), CONFIG\_SUCCESS\_INSTRUCTION (значення якого 0xF2), CONFIG\_FAILURE\_INSTRUCTION (значення якого 0xF3), CAPTURE\_APDS1\_INSTRUCTION (значення якого 0xF4), CAPTURE\_APDS2\_INSTRUCTION (значення якого 0xF5) APDS\_DATA\_INSTRUCTION (значення якого 0xF6), HELLO\_INSTRUCTION (значення якого 0xF6). Далі, з другого по тринадцятий байти, містяться зашифровані дані (якщо вони були додані). В передостанньому і останньому байтах містяться STOP\_FRAME (значення якого 0x0D) і END\_FRAME (значення якого 0x0A) відповідно.

|  |
| --- |
|  |
| Рисунок 3.4.7 — Структура пакета протоколу. |

Функція для пакування даних у шістнадцятибайтний пакет протоколу зображена на рисунку 3.4.8. Принцип роботи заключається в трансформуванні числа у біти та заповнені відведених байтів у пакеті, сформованими бітами.

|  |
| --- |
|  |
| Рисунок 3.4.8 — Функція для пакування даних у шістнадцятибайтний пакет протоколу |

Функція для розпакування даних з шістнадцятибайтного пакету протоколу зображена на рисунку 3.4.9.

|  |
| --- |
|  |
| Рисунок 3.4.9 — Функція для розпакування даних з шістнадцятибайтний пакет протоколу |

3.5. Висновки до розділу

1. У розділі було детально описано вибір апаратної бази пристрою, а саме чому Arduino Mega — це найкращий варіант для ядра системи. Причинами цього є потужність плати, великий функціонал, одна з найкращих програмних підтримок плати (Arduino IDE, бібліотеки для будь-якого пристрою, розроблені користувачами, і додані до онлайн бібліотеки). Можливість багатопоточного програмування при використанні мікро-операційної системи FREERTOS.
2. Також було розглянуто питання, чому важливо розділити I2C модулі від ядра системи Arduino Mega. Це робить девайс універсальним і дозволить в майбутньому його покращити, додавши для прикладу I2C дисплей.
3. Вибір програмної бази тісно пов’язаний із вибором апаратної бази, за допомогою Arduino IDE програмування мікроконтролера відбувається мовою програмування С++. Зі сторони комп'ютера, було обрано Qt фреймворк для розробки терміналу, який надасть змогу керувати оптичним спектрометром, і в загальному програмним процесом, використовуючи команди, на зразок того, як це відбувається в операційній системі Linux (Linux Terminal) або Windows (Windows Command Prompt).
4. Детально описано кожен із апаратних пристроїв: Arduino Mega, I2C мультиплексор TCA9548A і сенсори APDS-9960, особливості використання та програмування, технічні характеристики.
5. Зі сторони програмної архітектури, було описано детально роботу Qt терміналу, доступні команди для управління Arduino Mega плати і під’єднаних до неї пристроїв (TCA9548A і APDS-9960). Представлено простий приклад комунікації між Arduino Mega і Qt терміналу, на прикладі простого скрипта. Представлено детальний опис програмного забезпечення для управління Arduino Mega (megaMgr), який включає в себе стандартні для всіх Arduino проектів функції setup() та loop(), в яких відповідно відбувається ініціалізація і головний цикл програми. Управління Arduino Mega відбувається за допомогою стейт машини, яка працює в залежності від отриманих івентів, що є у свою чергу хорошим кейсом не тільки для продуктової розробки, але й в загальному для програмування.
6. Важливою деталлю програмного забезпечення, на яку слід звернути увагу — власне-розроблений шістнадцятибайтний протокол для передачі даних від Qt терміналу до Arduino Mega і навпаки. Розглянуто структуру пакета протоколу, і, також, принцип шифрування-дешифрування даних пакету протокола.