КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«ЛІЧИЛЬНИК ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ З ПЕРЕДАЧЕЮ ДАНИХ ПО WI-FI»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Олена Тягур _____

Науковий керівник:
кандидат фізико-математичних наук, асистент
Сергій Фесенко _____

Рецензент:
кандидат фізико-фатематичних наук, асистент
Юрій Лень

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра за об’ємом складає 45 сторінок, містить 27 рисунків, 3 додатки, використано 9 інформаційних джерел.

Об’єктом роботи є демонстрація принципу роботи лічильника електричної енергії з передачею даних по Wi-Fi, який здійснює моніторинг спожитої електроенергії. Предметом роботи є створений та запрограмований мною пристрій, що має змогу передавати дані щодо споживання електроенергії з допомогою бездротової мережі.

Головною метою є розробка приладу, що відповідає створеній принциповій електричній схемі на основі мікроконтролерів ESP8266 та Arduino Uno.

Інструменти розроблення: програмне середовище розробки Arduino IDE, що використовує мову програмування C++, що власне і надало змогу запрограмувати розроблений пристрій для коректної роботи.

МІКРОКОНТРОЛЕР, НАПРУГА, СТРУМ, ЕЛЕКТРОЕНЕРГІЯ, WI-FI, ЛІЧИЛЬНИК, ПРОТОКОЛ, ЕЛЕКТРИЧНА СХЕМА.

ЗМІСТ

ВСТУП

У зв’язку з подіями, що відбувалися з об’єктами критичної інфраструктури останнього року, енергосистема країни постраждала та потребує відновлення, тому може виникати потреба у використанні резервного живлення, зокрема безперебійних акумуляторних джерел живлення. У такому випадку, постає питання щодо оптимізації енергоспоживання окремих приладів.

Для вирішення вказаної задачі оптимізації можна використовувати представлений у даній роботі лічильник електричної енергії, який дозволяє не лише вимірювати і передавати на сервер для подальшої обробки дані щодо споживання, а й може вимикати споживача, якщо він перевищує заданий ліміт енергоспоживання або його струм надто високий.

Оскільки даний лічильник здатен відслідкувати постійну складову струму, то його можна використовувати як прилад контролю навантаження. Ще однією перевагою лічильника є наявність двох каналів, тобто за потреби можна контролювати одразу два певні прилади.

Основними компонентами розроблюваного пристрою стали плата Arduino Uno на основі мікроконтролера ATmega328P та мікроконтролер ESP8266, що оснащений Wi-Fi-модулем.

Результатом роботи є принципова електрична схема лічильника електроенергії, розроблений відповідний схемі прилад та написаний програмний код для обробки даних спожитої електроенергії.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Мікроконтролер ESP8266

ESP8266 – це мікроконтролер, який являє собою мережеве рішення з Wi-Fi-модулем на борту та з можливістю виконання додатків, що записуються в його пам'ять. Він має вбудований стек протоколів TCP/IP, що дозволяє йому зв'язуватися з мережами Wi-Fi. Мікроконтролер може бути програмований за допомогою середовища Arduino IDE або інших середовищ програмування.

Існує безліч модифікацій плат, які називаються зазвичай від ESP-01 до ESP-12. Відмінність в платах полягає в основному в портах введення-виведення, кількості флеш-пам'яті, виду конекторів і т.п. Процесор - той самий, отже з погляду програмування немає значення яку плату програмувати.

ESP8266 має вбудовану флеш-пам'ять, яка може бути використана для зберігання програмного коду та даних. Також мікроконтролер має GPIO-піни, які можна використовувати для з'єднання з датчиками, реле та іншими зовнішніми пристроями. [1]

Крім цього, ESP8266 підтримує різні протоколи комунікації, такі як HTTP, MQTT, WebSocket та інші, що дозволяє легко інтегрувати його з іншими системами та пристроями.

Специфікація ESP8266:

- Напруга живлення: 3,3В
- Енергоспоживання: 10мкА...170мА
- Флеш-пам'ять: до 16Мб максимум (зазвичай 512Кб)
- Процесор: Tensilica L106

- Швидкість процесора: 80...160 МГц
- ОЗП: 32Кб + 80Кб
- Порти введення /виведення загального призначення: 17
- АЦП: 1 введення з роздільною здатністю 1024
- Підтримка 802.11: b/g/n/d/e/i/k/r
- Максимальна кількість підключень TCP: 5

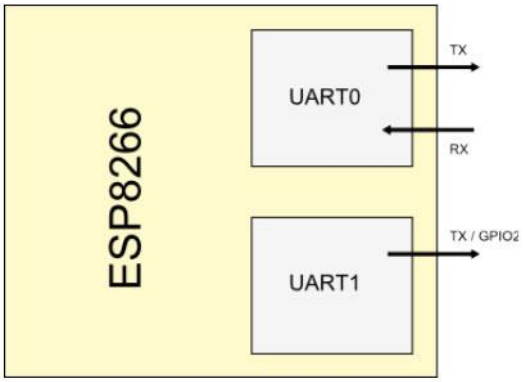


Рис. 1.1.1 – порт UART

Щоб спростити процес програмування модуля, можна скористатися послідовним портом UART (див. рис. 1.1.1), який дозволяє передавати дані між мікроконтролером та комп'ютером через серійний інтерфейс. ESP8266 має спеціальні два порти для цього, зазначені RX і TX. TX – служить для передачі даних, а RX – прийому. Найзручніше підключити цей порт до комп'ютера за допомогою перехідника USB-UART. З таким підключенням ми можемо надсилати команди до модуля прямо з програми терміналу, використовуючи клавіатуру комп'ютера, та отримувати відповіді від модуля до терміналу або записувати програму до модуля. [2]

Модуль ESP8266 має також другий послідовний порт. Головне його призначення – виведення діагностичної та налагоджувальної інформації. Вивід TX другого послідовного порту мультиплексовано з виводом GPIO2.

Найшвидший спосіб взаємодії з модулем ESP8266 полягає у передачі йому AT-команд і отриманні відповідей. AT-команди – спеціальний набір

інструкцій, які модуль розуміє і виконує певні дії під час їх отримання. Результати виконання команд потім виводяться в термінал для перегляду.

1.2 Плата Arduino Uno

Arduino Uno є однією з найбільш популярних плат мікроконтролера, яку можна використовувати для розробки електронних пристроїв та проектів. Вона має компактний розмір і забезпечує зручну розробку програмного забезпечення за допомогою мови програмування C++.

Плата Arduino Uno базується на мікроконтролері ATmega328P від компанії Microchip. Вона містить 14 цифрових входів/виходів, 6 аналогових входів, кварцовий резонатор на 16 МГц, USB-порт, роз'єм живлення, та інші компоненти, необхідні для розробки різноманітних проектів. На багатьох платах доступний вбудований лінійний стабілізатор напруги, який забезпечує стабільний вихідний струм з напругою +5В або +3,3В. Крім того, в мікроконтролері встановлений завантажувач (bootloader), що дозволяє завантажувати програму без потреби в зовнішньому програматорі. [3]

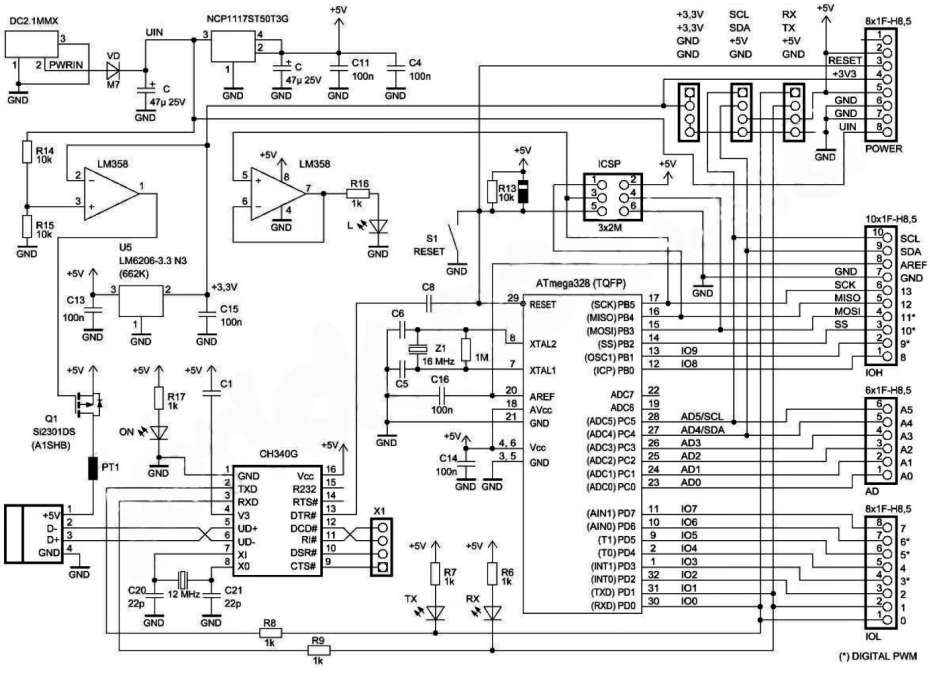


Рис. 1.2.1 – принципова схема плати Arduino Uno

З рисунка 1.2.1 можна побачити, що плата Arduino Uno складається з порта USB – для прошивки та живлення плати, кнопки скидання, що призначена для ручного скидання прошивки, роз’єму живлення DC – для підключення зовнішнього джерела від 7 до 12В, мікроконтролера ATmega328P, ICSP-роз’єму ATmega328P, світлодіодної індикації та двох знижувальних регуляторів 5В та 3.3В.

Світлодіодна індикація складається з таких світлодіодів:

- ON – індикатор живлення платформи
- L – користувальницький світлодіод на 13 піні мікроконтролера, який при заданні значення високого рівня вмикається, за низького – вимикається
- RX/TX – блимають під час прошивки та обміну даними між платою та комп'ютером, а також при використанні пінів 0 та 1

Знижувальний лінійний перетворювач 5В забезпечує живлення мікроконтролера та іншої логіки плати через роз’єм живлення DC. Діапазон вхідної напруги від 7 до 12В. Вихідна напруга 5В з максимальним вихідним струмом 1А.

Перетворювач 3.3В забезпечує напругу на піні 3V3. Приймає вхідну напругу від лінії 5В і видає напругу 3.3В з максимальним струмом 150mA.

Додаткові характеристики плати Arduino Uno:

- Робоча напруга – 5В
- Напруга живлення (рекомендована) – 7-12В
- Напруга живлення (критична) – 6-20В
- Flash-пам’ять – 32КБ (ATmega328P), з яких 0.5КБ використовуються загружчиком

- Максимальний струм одного виводу – 40мА
- Максимальний вхідний струм виводу 3.3В – 50мА
- Тактова частота – 16МГц

1.3 Мікроконтролер ATmega328P

Мікроконтролер Atmel ATmega328P – це 8-бітний megaAVR пристрій високої якості та відмінної надійності, який заснований на архітектурі RISC (покращеної AVR). Його особливістю є технологія picoPower, яка забезпечує наднизьке енергоспоживання та режими сну з низьким енергоспоживанням.

Контролер має три види пам'яті:

- 32КБ флеш-пам'яті, з яких 0,5 КБ використовується завантажувачем, що дозволяє прошити Uno зі звичайного комп'ютера через USB. Флеш-пам'ять є постійною і призначена для зберігання програм і відповідних статичних ресурсів.
- 2Кб оперативної пам’яті для зберігання тимчасових даних, наприклад програмних змінних. По суті, це оперативна пам'ять платформи. Оперативна пам'ять є енергозалежною, і при її вимкненні всі дані будуть стерті.
- 1КБ енергонезалежного EEPROM для тривалого зберігання даних, які не стираються, коли контролер вимкнено. За своїм призначенням це аналог жорсткого диска для Uno

Для прошивання даного мікроконтролера використовується ICSP-роз’єм, що зображений на рисунку 1.3.1.

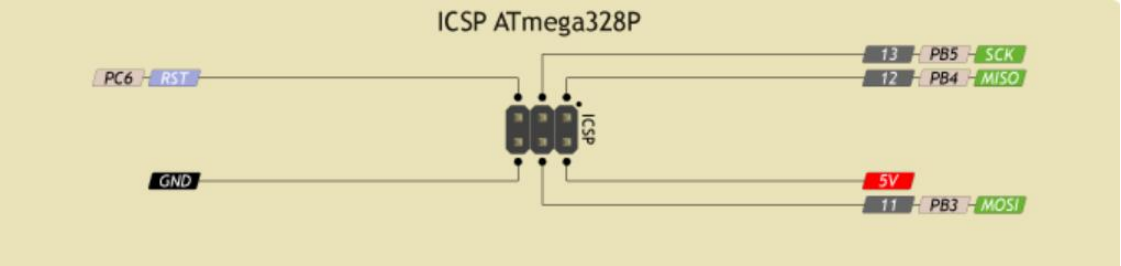


Рис. 1.3.1 – ICSP-роз'єм

Даний роз'єм мікроконтролера ATmega328P виконує дві корисні функції:

- Використовується для передачі сигнальних пінів інтерфейсу SPI при підключенні Arduino Shield або інших розширень. Лінії ICSP-роз'єму також продубльовані на цифрових пінах MOSI/11, MISO/12 та SCK/13. [4]
- Призначений для завантаження прошивки у мікроконтролер ATmega328P через зовнішній програматор. Одна з таких прошивок – Bootloader для Arduino Uno, яка дозволяє прошивати платформу по USB.

Що стосується його параметрів та найбільш видатних технічних характеристик:

- 23 лінії введення-виведення загального призначення
- 32 регістри загального призначення
- 3 таймери/лічильники з режимом порівняння
- Внутрішні/зовнішні переривання (24)
- Послідовний інтерфейс
- 8-канальний 10-бітовий АЦП
- 5 програмних режимів енергозбереження
- Живлення від 1.8 до 5.5 ст.

- Тактова частота 20МГц

1.4 Протокол UART

UART – універсальний асинхронний приймач, який визначає протокол обміну послідовними даними між двома пристроями.

UART – дуже простий протокол, в якому використовується лише два дроти між передавачем та приймачем для передачі та прийому даних в обох напрямках. Обидва кінці мають заземлення.

Зв'язок у UART може бути симплексним (дані передаються тільки в одному напрямку), напівдуплексним (кожна сторона здійснює передачу, але тільки по черзі), або повнодуплексним (обидві сторони можуть передавати одночасно).

Однією з найбільших переваг протоколу є його асинхронність – передавач і приймач не використовують загальний тактовий сигнал. Тому в даному випадку для роботи UART необхідно налаштувати однакову швидкість передачі або бітову швидкість на обох сторонах з'єднання.

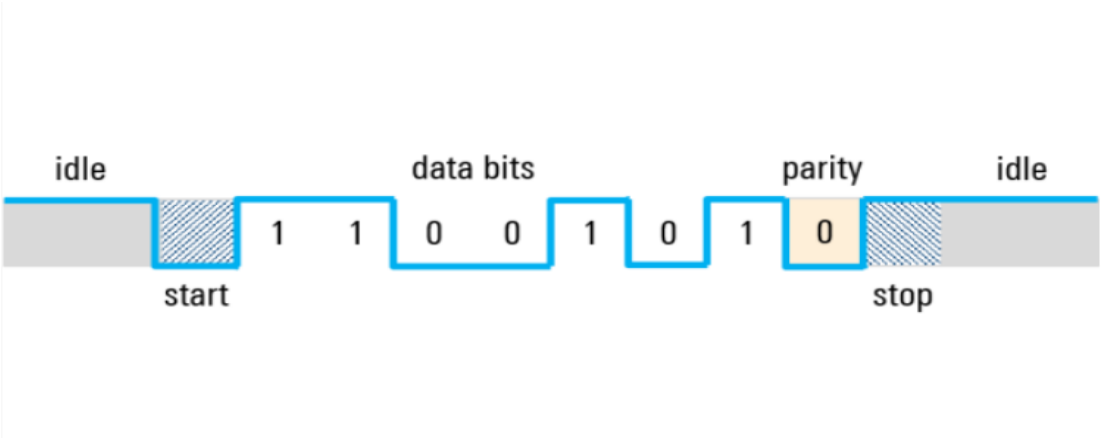


Рис. 1.4.1 – передача даних по протоколу UART

Дані UART передаються у вигляді кадрів. Кадри протоколу містять стартові і стопові біти та біти даних (див. рис. 1.4.1). Оскільки UART є асинхронним протоколом, передавач повинен сигналізувати надходження бітів даних. Це робиться за допомогою стартового біта. Стартовий біт – це перехід зі стану очікування високого рівня в стан низького рівня, за яким

відразу ж слідують біти даних користувача. [5]

Після того, як біти даних закінчилися, стоповий біт вказує на закінчення даних користувача. Стоповий біт – це перехід назад у стан високого рівня або стан очікування, або збереження цього стану протягом додаткового бітового інтервалу.

Біти даних є даними користувача або «корисними» бітами і йдуть відразу після стартового біта. Може бути від 5 до 9 бітів даних, хоча найчастіше використовується 7 або 8 бітів. Ці біти даних зазвичай передаються у форматі з першим молодшим бітом.

У багатьох пристроях, які підтримують UART (наприклад, Arduino та Arduino-сумісні плати), є два контакти, позначені як RX і TX. TX означає передачу (відправлення), а RX – отримання. Це означає, що потрібно підключити RX одного пристрою до TX іншого, так як це показано нижче на рисунку 1.4.2, щоб вони могли взаємодіяти. Якщо RX одного пристрою підключити до RX іншого, обидва пристрої будуть слухати один одного, оскільки їх входи будуть з'єднані. Підключення TX до TX є небезпечним, так як це вихідні контакти пристроїв низького опору, і якщо один має логічну одиницю, а інший – нуль, може статися коротке замикання. Хоча сучасні чіпи мають захист від цього, краще не розраховувати на нього. Також необхідно з'єднати референсні рівні двох пристроїв (GND-GND), якщо вони не мають гальванічного роз'єднання.

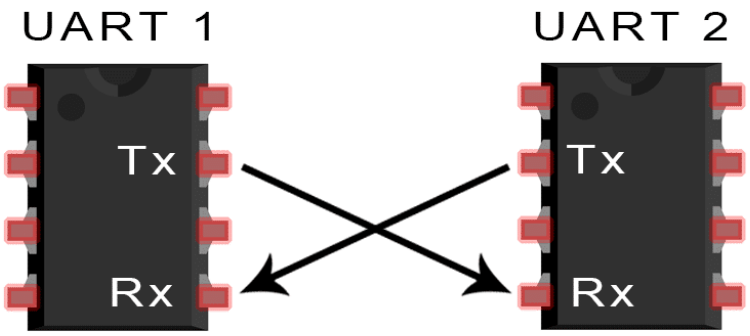


Рис. 1.4.2 – принцип з’єднання виводів RX та TX

1.5 Модуль датчика струму ACS712

Модуль ACS712 використовується для вимірювання постійного і змінного струму до 20А. Контроль та вимірювання струму є ключовим вимогами для широкого спектру застосувань, таких як захисні схеми від перевантаження по струму, зарядні пристрої для акумуляторів, імпульсні джерела живлення, програмовані джерела живлення та інші.

Мікросхема ACS712 використовує ефект Холла в своєму принципі роботи для вимірювання струму. Вона має вбудований чутливий елемент Холла, який реагує на магнітне поле, створене струмом, що протікає через провідник. Коли струм проходить через провідник на мікросхемі ACS712, виникає магнітне поле навколо провідника. Чутливий елемент Холла реагує на це магнітне поле і генерує вихідний сигнал, який пропорційний величині струму. Цей вихідний сигнал може бути зчитаний мікроконтролером, таким як ESP8266 або Arduino, для подальшого оброблення.

Ефект Холла у мікросхемі ACS712 дозволяє точно виміряти струм без необхідності втручання у провідник або вимірювальну схему. Це робить ACS712 зручним для використання в різних додатках, де необхідно здійснювати моніторинг струмових параметрів. Завдяки використанню ефекту Холла, мікросхема ACS712 може надати точне і надійне вимірювання струму, що зробило її популярною в багатьох проектах, таких як контроль споживання електроенергії, системи безпеки, керування двигунами та багато інших.

ACS712 живиться від постійної напруги 5В і споживає дуже малий струм – 11мА. Модуль здатний витримувати великі перевантаження вимірюваного струму, а ще він простий в підключенні і надійний в роботі. Крім цього, мікросхема має вбудований конденсатор фільтра, що знижує його шумові характеристики та підвищує точність вимірювань.

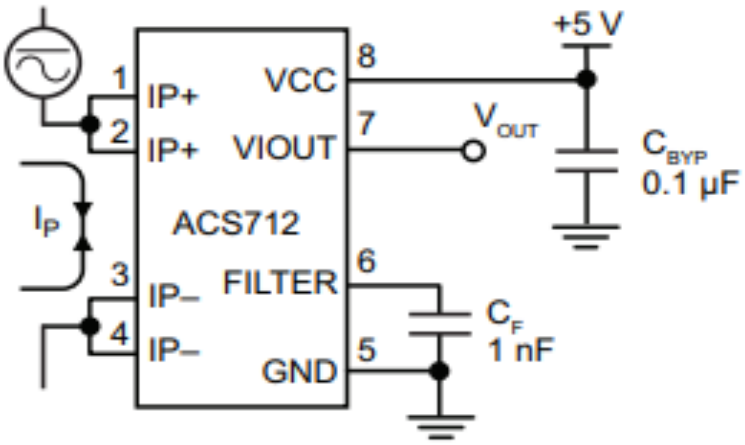


Рис. 1.5.1 – мікросхема ACS712

Для підключення мікросхеми ACS712 до плати Arduino використовуються порти (див. рис. 1.5.1):

- VCC – живлення (опорна напруга 5В)
- GND – земля
- OUT – сигнальний (підключається до аналогового виводу контролера Arduino)

При роботі мікросхеми ACS712 з ESP8266 або платою Arduino основна ідея полягає в тому, що ACS712 вимірює струм, а мікроконтролер або Arduino обробляють ці вимірювання та виконують додаткову логіку і керують зовнішніми пристроями або зберігають дані для подальшого використання.

Після встановлення правильних з'єднань між ними, ми можемо програмно працювати зі значеннями струму, які надає ACS712. Залежно від платформи (ESP8266 або Arduino), ми можемо використовувати вбудовані бібліотеки або створювати власний код для отримання аналогових значень з ACS712 та їх подальшого використання.

Додаткові характеристики мікросхеми:

- Чутливість: 100мВ/А
- Максимальна частота 50кГц
- Температурний діапазон: -40°С..+85°С
- Струм споживання не перевищує: 11мА
- Опір струмової шини 1.2мОм
- Напруга живлення: 5В
- Гальванічна розв'язка, пробивна напруга: 2.1кВ
- Розміри: 31 х 13мм

1.6 Ефект Холла

Ефект Холла був відкритий у 1879 році американським фізиком Е. Холлом (1855-1938) і полягає у виникненні в провіднику із струмом, який знаходиться у магнітному полі, електричного поля в напрямку, перпендикулярному до напряму струму і індукції магнітного поля.

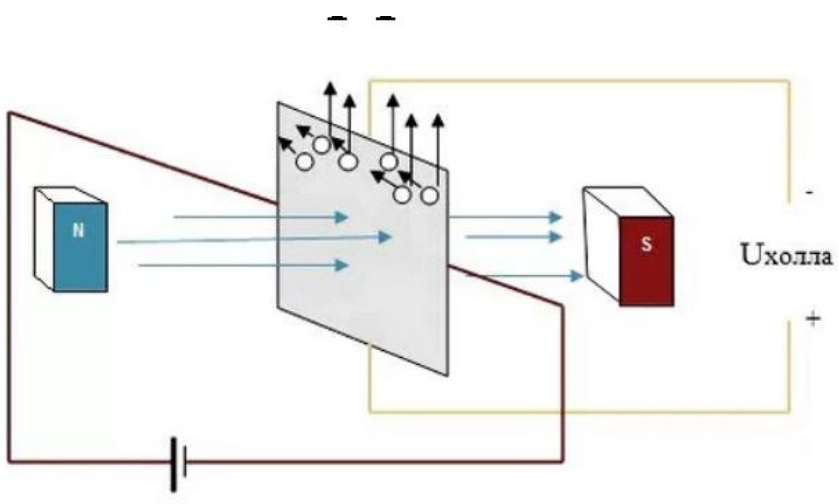


Рис. 1.6.1 – принцип роботи ефекту Холла

Причиною виникнення цього ефекту є відхилення руху заряджених частинок (електронів, дірок) під дією сили Лоренца. Внаслідок цього вільні

носії заряду відхиляються до однієї з бічних граней (див. рис. 1.6.1). Не дивлячись на те, що напрямок сили Лоренца, який діє і на електрони і на дірки однаковий (він визначається правилом лівої руки у відповідності із напрямком струму і індукції магнітного поля), до однієї з граней відхиляється більше тих носіїв, концентрація яких більша. [6]

Ефект Холла є важливим відкриттям, яке знайшло широке застосування в різних галузях промисловості. Цей ефект використовується у багатьох сучасних пристроях, які є невід'ємною частиною нашого повсякденного життя. Основою ефекту Холла є поява різниці потенціалів у поперечному напрямку провідника, який проводить струм у сильному магнітному полі (див. рис. 1.6.2). Це відкриття стало основою для розвитку безлічі пристроїв і систем, які потрібні для нашого сучасного життя.

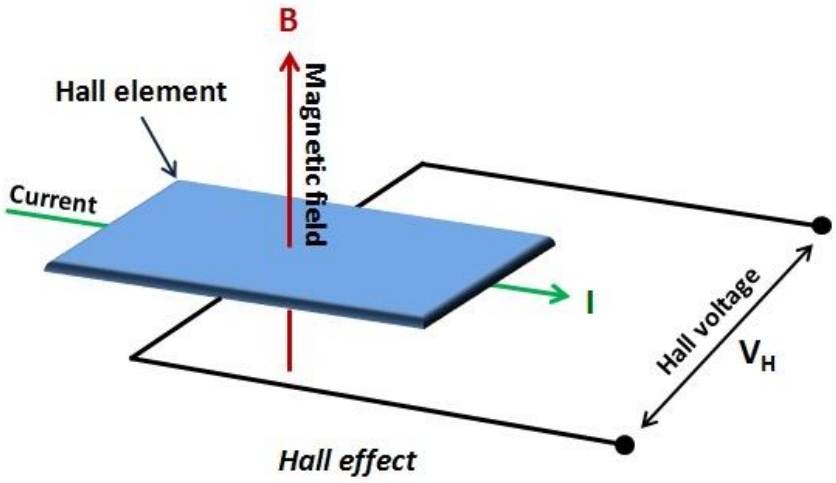


Рис. 1.6.2 – магнітне поле в ефекті Холла

Ефект Холла є чудовим прикладом того, як наука та технологія співпрацюють для створення інноваційних пристроїв. Його застосування розповсюджене у багатьох галузях, включаючи електроніку, енергетику, автомобільну промисловість та медицину. Головною перевагою ефекту Холла є його здатність точно виміряти магнітне поле і струм, що дозволяє розробляти високоточні пристрої та системи керування.

2. ПРАКТИЧНА ЧАСТИНА

2.1 Будова лічильника

Лічильник побудований на основі двох контролерів: перший з них – ESP8266, а другий – ATmega328P. Контролер ATmega є основою плати Arduino і власне він виконує функції вимірювання електроенергії, обробки даних та передачі їх на контролер ESP8266.

Також хочу додати, хоч контролер ESP8266 має модуль Wi-Fi і загалом здатен виконати всю роботу, проте в нього не дуже вдалий АЦП, який з Wi-Fi працює гірше, на відміну від АЦП контролера ATmega. Тому для цифрування напруги та обробки даних використовується додатковий контролер.

Дані з ATmega на ESP8266 передаються по протоколу UART (протокол com-порта).

Комунікація мікроконтролерів здійснена через роз’єм J3, що зображений на рисунку 2.1.1, за допомогою якого контролер ATmega по протоколу UART передає дані на контролер ESP8266. До нього входять дві лінії RXD та TXD, а також земля. З першого погляду може здатись, що дані лінії нікуди не підключені, проте всі провідники прямують до мікроконтролера ATmega. RXD та TXD – це відповідно вхід і вихід апаратного модуля UART.

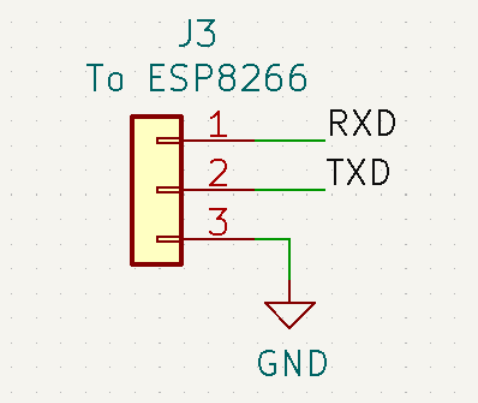


Рис. 2.1.1 – роз'єм J3

Для того, щоб розрахувати середню потужність споживання певного приладу, нам спочатку потрібно знайти миттєве значення потужності – як добуток миттєвих значень напруги та сили струму. А потім, проінтегрувавши миттєву потужність, знайдемо її середнє значення.

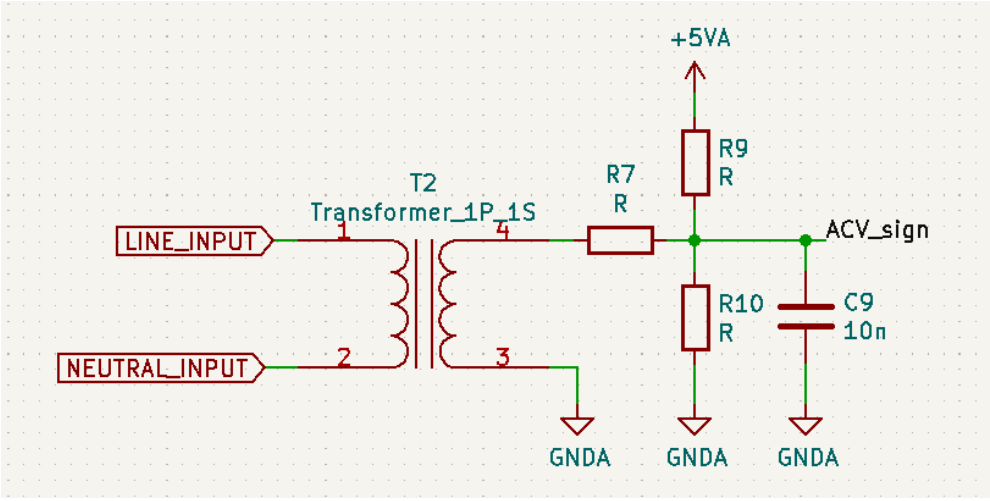


Рис. 2.1.2 – трансформатор напруги

Для вимірювання миттєвого значення напруги використовується трансформатор напруги (див. рис. 2.1.2), який підключається до двох провідників за допомогою lineinput (вивід фази) та neutralinput (вивід нуля). З виходу трансформатора напруга передається на подільник (R7, R9, R10), який окрім узгодження значень вимірюваної напруги з динамічним діапазоном АЦП, додає до змінної напруги постійну складову. Це зроблено, тому, що АЦП контролера міряє однополярну напругу лише від 0 до 5В. Тоді

на виході нашого трансформатора напруга буде коливатися від 0 до 5В. Окрім того, важливим наслідком використання трансформатора є гальванічна розв’язка між мережею і вимірювальною схемою.

Конденсатор С9 разом з опорами R утворює фільтр низьких частот, який використовується для фільтрування завад та комутаційних піків, щоб вони не потрапили на АЦП ATmega.

Виходом ACV_sign з’єднуємось з контролером по 25 виводу, знову ж таки не використовуючи провідник на схемі.

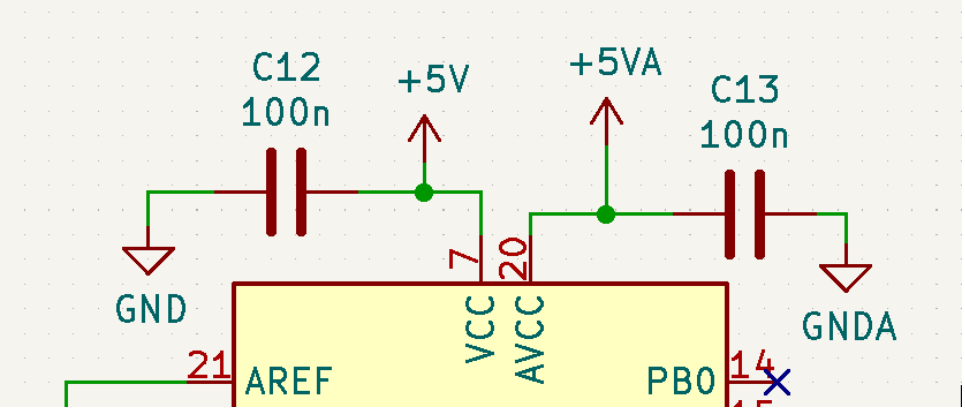


Рис. 2.1.3 – блокуючі конденсатори C12 та C13

Для живлення АЦП використовується окремий вхід напруги AVCC (аналогова напруга) для того, щоб пульсації, які присутні на напрузі не впливали на точність вимірювання.

Конденсатори С12 та С13, що зображені на рисунку 2.1.3, так звані блокуючі конденсатори, котрі якраз і блокують всі пульсації напруги, які могли б перейти на АЦП.

2.2 Вимірювання струму

Для вимірювання струму використовуємо мікросхему ACS712 (див. рис. 2.2.1), яка міряє струм на основі ефекту Холла. Також можна піти іншим шляхом та виміряти струм шунтом, проте дана мікросхема має перевагу в тому, що тут немає гальванічного зв’язку, а відповідно прямого контакту з

мережею. Тоді вимірювальна схема прямо до 0 або фази не під'єднується, на відміну від випадку з шунтом, який потрібно було би підключити до 0, і з нього ж отримувати струм, що не надто вдало для такого лічильника. Для даної схеми точніший результат зможе надати саме мікросхема на основі ефекту Холла. [7]

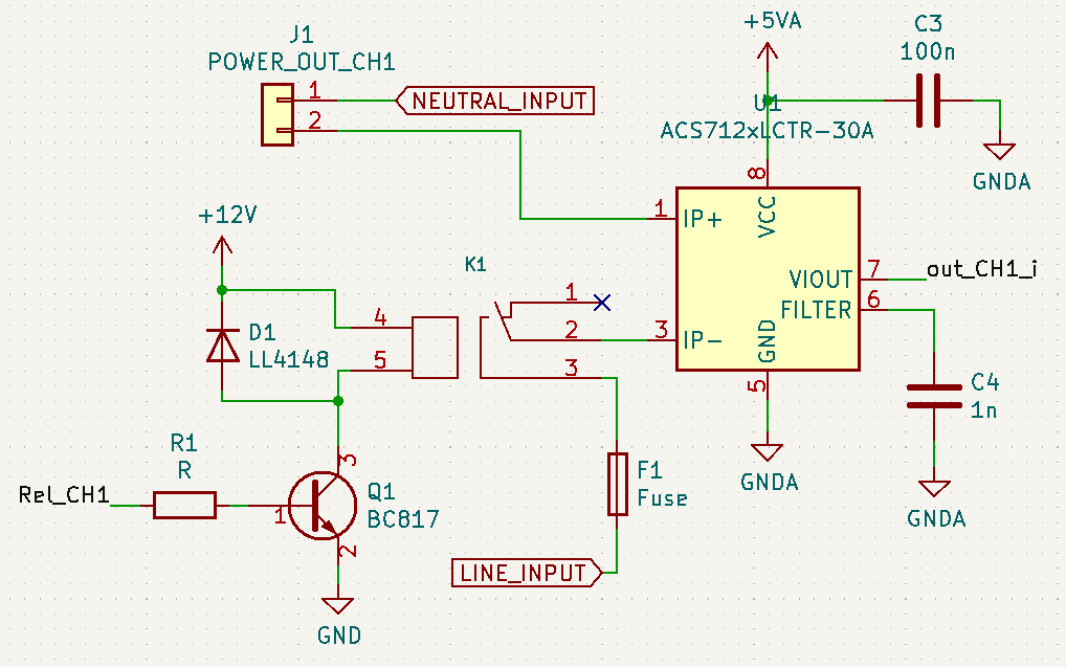


Рис. 2.2.1 – схема підключення IMC ACS712

Можна використати і трансформатор струму (пояс Роговського), який також дає гальванічну розв'язку та вимірює струм фази, проте, якщо навантаження має значну нелінійність (якщо, наприклад, послідовно з навантаженням увімкнутий діод, так, що струм проходить тільки під час однієї півхвилі синусоїди), то виникне постійна складова струму, яку трансформатор не зможе виміряти, на відміну від IMC ACS712.

Спираючись на наведені вище факти, можемо дійти висновку, що даний прилад можна використовувати для контролю навантаження, який також зможе відслідковувати постійну складову струму.

На дану мікросхему подається 5В живлення. Наявний блокуючий конденсатор С13, який поставлений паралельно до мікросхеми для уникнення пульсацій навантаження. Інший конденсатор С4 потрібен по

datasheet для роботи самої мікросхеми. Виходи out_CH1_i та out_CH2_i міряють вихід напруги, який пропорційний до струму і також заведені на АЦП.

POWER_OUT_CH1 – роз'єм, до якого ми підключаємо навантаження. До його першого виводу підключений нуль, а до другого – мікросхема вимірювання струму.

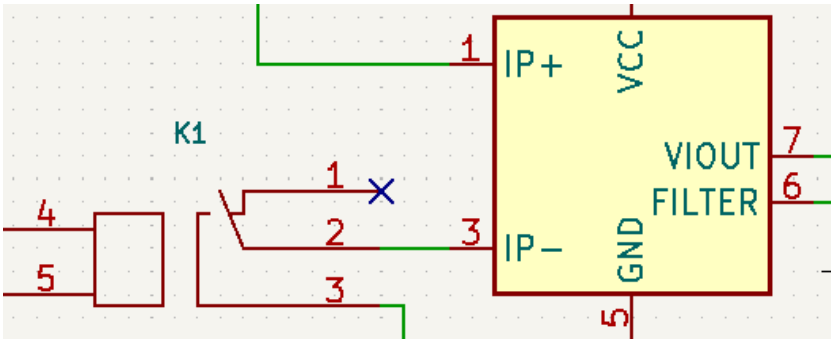


Рис. 2.2.2 – схема підключення комутуючого реле до ACS712

Перший та третій виводи мікросхеми, утворюють первинне коло, через них проходить струм мережі, який ми і міряємо. Рухаючись далі по третьому виводу можемо побачити контакти комутаційного реле (див. рис. 2.2.2), та його обмотку, яка рухає пластиною, і при ввімкненні/вимкненні струму обмотки, відповідно замикає/розмикає контакти. Контакт 2 – нерухомий, 1,3 – рухомі. Також контакти 1 та 2 являються нормально замкнутими, тобто, коли напруга на обмотку не подана – 1,2 замкнуті, а коли подана – контакт 2 розмикається.

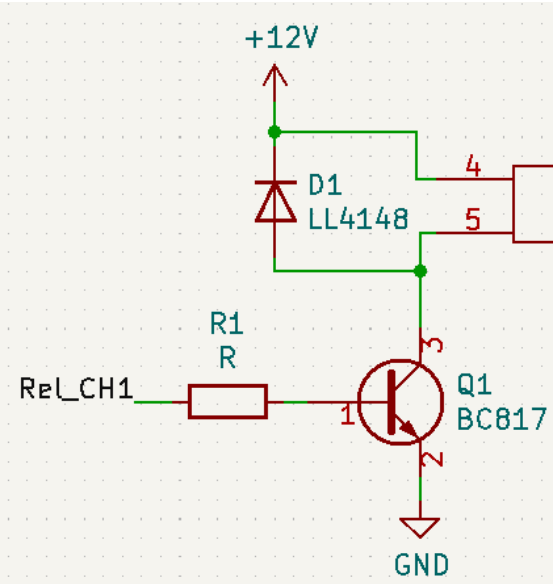


Рис. 2.2.3 – схема керування обмоткою реле

На схемі також наявний транзистор Q1 (див. рис. 2.2.3), який підсилює струм реле. Так як контролер напряму не може керувати обмоткою, адже її напруга 12В, а напруга контролера – 5В, то це відбувається через транзистор. Через вихід Rel_CH1, який під'єднаний до АЦП, струм потрапляє через обмежувальний резистор на транзистор, замикає або розмикає його, а останній в свою чергу пропускає або не пропускає струм на обмотку.

D1 – антипаралельний діод, який пропускає струм обмотки реле, коли транзистор закривається. Оскільки, обмотка реле є котушкою, яка може мати велику індуктивність, то для неї справедливі закони комутації. Це означає, що струм котушки не можна обривати, щоб уникнути зростання напруги самоіндукції, яка може пробити транзистор. Щоб цього не сталося, використовуємо діод і струм котушки після закриття транзистора буде плавно згасати на діоді.

2.3 Вимірювання напруги

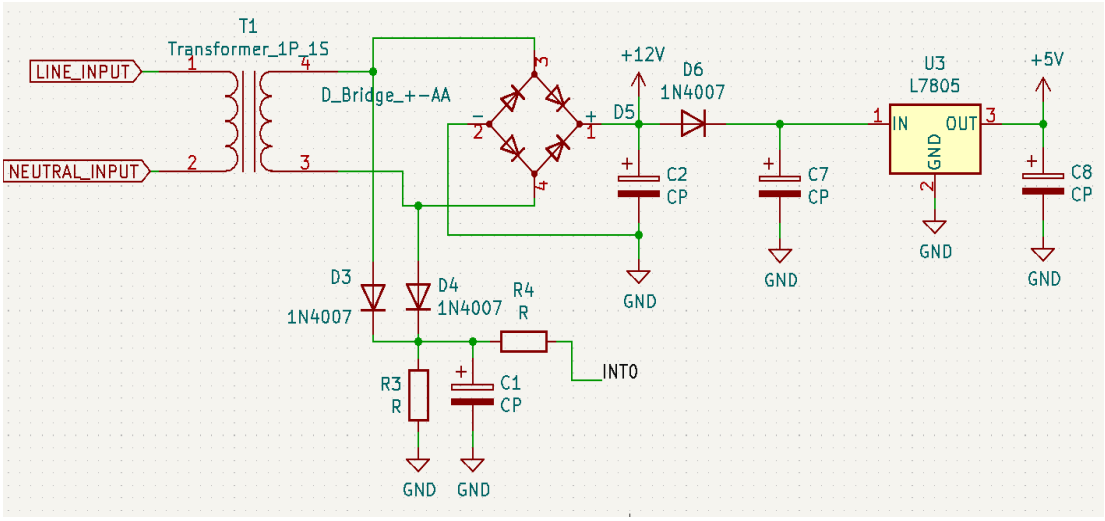


Рис. 2.3.1 – схема джерела живлення лічильника

На рисунку 2.3.1 показано схему джерела живлення лічильника, який складається з трансформатора T1, діодного моста, який зі змінної напруги робить постійну, фільтруючих конденсаторів, діода розгалуження ліній живлення D6, та лінійного стабілізатора напруги U3. Останній, стабілізує пульсуючу напругу з виходу діодного моста, утворюючи стабільну напругу 5В з малим коефіцієнтом пульсацій, яка використовується для живлення аналогової та цифрової частини схеми. По обидва боки від лінійного компенсаційного стабілізатора (мікросхема L7805) знаходяться конденсатори C7 та C8 для стабільності напруги.

Дані про спожиту електроенергію мають бути збережені контролером до енергонезалежної пам’яті, яка має скінченний ресурс перезапису. Тому, щоб уникнути швидкого зношення цієї пам’яті і в той же час не втратити дані при відключенні електроенергії використовується спеціальне коло, побудоване на діодах D3 та D4, резисторах R3 та R4 і конденсаторі C1. У момент зникнення живлення конденсатор C1 досить швидко розрядиться і сформує перепад рівня на вході INT0 мікроконтролера, який викличе переривання. У цьому перериванні і прописаний код, що запише необхідні дані до енергонезалежної пам’яті – EEPROM. Живлення схеми у цей час отримується від конденсаторів фільтрів живлення (C2, C7 та C8).

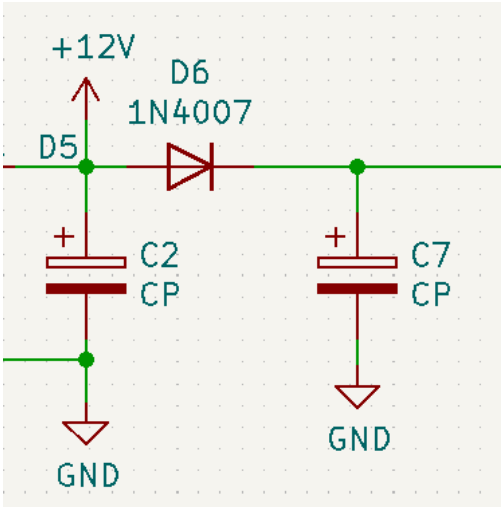


Рис.2.3.2 – діод D6 створює розгалуження шини живлення

Додаємо діод D6 та саме два конденсатори C2 та C7 (див. рис. 2.3.2), адже 12В-ова шина буде живити АЦП, а також може живити контакти реле. Таким чином, реле можуть розрядити конденсатор C2, проте за рахунок діода вони не розрядять конденсатор C7.

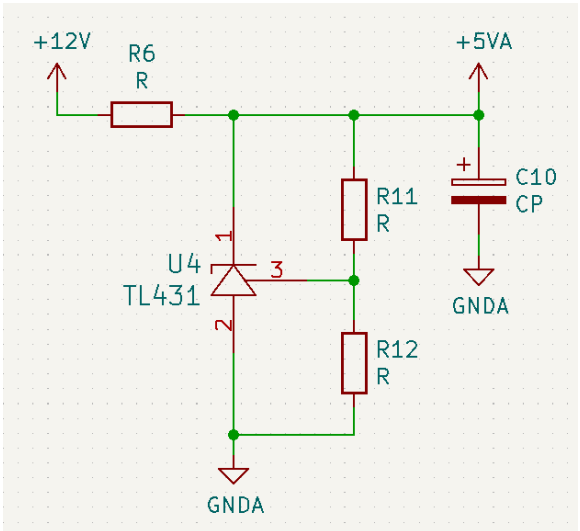


Рис.2.3.3 – схема джерела опорної напруги на основі TL431

На дану схему (див. рис. 2.3.3) подається нестабільна напруга 12В. Мікросхема TL431 – це прецизійний інтегральний стабілітрон, напруга якого слабко залежить від температури, тобто він досить точний, забезпечує стабільну напругу 5В з низьким рівнем пульсації.

Конденсатор C10 потрібен для фільтрування напруги. Необхідна дана схема для живлення аналогової частини АЦП та зокрема усіх інших мікросхем. Якщо раптом напруга стає більшою 5В, стабілітрон відкривається та обмежує напругу на виході.

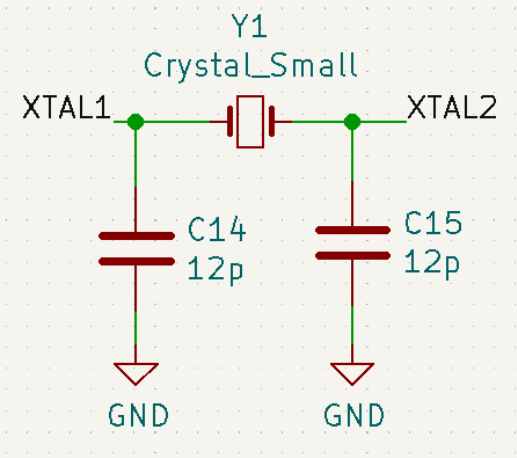


Рис.2.3.4 – зовнішній кварц

Для тактування ядра використовується зовнішній кварц (див. рис. 2.3.4), адже при обрахунку енергії потрібно інтегрувати по часу, тобто час має бути точним та стабільним.

Теж саме відноситься і до протоколу UART, який вимагає точні значення часу для коректної роботи.

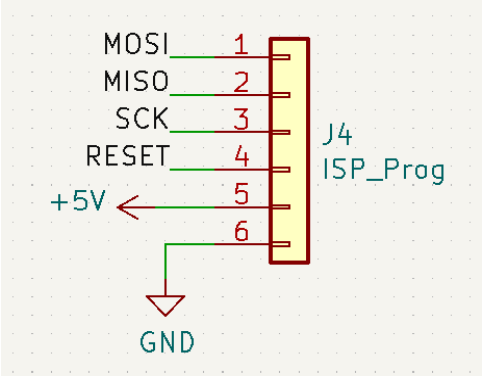


Рис.2.3.5 – роз'єм J4

Програмування контролера здійснюється за допомогою роз'єму J4, що

зображений на рисунку 2.3.5, за допомогою якого підключається програматор USBasp або його аналог.

Загальний вигляд принципової електричної схеми лічильника наведено в додатку А.

2.4 З’єднання плат Arduino та ESP8266

Для вдалого з’єднання плати Arduino Uno з мікроконтролером ESP8266 необхідно використати двоканальний перетворювач логічних рівнів.

Даний перетворювач рівнів розроблений для забезпечення підключення пристроїв, які мають різні значення логічних рівнів “0” та “1” та працюють з напругою 3.3В або 1.8В. Перетворювач може забезпечувати двонаправлене перетворення рівнів інтерфейсів SPI, UART, I2C (TWI), а також будь-яких цифрових сигналів. Важливо також пам'ятати, що він не працює з аналоговими сигналами. У даному випадку, перетворювач узгоджує логічні рівні мікроконтролерів, які мають різну напругу живлення.

Для використання перетворювача насамперед необхідно припаяти 6-пінові контактні роз'єми, так як зображено на рисунку 2.4.1.

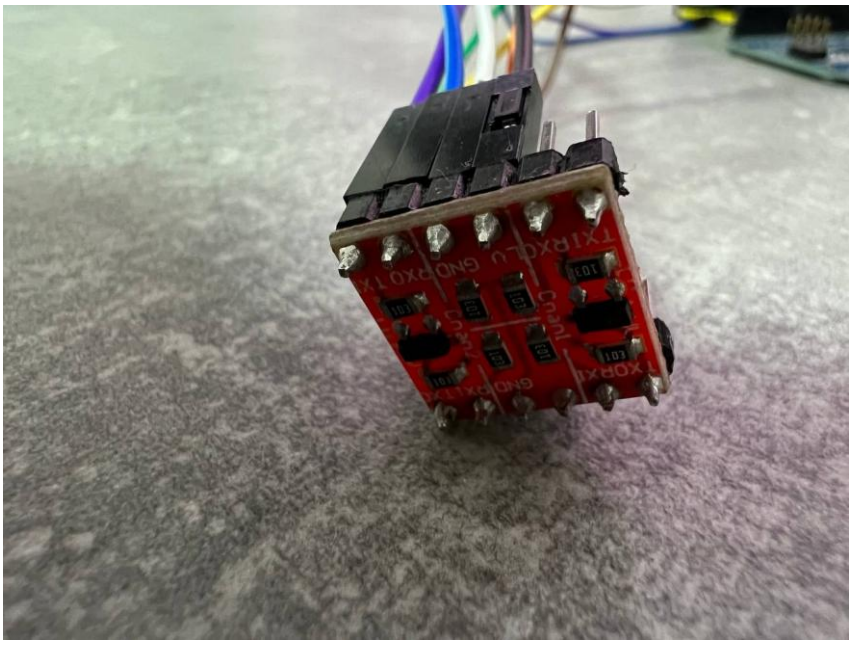


Рис.2.4.1 – зовнішній вигляд перетворювача логічних рівнів

Для з’єднання обох плат потрібно підключити високу напругу (наприклад, 5В) до виводу "HV", низьку напругу (наприклад, 2.8В) – до виводу "LV", а також з'єднати загальну точку схеми з виводом "GND" (див. рис. 2.4.2). Інші виводи модуля позначені як входи високої напруги (HV) або низької (LV). Логічна одиниця, яка надходить на вивід HV (на стороні 5В), з'явиться на виводі LV (з боку 3.3В) у вигляді напруги 3.3В. Аналогічно, логічна одиниця, яка надходить на вивід LV (на стороні 3.3В), з'явиться на виводі HV (на стороні 5В) у вигляді напруги 5В.

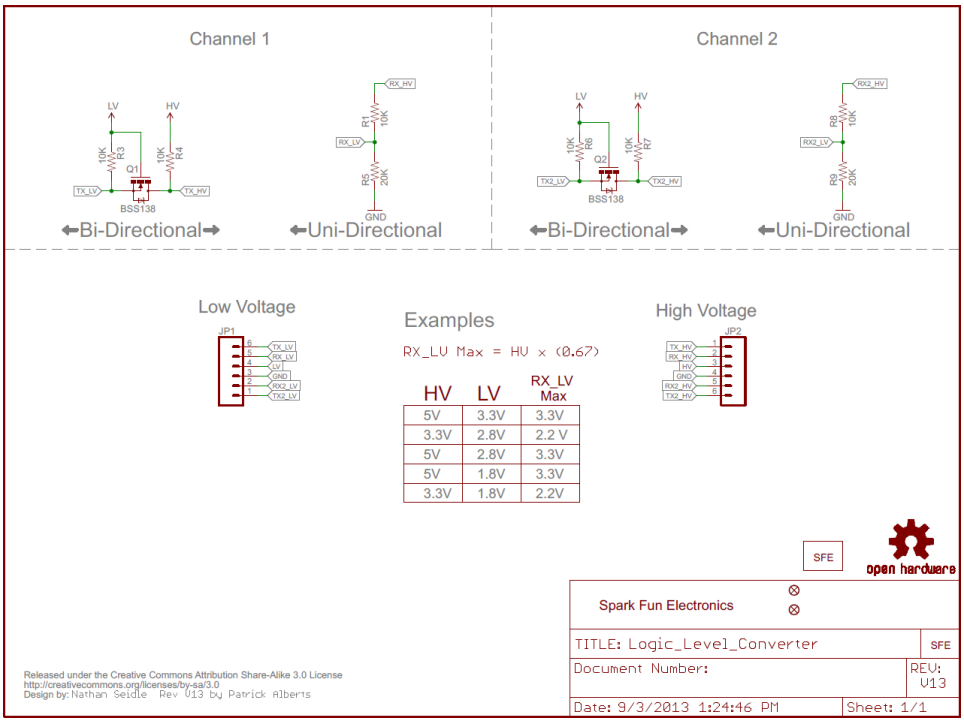


Рис.2.4.2 – принципова схема перетворювача рівнів

В основі пристрою лежать два транзистора MOSFET, завдяки чому перетворювач отримав двосторонню схему узгодження логічних рівнів на двох каналах і може як підвищувати, так і знижувати сигнали.

Для коректної роботи лічильника електричної енергії необхідно з’єднати виходи перетворювача з виходами плати Arduino та мікроконтролера ESP8266 (див. рис. 2.4.3).

Спочатку з’єднуємо два виводи землі живлення (GND) перетворювача відповідно з виходами CND плати Arduino та мікроконтролера. До виходу LV (низька напруга) підключаємо вивід 3.3В, що виходять з контролера ESP8266. Аналогічно для виводу HV (висока напруга) підключаємо вивід 5В з Arduino UNO. Необхідно це для того, щоб на мікроконтролер не подавалася надто висока напруга, яка здатна вивести його з ладу. Перетворювач логічних рівнів вирішує дану проблему і перетворює напругу 5В в 3.3В для кращої роботи мікроконтролера.

Наступним кроком з’єднуємо виводи RX/TX перетворювача з відповідними виводами ESP8266 та Arduino UNO, що відповідатимуть за двосторонній обмін даними між двома мікроконтролерами. Потрібно підключити RX одного пристрою до TX іншого, або навпаки, щоб плати могли взаємодіяти між собою.

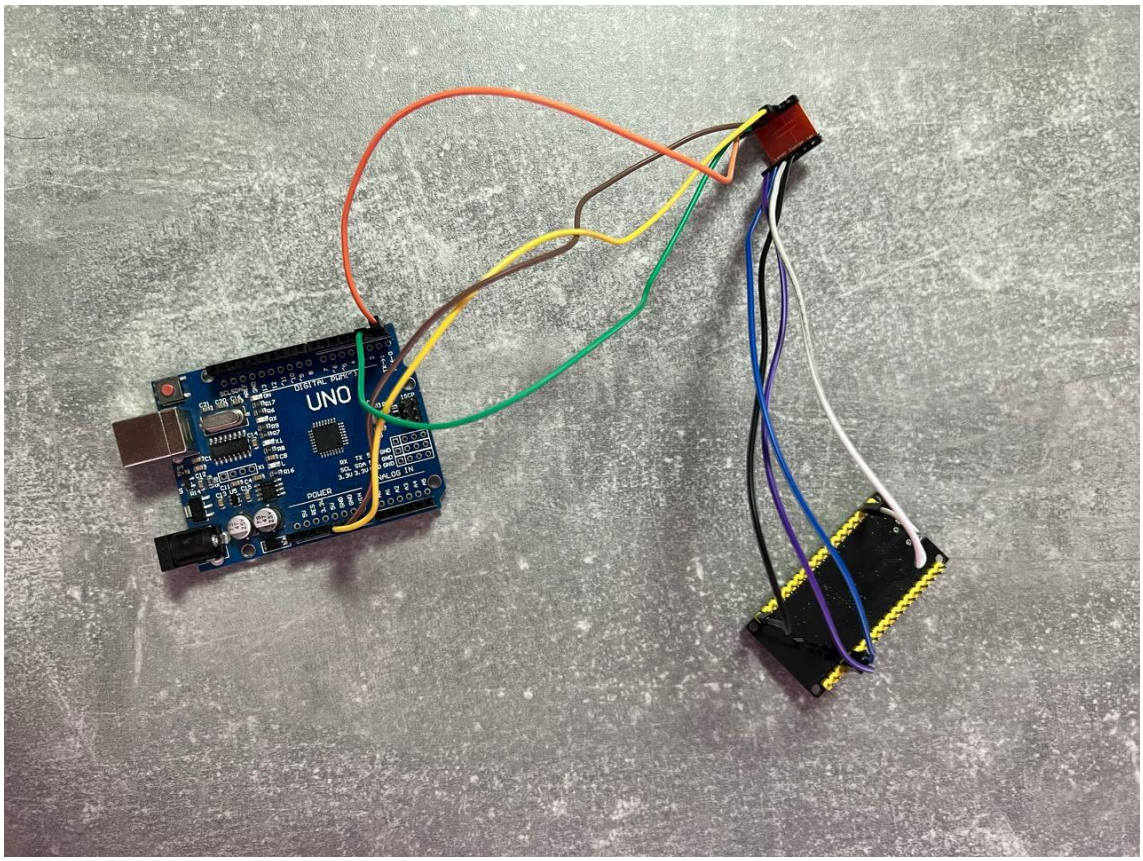


Рис.2.4.3 – з’єднання перетворювача з ESP8266 та Arduino Uno

2.5 Підключення до Arduino IDE

Arduino IDE – це програмне середовище розробки на основі мови програмування C++, що дозволяє писати програми за допомогою зручного текстового редактора, компілювати їх у машинний код і завантажувати на всі версії плати Arduino.

Для з'єднання пристрою з комп'ютером використовуємо кабель USB. При коректному підключенні на платі загоряється світлодіод "ON". Це означає, що на плату подано живлення. [8]

Для налаштування Arduino IDE під конкретну модель необхідно попередньо дізнатись, який номер COM-порту надав комп'ютер платі. Потрібну інформацію можна побачити, відкривши «Диспетчер пристроїв» Windows та вкладку «Порти (COM та LPT)» (див. рис. 2.5.1).

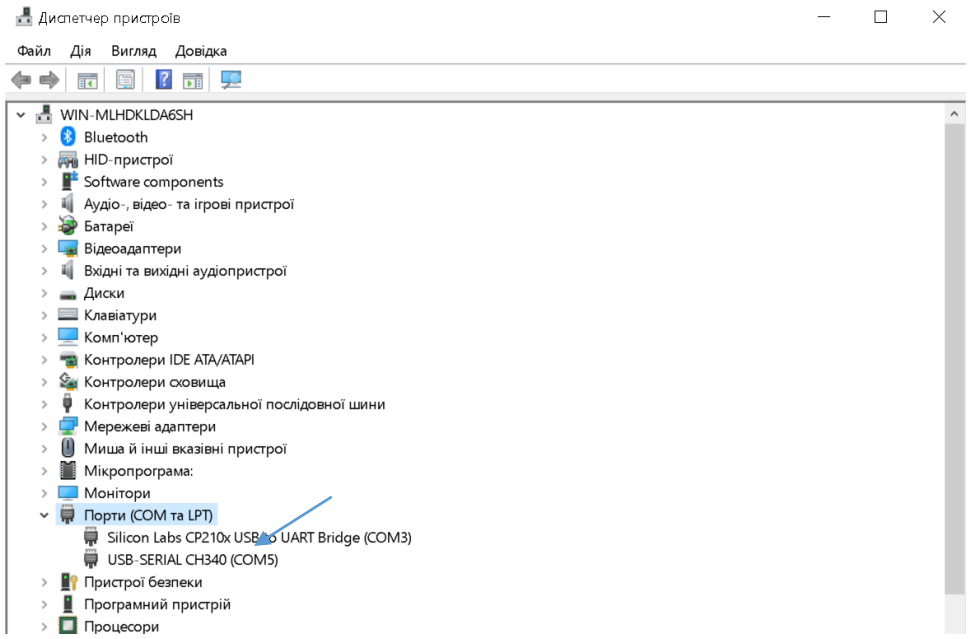


Рис.2.5.1 – пошук відповідного COM-порта

Для роботи середовища з конкретною платформою необхідно вибрати назву моделі та номер присвоєного платі COM-порту в Arduino IDE. Для вибору моделі заходимо у меню: Інструменти>Плата та вказуємо відповідну плату.

На рисунку 2.5.2 бачимо вибрану модель «Arduino Uno» та відповідний COM-порт – COM5.

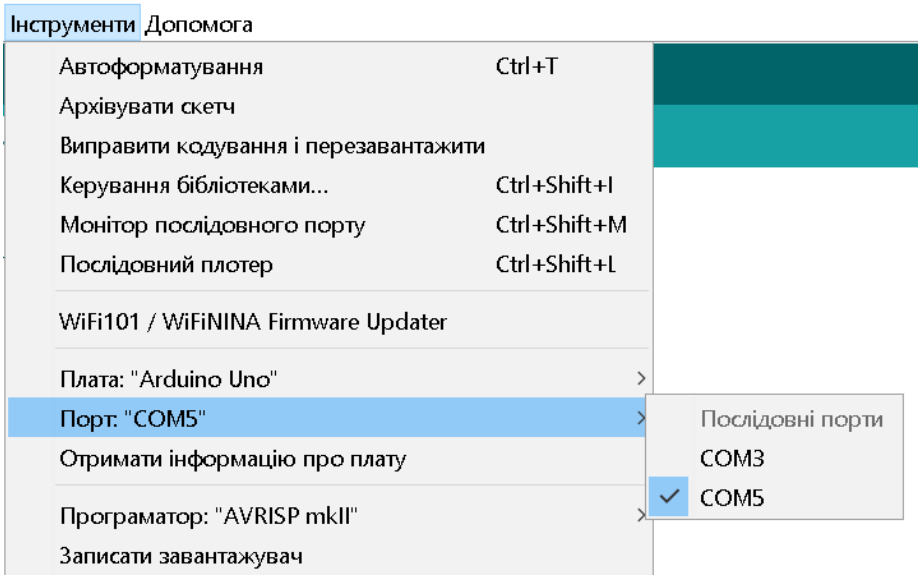


Рис.2.5.2 – панель інструментів Arduino IDE

Аналогічні дії повторюємо для підключення мікроконтролера ESP8266, але змінюємо обрану модель на «Generic ESP8266 Module» (див. рис. 2.5.3) та обираємо порт COM3.

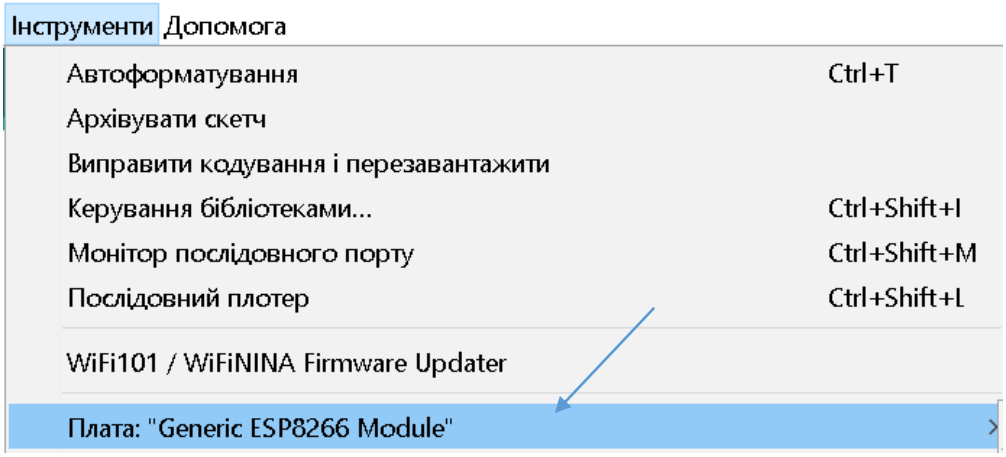


Рис.2.5.3 – підключення плати ESP8266 до Arduino IDE

2.6 Програмний код

Результатом роботи розробленого лічильника електричної енергії є HTML-сторінка. Вона відображає дані, отримані з плати Arduino Uno, щодо споживання електроенергії у веб-браузері з допомогою веб-сервера, яким, в

нашому випадку, виступає мікроконтролер ESP8266.

Веб-сервер – це місце, де веб-сторінки зберігаються, обробляються та видаються веб-клієнтом. Веб-клієнт – це програмне забезпечення, яке надає можливість користувачам отримувати доступ до веб-сайтів та взаємодіяти з ними на персональних комп'ютерах або мобільних пристроях. Зв'язок між клієнтом та сервером здійснюється за допомогою спеціального протоколу передачі гіпертексту (HTTP).

Одна з найважливіших функцій, яку забезпечує ESP8266, полягає в тому, що він може підключатися до існуючої Wi-Fi-мережі та працювати як веб-сервер. За допомогою функції WiFi.begin() підключаємось до існуючої бездротової мережі.

Поки ESP8266 намагається підключитись до мережі, ми перевіряємо стан підключення за допомогою функції WiFi.status(). Ця функція може повертати такі статуси: [9]

- WL_CONNECTED: повертається при підключенні до мережі Wi-Fi
- WL_NO_SHIELD: повертається, коли Wi-Fi модуль не підключено
- WL_IDLE_STATUS: тимчасовий стан, що повертається при виклику WiFi.begin(), і залишається активним доти, доки не закінчиться кількість спроб підключення (що призводить до WL_CONNECT_FAILED), або доки не буде встановлено з'єднання (що призводить до WL_CONNECTED)
- WL_NO_SSID_AVAIL: повертається, коли немає доступних SSID
- WL_SCAN_COMPLETED: повертається, коли сканування мереж завершено
- WL_CONNECT_FAILED: повертається при невдачі підключення після всіх спроб
- WL_CONNECTION_LOST: повертається у разі втрати з'єднання

- WL_DISCONNECTED: повертається при відключенні від мережі

Як тільки ESP8266 буде підключений до мережі, скетч друкує IP адресу, присвоєну ESP8266, відображаючи значення WiFi.localIP() в моніторі послідовного порту.

Функція SendHTML() відповідає за створення веб-сторінки при кожному отриманні запиту веб-сервером ESP8266 від веб-клієнта. Вона збирає HTML-код у один рядок і передає його у функцію server.send() для відправлення.

Перший текст, який завжди потрібно відправляти - це оголошення <!DOCTYPE>, яке вказує, що ми надсилаємо HTML-код. Потім <meta> елемент viewport робить веб-сторінку адаптивною у будь-якому веб-браузері. Далі з допомогою тегу title встановлюємо заголовок сторінки. Останнім кроком є стилізація веб-сторінки та встановлення заголовка на веб-сторінці.

Після завантаження скетчу відкриваємо монітор послідовного порту зі швидкістю 9600 біт/с та натискаємо кнопку RESET на ESP8266. Якщо все гаразд, він виводить динамічну IP-адресу, отриману від нашого маршрутизатора, і показує повідомлення, що HTTP сервер запущений, так як зображено на рисунку 2.6.1.

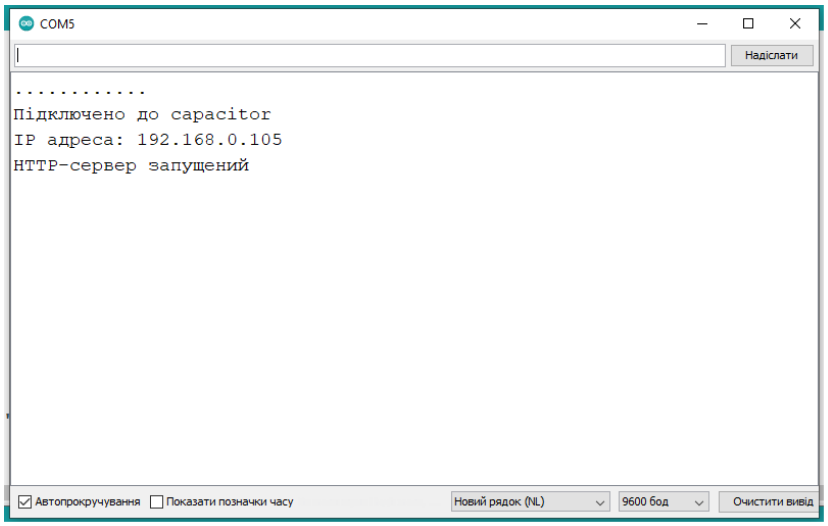


Рис.2.6.1 – монітор послідовного порту

Потім завантажуємо браузер і вводимо IP-адресу, вказану на моніторі послідовного порту. Отримуємо HTML-сторінку (див. рис. 2.6.2), що відображає розраховані дані напруги (voltage), струму (current of load), потужності навантаження (power of load) та загальної енергії навантаження (total energy of load) певного приладу.

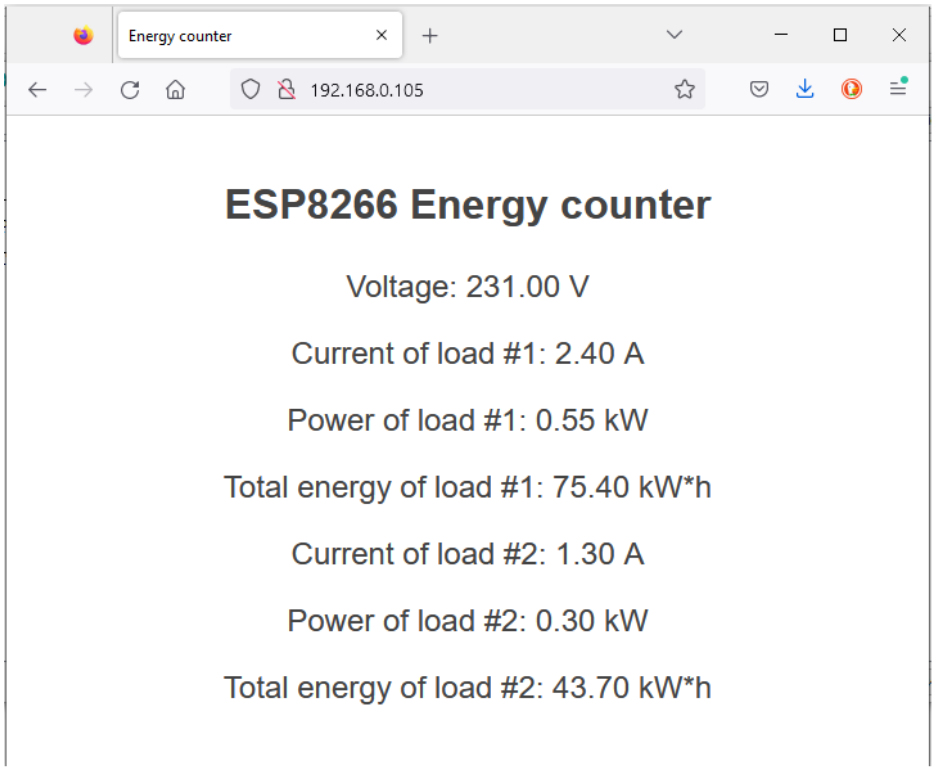


Рис.2.6.2 – зовнішній вигляд створеної HTML-сторінки

Відповідний програмний код, що використовується для створення сторінки у веб-браузері та відображення отриманих даних щодо споживання електроенергії, наведено в додатках Б та В.

ВИСНОВОК

В ході виконання випускної кваліфікаційної роботи бакалавра було спроектовано лічильник електричної енергії з Wi-Fi модулем.

Зокрема, розроблена принципова електрична схема вимірювальної та обчислювальної частини лічильника активної електричної енергії на основі мікроконтролера ATmega328 та розроблений відповідний прилад.

Схема дозволяє вимірювати споживану енергію двох незалежних навантажень та за потреби вимикати їх.

Для вимірювання струму використовується спеціалізована мікросхема ACS712, яка працює на основі ефекту Холла.

Для вимірювання напруги використовується розв’язувальний трансформатор.

Мікроконтролер за допомогою внутрішнього 10-бітного АЦП зчитує дані струму, напруги та розраховує потужність і споживану енергію. Ці дані він передає на модуль ESP32 за допомогою протоколу UART.

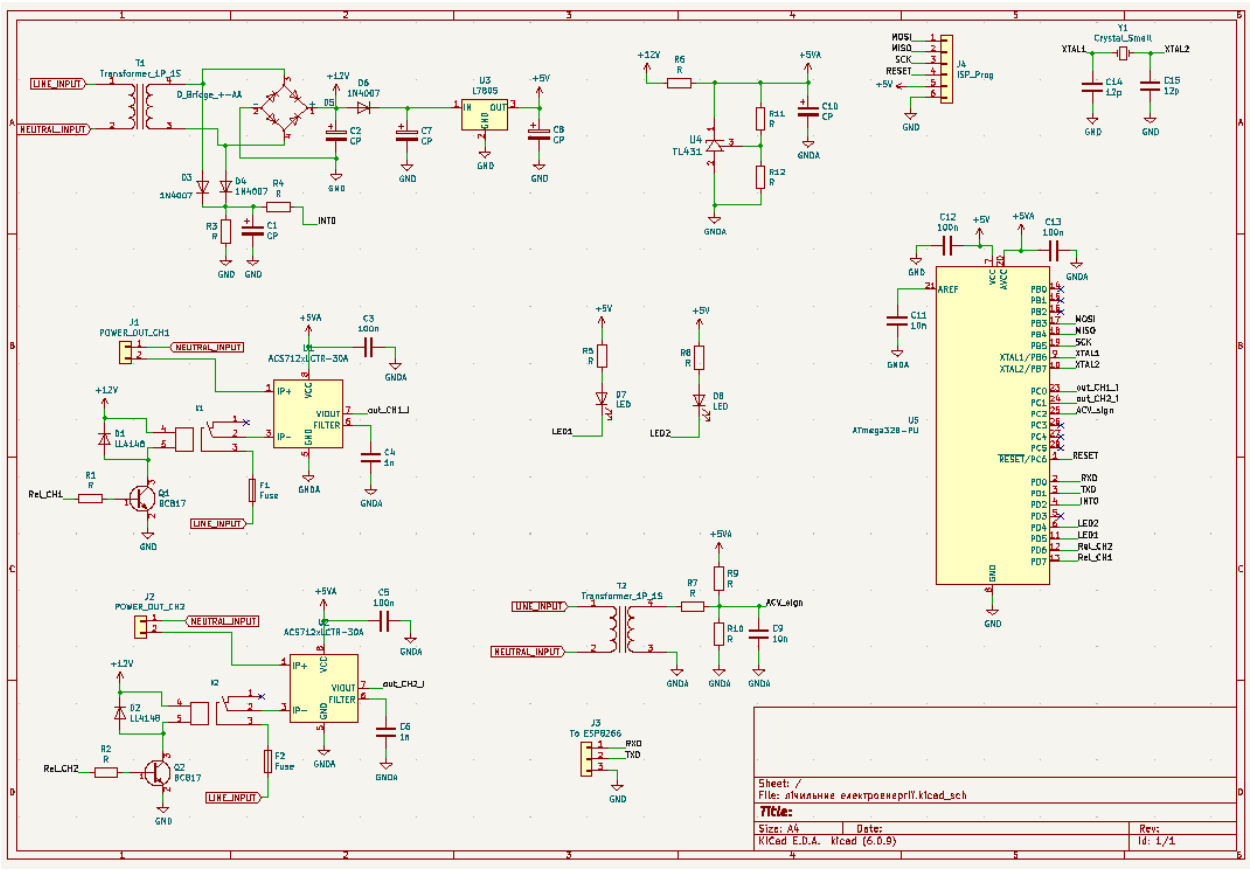
Створений прилад можна використовувати для моніторингу спожитої електроенергії з метою оптимізації її використання. Також лічильник має можливість вимикати навантаження, якщо воно перевищило заданий ліміт.

ПЕРЕЛІК ПОСИЛАНЬ

1. ESP8266 Wi-Fi модуль [Електронний ресурс]: Режим доступу URL: https://radioprog.ru/shop/merch/33
2. Програмування мікроконтролерів [Електронний ресурс]: Режим доступу URL: https://habr.com/ru/post/479250/
3. Ревич Ю. В. «Цікава електроніка» [Електронний ресурс]: Режим доступу URL: https://www.rulit.me/books/zanimatelnaya-elektronika-read-429715-1.html
4. Datasheet Atmel ATmega328P [Електронний ресурс]: Режим доступу URL: https://www.rlocman.ru/datasheet/data.html?di=505931&/ATmega328P
5. Основні відомості про протокол UART [Електронний ресурс]: Режим доступу URL: https://www.rohde-schwarz.com/cac/products/test-and-measurement/oscilloscopes/educational-content/understanding-uart_254524.html
6. Ефект Холла та його наслідки [Електронний ресурс]: Режим доступу URL: https://publish.com.ua/nashi-dni/efekt-kholla-i-jogo-naslidok.html
7. Захист струму на ACS712 [Електронний ресурс]: Режим доступу URL: https://www.radioradar.net/radiofan/radiofan_technology/current_protection_acs712_chip.html
8. Завантаження Arduino IDE [Електронний ресурс]: Режим доступу URL: https://www.arduino.cc/en/software
9. WiFi - WiFi.status() [Електронний ресурс]: Режим доступу URL: https://reference.arduino.cc/reference/en/libraries/wifi/wifi.status/

Додаток А

Принципова електрична схема розробленого лічильника електроенергії:



Додаток Б

Код для Arduino Uno:

```
#include "SoftwareSerial.h"

typedef struct
{
  char voltage;
  char current1;
  char energy1;
  char current2;
  char energy2;
}load_data;
load_data ld;

//Створення UARTу  rx в tx,  tx в rx
static const uint8_t PIN_MP3_TX = 3; //Підключення RX
static const uint8_t PIN_MP3_RX = 2; //Підключення TX

SoftwareSerial softwareSerial(PIN_MP3_RX, PIN_MP3_TX);


void setup() {
    Serial.begin(9600);
  softwareSerial.begin(9600);
    Serial.println("OK");
    ld.voltage=231;
    ld.current1=10;
    ld.energy1=23;
    ld.current2=12;
    ld.energy2=35;

}

void loop() {
    softwareSerial.write( ld.voltage);
     softwareSerial.write( ld.current1);
     softwareSerial.write( ld.energy1);
     softwareSerial.write( ld.current2);
     softwareSerial.write( ld.energy2);
     delay(500);
}
```

Додаток В

Код для ESP8266:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include "SoftwareSerial.h"


//Створення UARTу rx в tx,  tx в rx
static const uint8_t PIN_TX = 4; //Підключення TX (D2 of ESP8266)
static const uint8_t PIN_RX = 5; //Підключення RX (D1 of ESP8266)

SoftwareSerial softwareSerial(PIN_RX, PIN_TX);

const char* ssid = "capacitor"; //Вказуємо ім'я існуючої точки доступу
const char* password = "vtytpdfnbcthusq"; //Вказуємо пароль існуючої точки доступу

float global_voltage;
float global_current1;
float global_energy1;
float global_current2;
float global_energy2;

ESP8266WebServer server(80);
```

```
void handleRoot() { //Обробник запиту клієнта на кореневу адресу
  //Відправляємо клієнту
   server.send(200, "text/html", SendHTML(global_voltage,global_current1,global_energy1, global_current2,global_energy2));
}

void handleNotFound() { //Обробляємо відому помилку 404
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
 }
  server.send(404, "text/plain", message);
}

void setup(void) {
  Serial.begin(9600);
  //softwareSerial.setRxBufferSize(5);//Розмір прийомного буфера ЮАРТу = кількості переданих байт з ардуіно
```

```
  softwareSerial.begin(9600);

  WiFi.mode(WIFI_STA); //Встановлюємо модуль Wi-Fi в режим клієнта (STA)
  WiFi.begin(ssid, password); //Встановлюємо ssid та пароль від мережі, підключаємося

  while (WiFi.status() != WL_CONNECTED) { //Чекаємо на підключення до Wi-Fi
    delay(500);
    Serial.print(".");
  }

  //Виводимо інформацію про підключення
  Serial.println("");
  Serial.print("Підключено до ");
  Serial.println(ssid);
  Serial.print("IP адреса: ");
  Serial.println(WiFi.localIP());

  // Встановлюємо обробники. Можна зробити двома способами:
  server.on("/", handleRoot);

  //server.on("/inline", []() {
  //  server.send(200, "text/plain", "Чудова робота!");
//  });

  server.onNotFound(handleNotFound); //Викликається, коли обробник не призначений
```

```
  //Запускаємо сервер
  server.begin();
  Serial.println("HTTP-сервер запущений");
  global_voltage=0;
  global_current1=0;
 global_energy1=0;
 global_current2=0;
 global_energy2=0;

}

void loop(void)
{
  char temp;
  server.handleClient();
  softwareSerial.print("a");
     if (softwareSerial.available() > 4)
        {
                //Зчитуємо дані з програмного ЮАРТу:
                global_voltage = softwareSerial.read();
                global_current1 =
softwareSerial.read();
                global_energy1 = softwareSerial.read();
                global_current2 =
softwareSerial.read();
                global_energy2 = softwareSerial.read();
                while(softwareSerial.available() != 0)
                {
                  temp=softwareSerial.read();
```

```
                }
            }
}

String SendHTML(float voltage,float current1, float total_energy1,float current2, float total_energy2)
{
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr +="<title>Energy counter</title>\n";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
  ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<div id=\"webpage\">\n";
  ptr +="<h1>ESP8266 Energy counter</h1>\n";
  //Voltage
  ptr +="<p>Voltage: ";
  ptr +=voltage;
  ptr +=" V</p>";
  //Current 1
  ptr +="<p>Current of load #1: ";
```

```
  ptr +=current1;
  ptr +=" A</p>";
  //Power 1
  ptr +="<p>Power of load #1: ";
  ptr +=current1 * voltage*0.001;
  ptr +=" kW</p>";
  //Energy 1
  ptr +="<p>Total energy of load #1: ";
  ptr +=total_energy1;
  ptr +=" kW*h</p>";
  //Current 2
  ptr +="<p>Current of load #2: ";
  ptr +=current2;
  ptr +=" A</p>";
  //Power 2
  ptr +="<p>Power of load #2: ";
  ptr +=current2 * voltage*0.001;
  ptr +=" kW</p>";
  //Energy 2
  ptr +="<p>Total energy of load #2: ";
  ptr +=total_energy2;
  ptr +=" kW*h</p>";
  ptr +="</div>\n";
  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```