



Politechnika Wrocławska

Projekt zespołowy

Politechnika Wrocławska

Katedra Metrologii Elektronicznej i Fotonicznej

Automat do napojów

(Projekt wykonano w ramach kursu projektu zespołowego)

Skład grupy

Radosław Mierzwa 263675

Wiktor Michalak 263652

Mateusz Gwioździk 263658

Damian Łojko 264376

Krzysztof Zawada 263633

Spis treści

1. Wstęp	3
2. Wprowadzenie	3
3. Założenia Projektu	3
3.1 Założenia funkcjonalne:	3
3.2 Założenia projektowe:	4
3.3 Założenia konstrukcyjne:	4
4 System zarządzania akumulatorem (BMS).....	5
4.2 Wprowadzenie:	5
4.3 Funkcje i komponenty BMS.....	5
5 Opis części sprzętowej	6
5.1 Schematy układów:	6
Opis układu MPPT (Maximum Power Point Tracking) na podstawie dostarczonego schematu.....	6
Działanie układu:	8
Opis działania układu.....	10
Działanie układu.....	10
Opis działania układu z regulatorem napięcia LM7805	11
Działanie układu:	11
5.2 Opis układów:.....	13
5.3 Dobór oraz opis elementów	13
Opis układu MPPT (Maximum Power Point Tracking).....	19
6 Projekt obudowy.....	23
7 Aplikacja mobilna.....	29
8.Komunikacja z układem	30
Opis działania aplikacji stworzonej w MIT App Inventor	34
11. Opis części programowej	35
9.1 Arduino Uno.....	35
9.2 Nano.....	43
10. Wykres Gantta	47
11. Status wykonania założeń projektowych	47
12. Ryzyka oraz przebieg prac.....	49
13. Podsumowanie	50
Rekomendacje.....	51

1. Wstęp

Niniejszy dokument przedstawia projekt automatu do napojów, który łączy różnorodne technologie elektroniczne oraz mechaniczne w celu stworzenia kompleksowego i zaawansowanego systemu. Głównym celem projektu jest opracowanie i wdrożenie automatycznego urządzenia zdolnego do mieszania i chłodzenia napojów, które może być zasilane zarówno z akumulatora, jak i z zasilacza. Taka wszechstronność zasilania zapewnia nieprzerwaną pracę urządzenia w różnych warunkach i lokalizacjach.

Projekt obejmuje również implementację zaawansowanego systemu monitorowania i kontroli, który umożliwia zarządzanie urządzeniem za pomocą aplikacji mobilnej. Dzięki temu użytkownicy mogą w łatwy sposób monitorować stan automatu, kontrolować proces mieszania i chłodzenia napojów oraz dostosowywać ustawienia zgodnie ze swoimi preferencjami. Aplikacja mobilna zapewnia intuicyjny interfejs oraz liczne funkcje, które zwiększają wygodę użytkowania i umożliwiają zdalne zarządzanie urządzeniem.

2. Wprowadzenie

- Cel projektu

Celem projektu jest stworzenie automatu do napojów, który będzie w stanie przygotować, mieszać i chłodzić napoje zgodnie z preferencjami użytkownika. Urządzenie będzie zasilane z akumulatora kwasowego 12V lub zasilacza sieciowego 12 V akumulator posiada system kontroli (ładowania, rozładowywania) akumulator ładowany z zasilacza dołączonego do projektu, opcjonalnie z paneli słonecznych, a jego stan naładowania i działanie będą monitorowane i kontrolowane za pomocą aplikacji mobilnej.

- Zakres projektu

Projekt obejmuje:

- Projektowanie układów zasilania i sterowania.
- Implementację systemu chłodzenia opartego na ogniwach Peltiera.
- Integrację z aplikacją mobilną umożliwiającą kontrolę i monitorowanie.
- Testowanie i optymalizację urządzenia.

3. Założenia Projektu

3.1 Założenia funkcjonalne:

- **Zasilanie:** Akumulator kwasowy 12V, 20Ah oraz zasilacz 12V, 12.5A.

- **System zarządzania akumulatorem:** układ ograniczenia prądowego do ładowania akumulatora, układ zabezpieczający przed rozładowaniem, układ MPPT (Maximum Power Point Tracking) - śledzenie punktu mocy maksymalnej, moduły pomiaru prądu, dzielniki napięcia służące do pomiarów napięcia w projekcie, moduł zabezpieczający akumulator przed przeładowaniem (ładowanie do określonego napięcia)
- **Komponenty elektroniczne:** Arduino Nano, Arduino Uno, tranzystory MOSFET, diody, rezystory, kondensatory, moduły
- **System chłodzenia:** Ogniwa Peltiera, wentylatory.
- **Monitorowanie:** Czujnik ultradźwiękowy, moduł Bluetooth HC-05.
- **Interfejs:** Aplikacja mobilna do kontroli i monitorowania.

3.2 Założenia projektowe:

- **Ciągłość zasilania:** Akumulator musi zapewniać nieprzerwane zasilanie systemu przez minimum 2 godziny pracy.
- **Kontrola ładowania:** Proces ładowania akumulatora powinien być ściśle monitorowany i kontrolowany. System powinien automatycznie odcinać ładowanie przy osiągnięciu napięcia 12.7V i ponownie je załączać, gdy napięcie spadnie do 10V.
- **Zabezpieczenie przed nadmiernym rozładowaniem:** System musi być wyposażony w mechanizm ochrony akumulatora przed nadmiernym rozładowaniem. Tranzystor MOSFET powinien odłączać akumulator od obciążenia, gdy napięcie spadnie poniżej 9.8V.
- **Ograniczenie prądowe:** Akumulator powinien być ładowany prądem o natężeniu nieprzekraczającym 3A, aby zapewnić bezpieczny i efektywny proces ładowania.
- **Przełączanie zasilania:** System musi umożliwiać automatyczne przełączanie zasilania pomiędzy akumulatorem a siecią energetyczną, realizowane za pomocą mikrokontrolera Arduino.
- **Użycie układu 7805:** Do zasilania modułów pomiaru prądu oraz modułu Bluetooth należy zastosować układ stabilizatora napięcia 7805, aby nie obciążać zasilania Arduino.
- **Ładowanie za pomocą panelu słonecznego:** System powinien umożliwiać ładowanie akumulatora z wykorzystaniem paneli słonecznych, zapewniając maksymalne wykorzystanie odnawialnych źródeł energii.

3.3 Założenia konstrukcyjne:

1 Dioda RGB z rezystorami:

- Wykorzystanie diody RGB wraz z rezystorami o wartości 220 Ohm dla każdego koloru (czerwonego, zielonego i niebieskiego). Dioda RGB będzie sygnalizować poziom naładowania akumulatora, umożliwiając wizualną kontrolę stanu naładowania.

2 Dzielniki napięcia:

- Zastosowanie dzielników napięcia składających się z rezystorów 200 Ohm i 1000 Ohm. Taki układ zmniejsza napięcie sześciokrotnie, przekształcając napięcie 12V na 2V, co jest bezpieczne do pomiaru przez przetwornik analogowo-cyfrowy w Arduino.

3 Ograniczenie prądowe:

- Użycie wzmacniacza operacyjnego LM358 oraz układu Darlingtona BDX33C do ograniczenia prądu. Prąd wyznacza rezystor emiterowy o wartości 0,1 Ohm, a napięcie 0,3V z dzielnika (1000 Ohm do plusa i 22 Ohm do minusa) daje prąd ładowania 3A ($0,3V/0,1\text{ Ohm} = 3A$).

4 Układ zabezpieczenia przed rozładowaniem MOSFET IRF9540 z diodą Zenera TL431:

- MOSFET IRF9540 działa wraz z diodą Zenera TL431, która otwiera tranzystor, gdy napięcie na diodzie wynosi 2,5V, zgodnie z dokumentacją. Przy napięciu poniżej 9,8V dioda nie otwiera tranzystora, co odłącza układ, chroniąc akumulator przed nadmiernym rozładowaniem.

5 Układ MPPT (Maximum Power Point Tracking):

- Układ MPPT jest odpowiedzialny za optymalne wykorzystanie energii z paneli słonecznych. Składa się z elementów przedstawionych na schematach, w tym wzmacniaczy operacyjnych, rezystorów i tranzystorów. Układ MPPT dostosowuje napięcie z paneli słonecznych do maksymalnej wydajności ładowania akumulatora.

6 Pomiar prądów za pomocą modułów ACS712:

- Moduły ACS7200, działające na efekcie Halla, są używane do pomiaru prądów pompek. Dane te są przesyłane do Arduino, a następnie do aplikacji mobilnej, umożliwiając użytkownikowi monitorowanie prądów w systemie.

7 Moduł XY-CD60L do kontroli ładowania

- Moduł XY-CD60L jest używany do kontroli procesu ładowania, zapewniając, że napięcie akumulatora nie przekracza 12.7V. Moduł ten odcina ładowanie, gdy napięcie osiąga ustaloną wartość, chroniąc akumulator przed przeładowaniem.

4 System zarządzania akumulatorem (BMS)

4.2 Wprowadzenie:

System zarządzania baterią (BMS) jest kluczowym elementem projektu, odpowiedzialnym za kontrolę zasilania i ładowania akumulatora kwasowego, zapewniając jego bezpieczną i efektywną pracę.

4.3 Funkcje i komponenty BMS

BMS w projekcie automatu do napojów pełni następujące funkcje:

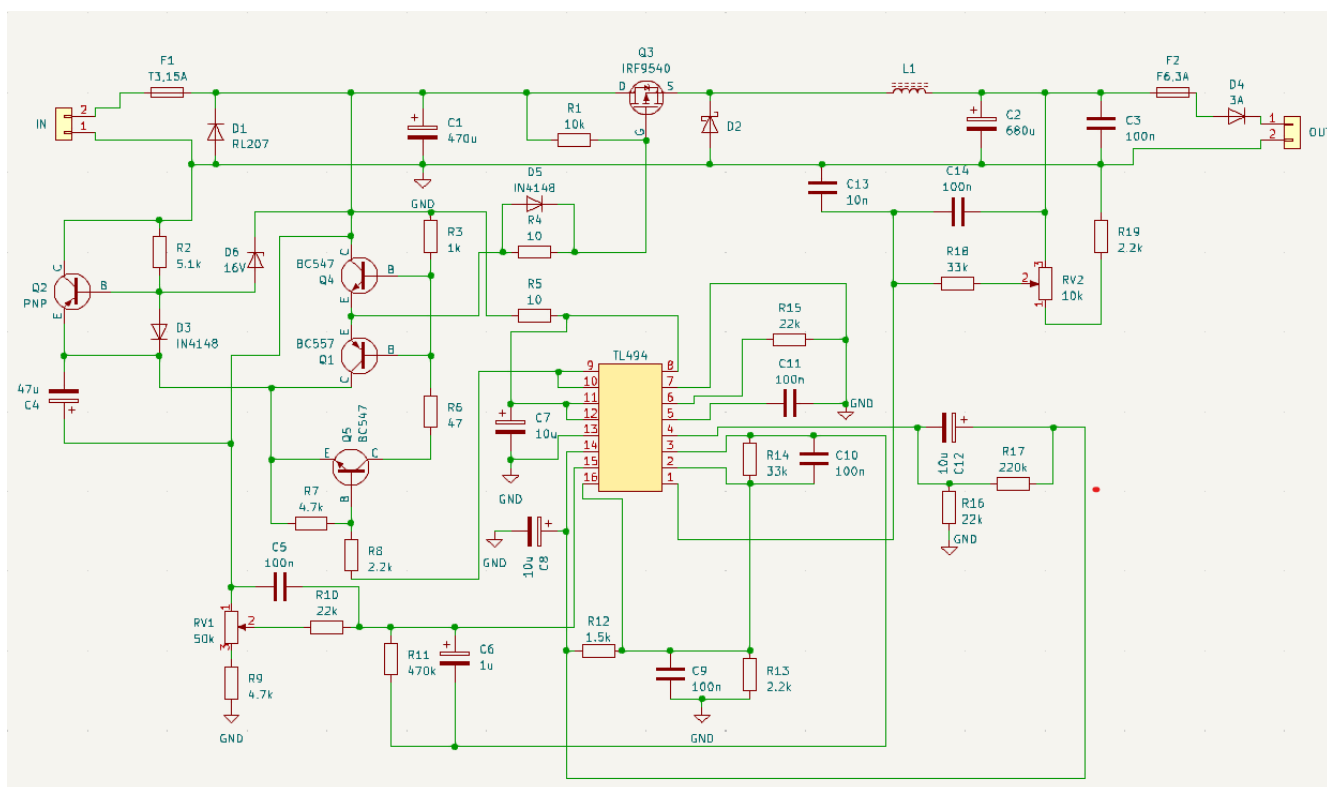
- **Monitorowanie napięcia i prądu ładowania akumulatora:** Zapewnia ochronę przed nadmiernym rozładowaniem i przeładowaniem.
- **Zarządzanie ładowaniem:** Steruje procesem ładowania akumulatora z paneli słonecznych lub zasilacza, optymalizując ładowanie za pomocą układu MPPT.
- **Zabezpieczenia:** Obejmuje zabezpieczenia przed zwarciami, przeciążeniami, nadmiernym rozładowaniem oraz ładowanie określonym prądem za pomocą tranzystora, bezpieczników, modułów oraz ograniczenia prądowego.
- **Sygnalizacja:** Informuje o stanie naładowania akumulatora za pomocą diody RGB oraz interfejsu aplikacji mobilnej.

5 Opis części sprzętowej

5.1 Schematy układów:

Schematy układów zostały załączone w postaci zdjęć, przedstawiających szczegółowe połączenia i komponenty użyte w projekcie. W skład układów wchodzi:

- Układ MPPT do sterowania panelem fotowoltaicznym.



Rys 1. Schemat ideowy układu MPPT

Opis układu MPPT (Maximum Power Point Tracking) na podstawie dostarczonego schematu

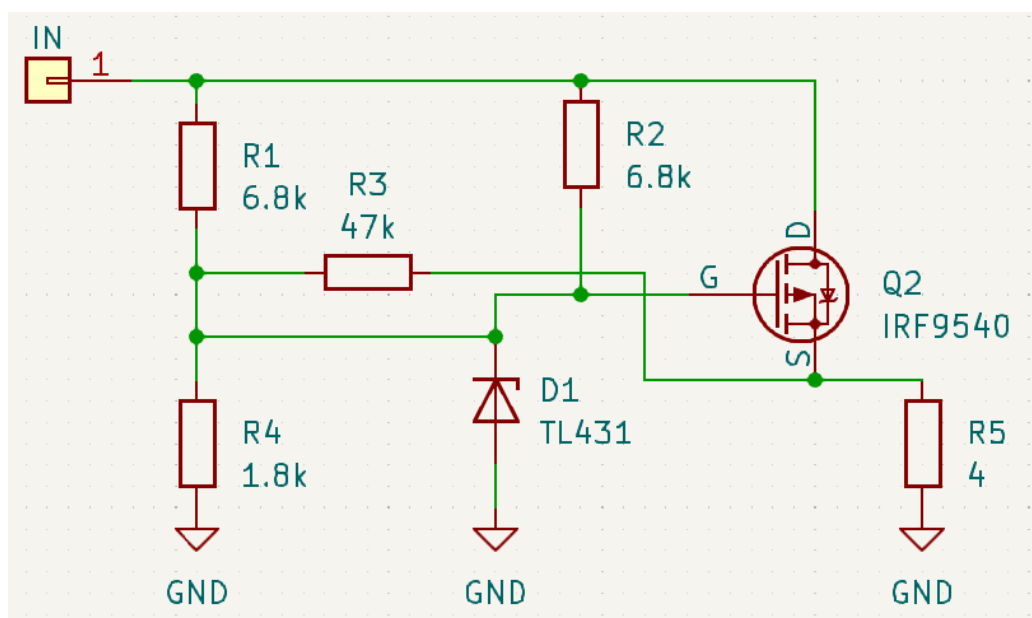
Układ MPPT (Maximum Power Point Tracking) optymalizuje ładowanie akumulatora z paneli słonecznych. Poniżej znajduje się szczegółowy opis poszczególnych komponentów i ich funkcji w układzie:

1. **Wejście z panelu słonecznego (IN):**
 - **F1 (15A):** Bezpiecznik chroniący układ.
 - **D1 (RL207):** Dioda, która zabezpiecza przez odwrotną polaryzacją.
 - **C1 (470uF):** Kondensator wejściowy.
2. **Ogranicznik napięcia:**
 - **R2 (5.1k):** Rezystor ogranicznika napięcia

- **Q2 (BC547):** Tranzystor PNP w układzie wspólnego kolektora.
 - **D3 (1N4148):** Dioda zabezpieczająca.
 - **D6 (16V):** Dioda Zenera stabilizująca napięcie.
 - **C4 (47uF):** Kondensator filtrujący.
- 3. Kontroler TL494:**
- **TL494:** Układ PWM zarządzający procesem MPPT.
 - **Pin 1 (GND):** Masa.
 - **Pin 2 (Error Amp Inverting Input):** Wejście odwracające wzmacniacza błęd.
 - **Pin 3 (Feedback):** Wyjście sprzężenia zwrotnego.
 - **Pin 4 (Dead Time Control):** Sterowanie czasem martwym.
 - **Pin 5 (Control Voltage):** Napięcie kontrolne.
 - **Pin 6 (Timing Capacitor):** Kondensator czasowy.
 - **Pin 7 (Timing Resistor):** Rezystor czasowy.
 - **Pin 8 (GND):** Masa.
 - **Pin 9 (Collector):** Kolektor tranzystora wewnętrznego.
 - **Pin 10 (Emitter):** Emiter tranzystora wewnętrznego.
 - **Pin 11 (Vcc):** Zasilanie.
 - **Pin 12 (Output Control):** Kontrola wyjścia.
 - **Pin 13 (Reference Output):** Wyjście referencyjne.
 - **Pin 14 (Error Amp Non-inverting Input):** Wejście nieodwracające wzmacniacza błęd.
 - **Pin 15 (Error Amp Inverting Input):** Wejście odwracające wzmacniacza błęd.
 - **Pin 16 (Vcc):** Zasilanie.
 - **C7(10uF):** Kondensator wygładzający.
 - **R5(10):** Rezystor zasilający.
- 4. Driver MOSFET'a (IRF9540):**
- **Q3 (IRF9540):** MOSFET sterujący przepływem prądu do obciążenia (akumulatora).
 - **Q4 (BC547):** Tranzystor PNP, który ma kolektor podłączony do plusa (wspólny kolektor).
 - **Q1 (BC557):** Tranzystor NPN, kolektor podłączony do ogranicznika napięcia (wspólny kolektor).
 - **R3(1k), R6(47):** Rezystory podłączone pod bazę tranzystorów
 - **R4(10):** Rezystor na emiterze drivera
 - **D5(1N4148):** Dioda równolegle podłączona do R4
- 5. Przekładanie napięcia z ogranicznika napięcia na driver:**
- **Q5(BC547):** Podaje napięcie na driver.
 - **R7(4,7k), R8(2,2k):** Rezystory podłączone do bazy i emitera tranzystora.
- 6. Przetwornica step-down:**
- **D2 (3A):** Dioda shottkiego.
 - **L1 (100uH):** Dławik filtrujący.
 - **C2 (680uF):** Kondensator wyjściowy.
 - **F2(3A):** Bezpiecznik chroniący przed sprzężeniem zwrotnym.
 - **D4 (3A):** Dioda na wyjściu, zapobiegająca powrotowi napięcia z akumulatora do układu, chroniąc przed przepływem wstecznym.
- 7. Elementy ustalające częstotliwość pracy TL494:**
- **R15(22k), C11(100nF):** ustalają częstotliwość pracy (45kHz).
- 8. Układ soft startu**
- **R16(22k), R17(220k), C11(10uF):** elementy soft startu.
- 9. Układ sprzężenia zwrotnego wyjścia:**
- **RV2(10k), R18(33k), R19(2,2k), C3(100nF):** elementy układu sprzężenia zwrotnego wyjścia.
 - **C13(10nF), C14(100nF):** kondensatory antyoscyłacyjne.
 - **R13(2,2k), R14(33k), C10(100nF):** dzielnik napięcia.

10. Układ sprzężenia zwrotnego wejścia:

- **RV1(50k), R9(4,7k), R19(22k), C3(100nF):** elementy układu sprzężenia zwrotnego wejścia.
 - **R11(470k), C6(1uF):** układ antyoscylicyjny.
 - **R12(1,5k), C9(100nF), R13(2,2k):** dzielnik napięcia
 -
- **Układ zabezpieczający przed rozładowaniem akumulatora**



Rys 2. Schemat ideowy układu zabezpieczającego przed rozładowaniem akumulatora

Działanie układu:

1. Wejście zasilania (IN):

- Napięcie wejściowe jest podawane na punkt oznaczony jako "IN".

2. Dzielnik napięcia:

- Rezystory R1 (6.8k) i R4 (1.8k) tworzą dzielnik napięcia, który redukuje napięcie wejściowe do bezpiecznego poziomu dla diody Zenera TL431.
- Rezystor R3 (47k) jest połączony szeregowo z R1 i R4, wpływając na napięcie referencyjne.

3. Dioda Zenera (TL431):

- Dioda TL431 działa jako precyzyjny regulator napięcia, stabilizując napięcie na swojej katodzie.
- Gdy napięcie na anodzie TL431 (przez dzielnik napięcia) osiąga napięcie referencyjne (około 2.5V), TL431 zaczyna przewodzić.

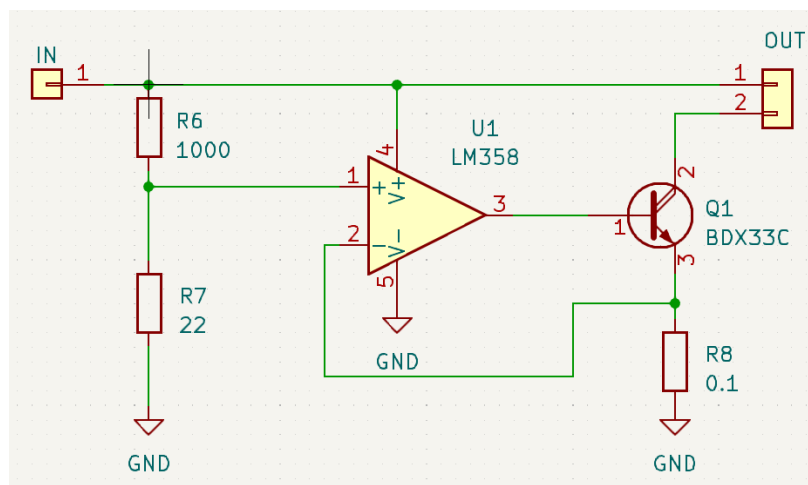
4. Sterowanie MOSFET-em (Q2):

- Gdy TL431 przewodzi, napięcie na bramce MOSFET-a Q2 (IRF9540) jest ściągane do masy przez TL431 i rezystor R2 (6.8k).
- Spadek napięcia na bramce MOSFET-a powoduje, że tranzystor Q2 zaczyna przewodzić.
- Przewodzenie MOSFET-a Q2 pozwala na przepływ prądu z źródła (S) do drenu (D), umożliwiając zasilanie obciążenia podłączonego do wyjścia.
- **Moduł zabezpieczający układ przed przeladowaniem akumulatora**



Rys 3. Moduł XY-CD60L

- **Układ ograniczenia prądu do ładowania akumulatora**



Rys 4. Schemat ideowy układu ograniczenia prądu do ładowania akumulatora

Opis działania układu

Układ przedstawiony na schemacie jest regulatorem prądu wykorzystującym wzmacniacz operacyjny LM358 oraz tranzystor darlingtona BDX33C. Poniżej znajduje się szczegółowy opis działania poszczególnych komponentów i ich funkcji w układzie.

Komponenty:

1. **R6 (1000 Ohm), R7 (22 Ohm), R8 (0.1 Ohm):** Rezystory
2. **U1 (LM358):** Wzmacniacz operacyjny
3. **Q1 (BDX33C):** Tranzystor darlingtona

Działanie układu

1. Wejście zasilania (IN):

- Napięcie wejściowe jest podawane na punkt oznaczony jako "IN".

2. Dzielnik napięcia:

- **R6 (1000 Ohm)** i **R7 (22 Ohm)** tworzą dzielnik napięcia, który dostarcza napięcie odniesienia do wejścia nieodwracającego (+) wzmacniacza operacyjnego LM358.

3. Wzmacniacz operacyjny (U1 - LM358):

- **Pin 2:** Wejście odwracające (-) jest podłączone do rezystora **R8 (0.1 Ohm)**, który jest rezystorem pomiarowym umieszczonym w obwodzie kolektora tranzystora Q1.
- **Pin 3:** Wejście nieodwracające (+) jest podłączone do dzielnika napięcia utworzonego przez rezystory R6 i R7.
- **Pin 1:** Wyjście wzmacniacza operacyjnego jest podłączone do bazy tranzystora darlingtona Q1.

4. Tranzystor darlingtona (Q1 - BDX33C):

- Tranzystor Q1 steruje przepływem prądu przez rezystor R8 do obciążenia podłączonego do wyjścia (OUT).
- Gdy wzmacniacz operacyjny LM358 dostarcza odpowiednie napięcie na bazę tranzystora Q1, tranzystor przewodzi, umożliwiając przepływ prądu.

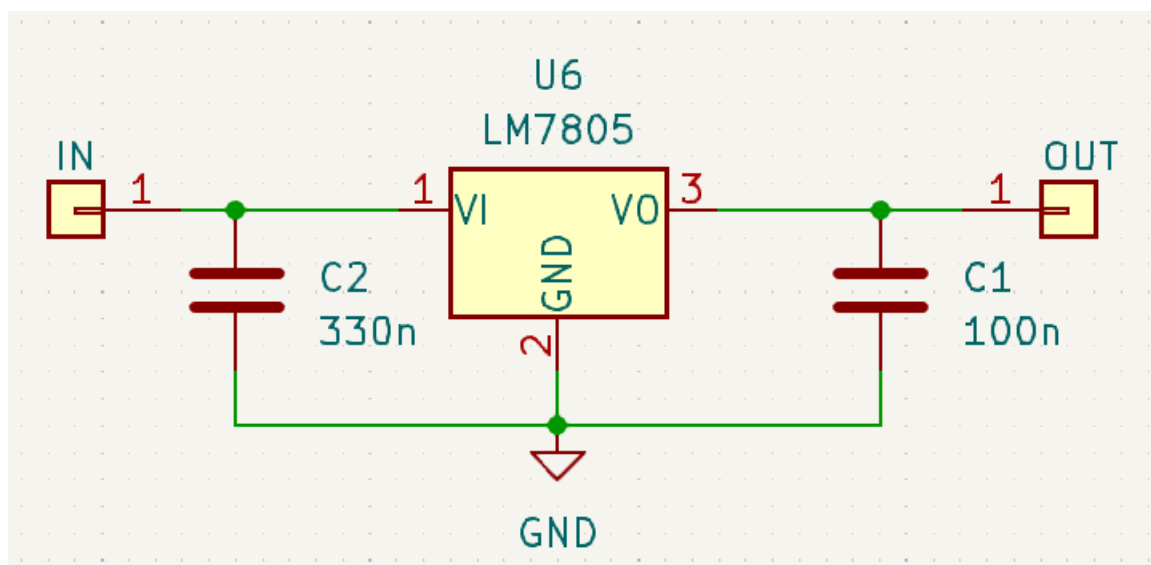
5. Pomiar prądu:

- Prąd przepływający przez tranzystor Q1 powoduje spadek napięcia na rezystorze pomiarowym R8 (0.1 Ohm).
- Spadek napięcia na R8 jest proporcjonalny do przepływającego prądu (Prawo Ohma: $V = I * R$).
- Napięcie to jest podawane na wejście odwracające (-) wzmacniacza operacyjnego LM358.

6. Regulacja prądu:

- Wzmacniacz operacyjny LM358 porównuje napięcie na wejściu nieodwracającym (+) (pochodzące z dzielnika napięcia) z napięciem na wejściu odwracającym (-) (pochodzące z rezystora R8).
- Jeśli napięcie na wejściu odwracającym (-) przekracza napięcie odniesienia na wejściu nieodwracającym (+), wzmacniacz operacyjny zmniejsza napięcie na bazie tranzystora Q1, ograniczając prąd przepływający przez obciążenie.

- W ten sposób układ stabilizuje prąd wyjściowy na poziomie określonym przez dzielnik napięcia i wartość rezystora R8.
- Bezpieczniki wyjścia zasilacza i akumulatora
- Układ 7805 do zasilania modułów pomiaru prądu ACS712



Rys 5. Schemat ideowy stabilizatora opartego na 7805

Opis działania układu z regulatorem napięcia LM7805

Układ przedstawiony na schemacie jest prostym regulatorem napięcia wykorzystującym stabilizator liniowy LM7805. Poniżej znajduje się szczegółowy opis działania poszczególnych komponentów i ich funkcji w układzie.

Komponenty:

1. **C2 (330nF):** Kondensator filtrujący na wejściu.
2. **C1 (100nF):** Kondensator filtrujący na wyjściu.
3. **U6 (LM7805):** Stabilizator napięcia 5V.

Działanie układu:

1. Wejście zasilania (IN):

- Napięcie wejściowe jest podawane na punkt oznaczony jako "IN". LM7805 jest zaprojektowany do pracy z napięciem wejściowym od 7V do 35V.

2. Filtracja wejściowa:

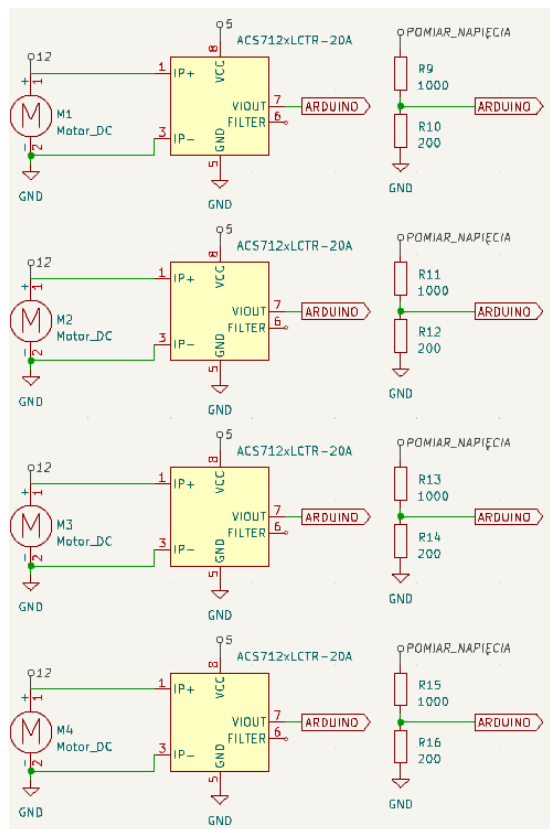
- **C2 (330nF):** Kondensator umieszczony na wejściu stabilizatora LM7805 ma za zadanie filtrować zakłócenia oraz wygładzać napięcie wejściowe. Kondensator ten pomaga w eliminacji szumów oraz niestabilności zasilania, co jest szczególnie ważne przy nieregularnych źródłach zasilania.

3. Stabilizator napięcia (U6 - LM7805):

- **Pin 1 (VI):** Wejście napięcia. Napięcie z punktu IN jest podawane na to wejście.
- **Pin 2 (GND):** Masa. Wszystkie napięcia odniesienia są mierzone względem tego pinu.
- **Pin 3 (V0):** Wyjście stabilizowanego napięcia. LM7805 zapewnia stałe napięcie wyjściowe 5V.

4. Filtracja wyjściowa:

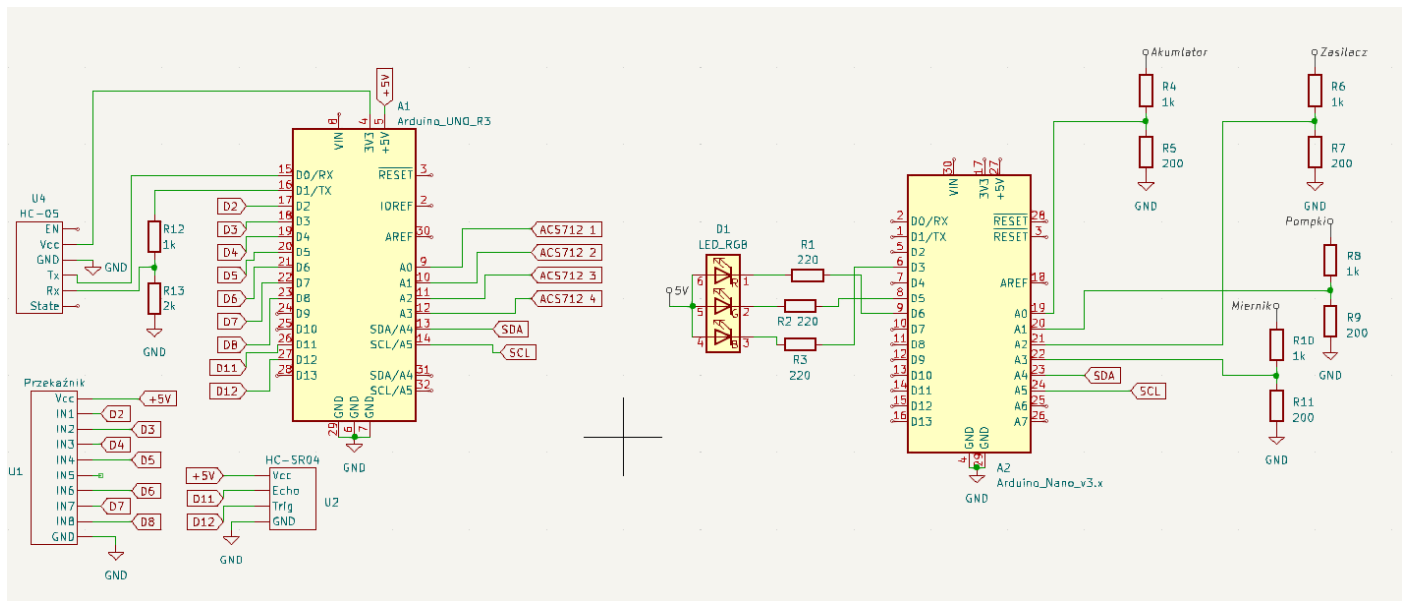
- **C1 (100nF):** Kondensator umieszczony na wyjściu stabilizatora LM7805 ma za zadanie filtrować wszelkie pozostałe zakłócenia oraz wygładzać napięcie wyjściowe. Zapewnia to stabilne i czyste napięcie 5V na wyjściu.
- Dzielnik napięć do pomiarów oraz moduły do pomiaru prądów



Rys 6. Schemat ideowy modułów oraz dzielników napięć do pomiaru napięcia

Układ służy do pomiaru prądów i napięć i wysyłania ich wartości na wejścia analogowe Arduino.

Schemat ideowy połączeń arduino Uno i Nano:



Rys 6 Schemat całego układu

5.2 Opis układów:

- **Układ MPPT:** Odpowiada za optymalne wykorzystanie energii z paneli słonecznych, dostosowując napięcie do maksymalnej wydajności.
- **Układ zabezpieczający:** Monitoruje napięcie akumulatora i odłącza go w celu zabezpieczenia przed nadmiernym rozładowaniem.
- **Układ pomiarowy:** Rejestruje wartości prądów i napięć, przesyłając je do aplikacji mobilnej w celu monitorowania stanu systemu.
- **Moduł zabezpieczający przed przeładowaniem akumulatora**
- **Układ ograniczenia prądowego do ładowania akumulatora**

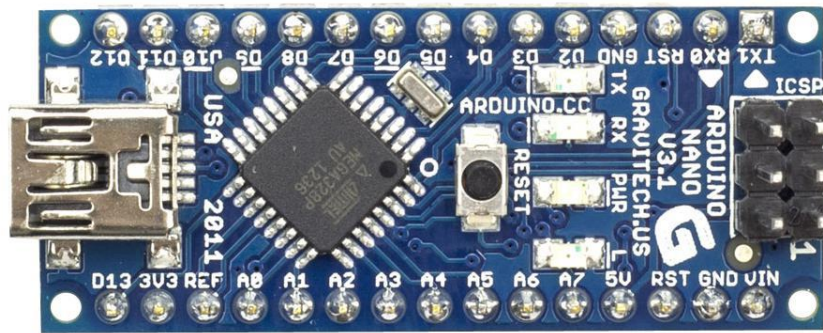
5.3 Dobór oraz opis elementów

Komponenty zostały dobrane na podstawie wymagań prądowych i napięciowych systemu. Użyto popularnych i łatwo dostępnych elementów, takich jak tranzystory MOSFET, diody zabezpieczające, rezystory, kondensatory oraz moduły Arduino.

Schematy układów BMS są załączone w postaci zdjęć (pkt 5), przedstawiających szczegółowe połączenia i komponenty użyte w systemie. Główne elementy to:

- **Tranzystory MOSFET:** Sterujące przepływem prądu i zabezpieczające akumulator.
- **Moduł MPPT:** Optymalizujący ładowanie z paneli słonecznych.
- **Czujniki:** Monitorujące napięcia i prądy.
- **Arduino Nano:** Centralny kontroler systemu BMS.

1.Arduino Nano



Rys 7. Arduino Nano

Arduino Uno:

- **Funkcja:** Arduino Uno jest centralnym kontrolerem systemu zarządzania baterią (BMS) w projekcie automatu do napojów.
- **Specyfikacja (Nano):**
 - **Procesor:** ATmega328
 - **Napięcie pracy:** 5V
 - **Napięcie wejściowe (rekomendowane):** 7-12V
 - **Wejścia/Wyjścia:** 14 cyfrowych pinów I/O (z czego 6 może być używane jako wyjścia PWM), 8 analogowych wejść
 - **Pamięć:** 32KB pamięci flash (ATmega328) z 2KB używanymi przez bootloader, 2KB SRAM, 1KB EEPROM
 - **Prędkość zegara:** 16 MHz

Arduino Uno w projekcie pełni kluczową rolę, sterując i monitorując cały system zarządzania baterią. Posiada liczne wejścia i wyjścia, które pozwalają na podłączenie różnych czujników i modułów, takich jak czujniki prądu ACS712, moduł Bluetooth HC-05 oraz inne elementy systemu BMS.

2.Dioda RGB



Rys 8. dioda RGB

Dioda RGB:

- **Funkcja:** Dioda RGB jest używana do sygnalizacji poziomu naładowania akumulatora w projekcie automatu do napojów. Każdy z trzech kolorów diody (czerwony, zielony, niebieski) może być kontrolowany niezależnie, umożliwiając wyświetlanie różnych stanów naładowania poprzez kombinacje kolorów.
- **Specyfikacja:**
 - **Typ diody:** LED RGB (Red-Green-Blue)
 - **Napięcie pracy:**
 - Czerwony: typowo 2V
 - Zielony: typowo 3.2V
 - Niebieski: typowo 3.2V
 - **Prąd pracy:** Zazwyczaj 20mA na każdy kolor

Rezystory:

- **Wartości rezystorów:** Każdy kolor diody RGB jest połączony z rezystorem o wartości 220 Ohm, aby ograniczyć prąd i zapobiec uszkodzeniu diody.
 - **Czerwony:** 220 Ohm
 - **Zielony:** 220 Ohm
 - **Niebieski:** 220 Ohm

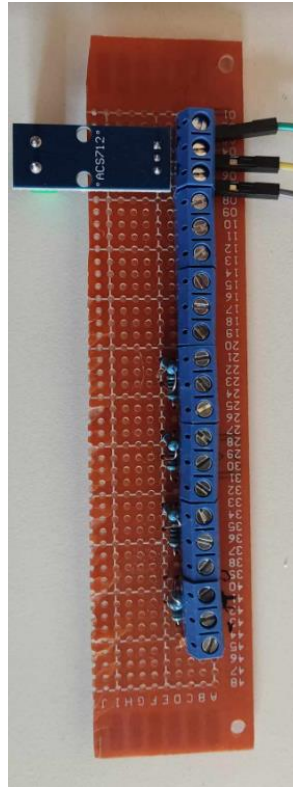
Tab 1. Określanie procentowe naładowania akumulatora

12V Battery Voltage	Volts Per Cell	State of charge
12.7	2.12	100%
12.5	2.08	90%
12.42	2.07	80%
12.32	2.05	70%
12.2	2.03	60%
12.06	2.01	50%
11.9	1.98	40%
11.79	1.96	30%
11.58	1.93	20%
11.31	1.89	10%
10.5	1.75	0%

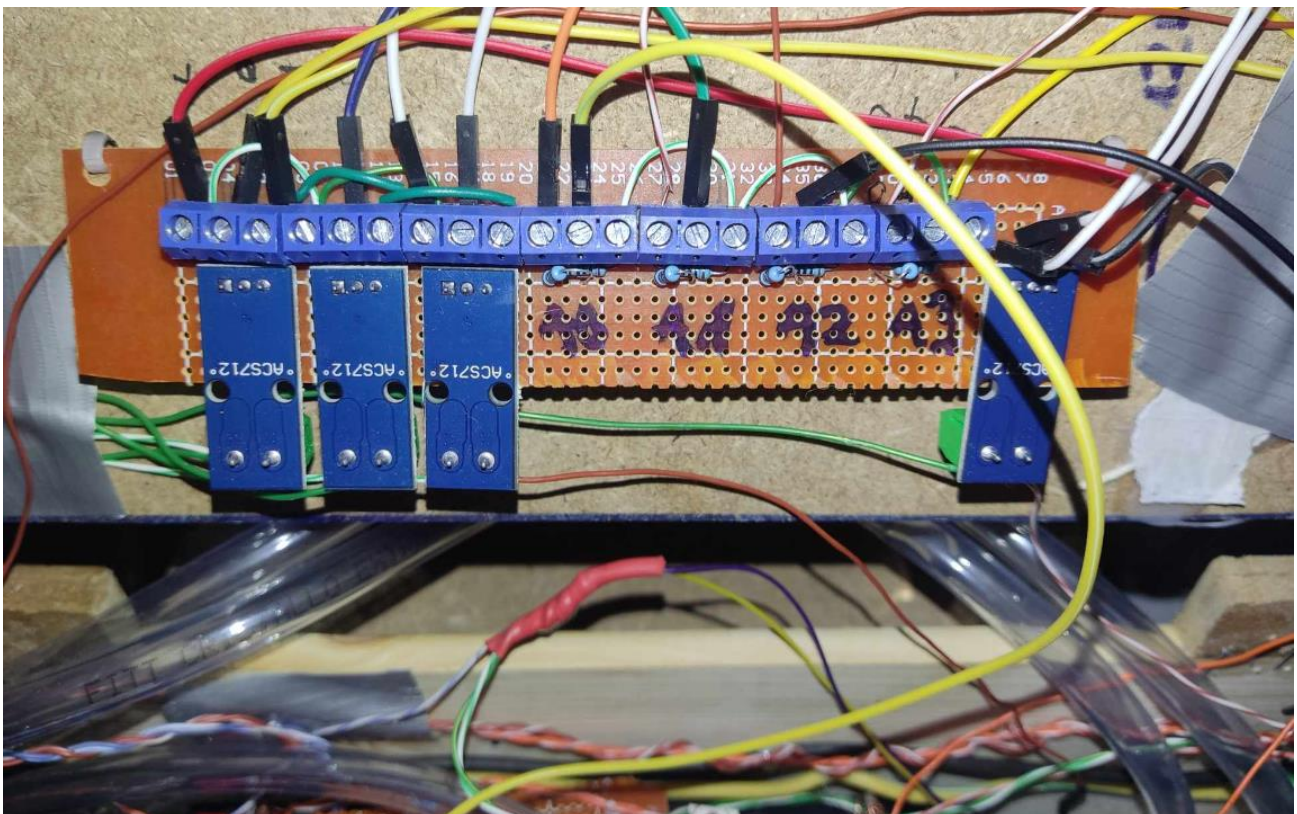
3. Moduł ACS712 oraz dzielniki napięć.



Rys 9. Moduł ACS712



Rys 10. Moduł ACS712 i dzielniki napięcia



Rys 11. Dzielniki napięcia i moduły prądów w układzie

Czujnik prądu ACS712 - 20A:

- **Funkcja:** Moduł z czujnikiem prądu ACS712 jest używany do precyzyjnego pomiaru prądów AC i DC w zakresie od 0 do 20 A. W projekcie automatu do napojów, czujnik ten monitoruje prądy przepływające przez system, dostarczając dane do Arduino Nano, które następnie przesyła informacje do aplikacji mobilnej.
- **Specyfikacja:**
 - **Zakres pomiaru:** 0 A do 20 A (AC/DC)
 - **Czułość:** 100 mV/A
 - **Napięcie zasilania:** 5V
 - **Technologia:** Efekt Halla
 - **Złącze:** 4-pinowe (VCC, GND, OUT, dodatkowy pin)

Podłączenie modułu ACS712:

1. **Zasilanie:** Podłącz 5V do pinu VCC oraz masę do pinu GND.
2. **Sygnał wyjściowy:** Pin OUT podłącz do analogowego wejścia Arduino Nano.
3. **Wejścia prądowe:** Złącza śrubowe są używane do podłączenia obciążenia, przez które przepływa mierzony prąd.

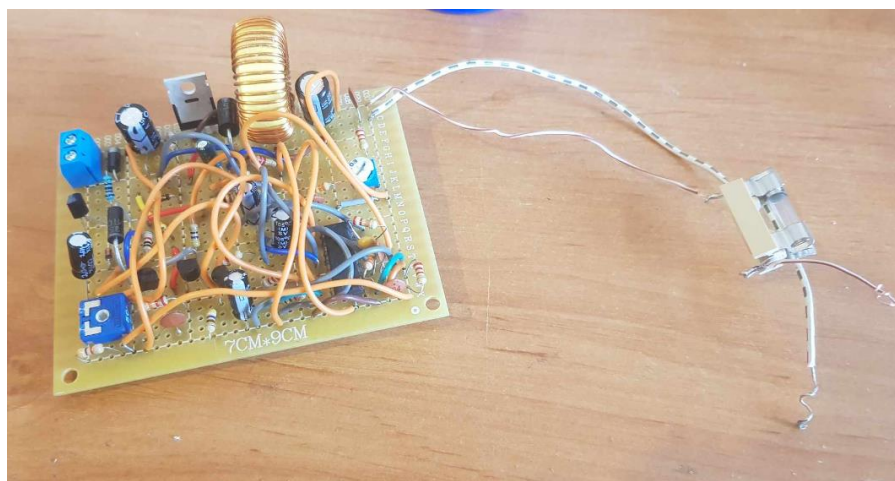
Schemat podłączenia:

- VCC (5V) do zasilania modułu.
- GND do masy.
- OUT do analogowego wejścia na Arduino Nano (np. A0).
- Złącza śrubowe do obciążenia i masy.

Opis działania:

- Czujnik ACS712 mierzy prąd przepływający przez obciążenie przy użyciu efektu Halla, co pozwala na bezkontaktowy pomiar prądu. Sygnał wyjściowy jest proporcjonalny do mierzonego prądu (100 mV na każdy 1 A), co jest odczytywane przez Arduino Nano. Dane te są następnie przesyłane do aplikacji mobilnej, umożliwiając monitorowanie prądów w systemie w czasie rzeczywistym.

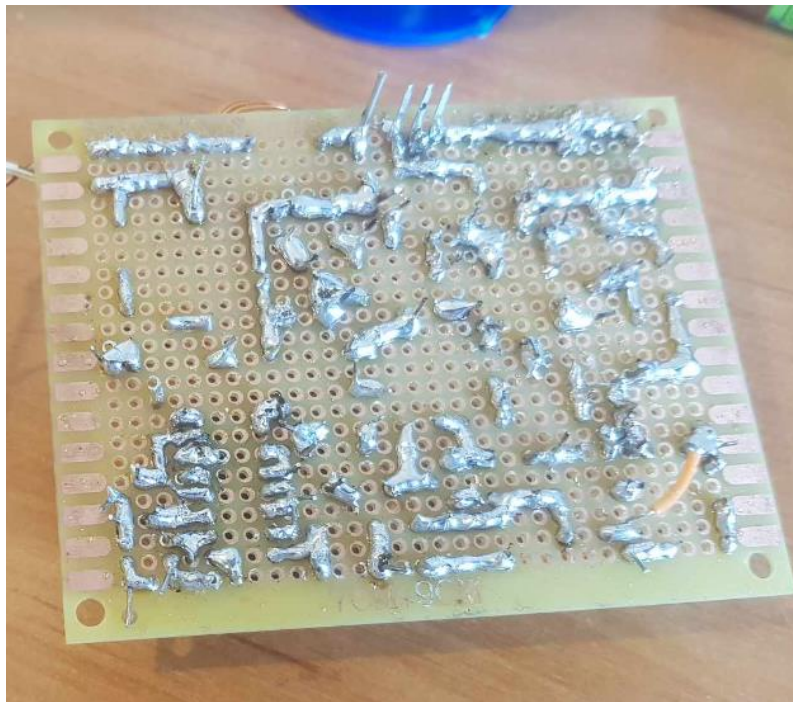
4.Układ MPPT (Schemat pkt. 5)



Rys 12. Układ MPPT



Rys 13. Układ MPPT front



Rys 14. Układ MPPT back

Opis układu MPPT (Maximum Power Point Tracking)

Układ MPPT: Układ MPPT (Maximum Power Point Tracking) jest kluczowym komponentem w systemach zasilania z paneli słonecznych, umożliwiającym maksymalne wykorzystanie dostępnej energii słonecznej. MPPT śledzi punkt maksymalnej mocy (MPP) panelu słonecznego i dostosowuje warunki obciążenia, aby zapewnić optymalne napięcie i prąd dla ładowania akumulatora.

Komponenty układu MPPT

1. Kontroler TL494:

- **Funkcja:** Kontroler TL494 jest używany do regulacji PWM (Pulse Width Modulation) w celu kontrolowania napięcia i prądu dostarczanego do akumulatora. Działa jako serce układu MPPT, zarządzając konwersją mocy.
- **Specyfikacja:**
 - Podwójny generator PWM.
 - Zakres napięcia zasilania: 7V-40V.
 - Możliwość regulacji częstotliwości.

2. Tranzystor MOSFET (IRF9540):

- **Funkcja:** MOSFET-y są używane do sterowania napięciem w układzie. W układzie MPPT, MOSFET-y działają jako przełączniki, które są kontrolowane przez sygnały PWM z kontrolera TL494.
- **Specyfikacja:**
 - Niskie napięcie progowe.
 - Szybkie przełączanie.

3. Rezystory i kondensatory:

- **Funkcja:** Rezystory i kondensatory są używane do ustawienia punktu pracy układu, filtracji sygnałów oraz stabilizacji pracy kontrolera TL494.
- **Specyfikacja:**
 - Rezystory o odpowiednich wartościach dla filtracji i stabilizacji.

- Kondensatory do filtracji sygnału PWM.

Schemat działania układu MPPT

1. Wejście i Zasilanie

- **Panel Słoneczny:** Maksymalne napięcie wejściowe to 35V, ale układ TL494 pracuje do 32V. Zaleca się margines bezpieczeństwa między maksymalnym napięciem panelu a układem TL494.
- **Dioda:** Umieszczona na wejściu, chroni przed odwrotną polaryzacją.
- **Kondensator Wejściowy:** Stabilizuje napięcie wejściowe.
- **MOSFET typu P:** Użyty zamiast MOSFETa typu N, aby uprościć pomiar napięcia wyjściowego względem masy. MOSFET typu P przełącza napięcie względem masy, co jest łatwiejsze do kontrolowania.

2. Przetwornica Step-Down

- **Cewka:** Służy do magazynowania energii; jej indukcyjność nie jest krytyczna, ale musi być nawinięta grubym drutem (w tym przypadku 100mH).
- **Dioda Schottky'ego:** Dioda o niskim spadku napięcia, używana do szybkiego przełączania.
- **Kondensator Wyjściowy:** Stabilizuje napięcie na wyjściu.

3. Układ Ograniczający Napięcie

- **Komponenty:** Tranzystor (np. NPN), dioda Zenera (16V), rezystory i dodatkowe elementy pasywne.
- **Działanie:**
 - **Normalne Warunki:** Gdy napięcie wejściowe jest poniżej napięcia przewodzenia diody Zenera (16V), tranzystor działa w układzie wspólnego kolektora, przekazując prawie to samo napięcie na emiter co na bazę (minus względem plusa zasilania).
 - **Wysokie Napięcie:** Gdy napięcie wzrośnie powyżej 20V, dioda Zenera przewodzi, ograniczając napięcie na bazie tranzystora względem plusa (około -16V). Na emiterze tranzystora pojawia się napięcie ograniczone do około -15,3V względem plusa.
 - **Bezpieczne Przełączanie:** Dzięki temu napięcie na bramce MOSFETa nigdy nie spada poniżej -15V, co pozwala na bezpieczne przełączanie MOSFETa przy wysokich napięciach wejściowych.

4. Sterowanie MOSFET-em

- **Driver Bramy MOSFET-a:** Składa się z dwóch tranzystorów. Jeden z nich jest podłączony do ogranicznika napięcia, a drugi do zasilania. Umożliwia to uzyskanie większej wydajności prądowej.
- **Sterowanie z TL494:** TL494 steruje tranzystorem, który przekazuje napięcie z ogranicznika na driver bramki MOSFET-a. Driver bramki MOSFET-a przekazuje to napięcie na bramkę MOSFET-a z większą wydajnością prądową, dzięki czemu MOSFET jest bezpiecznie przełączany.

5. TL494 - Kontroler PWM

- **Zasilanie TL494:** Zasilany przez rezystor 10 omów, z kondensatorem wygładzającym, aby eliminować tętnienia.
- **Ustawienie Częstotliwości:** Elementy ustalające częstotliwość pracy, teoretycznie 45 kHz, w rzeczywistości 60 kHz.
- **Soft Start:** Kondensator ładuje się po włączeniu układu, co powoduje stopniowe zwiększanie cyklu pracy przez TL494, zapobiegając gwałtownemu wzrostowi prądu.

6. Sprężenie Zwrotne

- **Kontrola Napięcia Wyjściowego:** Dwa układy, jeden monitorujący napięcie wyjściowe, drugi napięcie wejściowe. Jeśli napięcie wyjściowe spada poniżej ustawionej wartości, TL494 zwiększa cykl pracy, aby ustabilizować napięcie.
- **Kontrola Napięcia Wejściowego:** Monitoruje napięcie panelu słonecznego, aby nie spadło poniżej punktu pracy. Gdy napięcie spada poniżej ustawionego punktu, TL494 zmniejsza cykl pracy, co podwyższa napięcie panelu.

7. Ochrona Układu

- **Bezpiecznik:** Chroni przed zbyt dużym prądem.
- **Dodatkowa Dioda:** Umieszczona za bezpiecznikiem, aby zapobiec przepływowi prądu z akumulatora do układu.

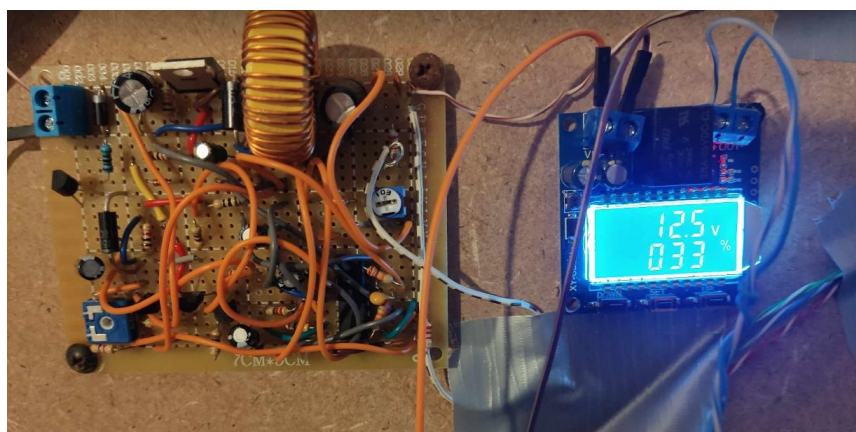
8. Dostosowanie do Ładowania Akumulatora

- **Układ MPPT:** Dostosowuje prąd ładowania akumulatora, aby utrzymywać napięcie panelu w punkcie maksymalnej mocy. Potencjometry pozwalają na ustawienie maksymalnego napięcia ładowania akumulatora, co zapobiega jego przeładowaniu.

Korzyści stosowania MPPT

- **Wzrost wydajności:** Układ MPPT zwiększa wydajność systemu PV (fotowoltaicznego) o 20-30% w porównaniu do konwencjonalnych regulatorów ładowania.
- **Optymalizacja energii:** Umożliwia maksymalne wykorzystanie dostępnej energii słonecznej poprzez ciągłe dostosowywanie punktu pracy paneli słonecznych.

5. Moduł XY-CD60L



Rys 15. Moduł XY-CD60L

Moduł XY-CD60L to zaawansowany układ sterowania, który znajduje zastosowanie w systemach zarządzania zasilaniem, takich jak nasz projekt automatu do napojów. Jego główną funkcją jest kontrola napięcia akumulatora i zarządzanie procesem ładowania, co zapewnia optymalne działanie systemu i

przedłuża żywotność akumulatora. XY-CD60L umożliwia precyzyjne ustawienie napięcia odcięcia, przy którym ładowanie akumulatora zostaje przerwane. W naszym przypadku moduł jest skonfigurowany tak, aby przerwać ładowanie, gdy napięcie akumulatora osiągnie 12,7V. Taka wartość została wybrana, aby zapewnić, że akumulator nie będzie przeładowany, co mogłoby prowadzić do jego uszkodzenia lub skrócenia żywotności. Moduł XY-CD60L jest wyposażony w cyfrowy wyświetlacz, który pokazuje aktualne napięcie akumulatora, co umożliwia ciągłe monitorowanie stanu ładowania. Dzięki temu użytkownicy mogą łatwo sprawdzić, czy akumulator jest naładowany do odpowiedniego poziomu.

6. Czujnik Ultradźwiękowy HC-SR04 do kontroli czy jest kubek



Rys 16. Czujnik Ultradźwiękowy HC-SR04

Czujnik ultradźwiękowy HC-SR04 jest kluczowym komponentem w naszym projekcie automatu do napojów, odpowiedzialnym za wykrywanie obecności kubka. Działa na zasadzie emisji fal ultradźwiękowych i pomiaru czasu ich odbicia od obiektu, co pozwala na precyzyjne określenie odległości. HC-SR04 składa się z nadajnika i odbiornika ultradźwięków. Nadajnik wysyła impulsy ultradźwiękowe o częstotliwości 40 kHz, które po napotkaniu kubka odbijają się i wracają do odbiornika. Czujnik mierzy czas od wysłania impulsu do jego powrotu, a następnie przelicza ten czas na odległość. Dokładność tego pomiaru wynosi kilka milimetrów, co jest wystarczające do naszych potrzeb. W automacie do napojów HC-SR04 jest umieszczony w miejscu, gdzie użytkownik umieszcza kubek.

6 Projekt obudowy

Obudowa została zaprojektowana tak, aby pomieścić wszystkie komponenty elektroniczne oraz akumulator. Uwzględniono również miejsca na panele słoneczne i wentylację ogniwo Peltiera.



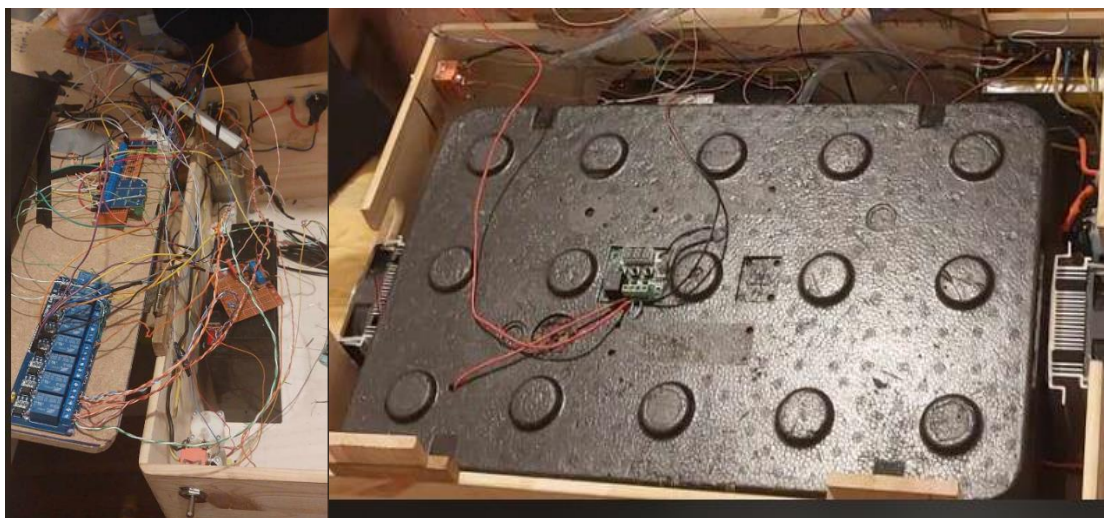
Rys 17. Obudowa otwarta i zamknięta



Rys 18. Lodówka otwarta, miejsce na kubek



Rys 19. Lodówka otwarta, testowanie



Rys 20. Początkowe łączenie modułów układów

Montaż układów

Prototyp został zmontowany zgodnie z projektami elektronicznymi i mechanicznymi. Zamontowano wszystkie komponenty i przeprowadzono pierwsze testy.

Testy i kalibracja

Lodówka udało się uzyskać temperaturę 8,8 stopnia



Rys 21. Test lodówki

- Ograniczenie prądowe test działania

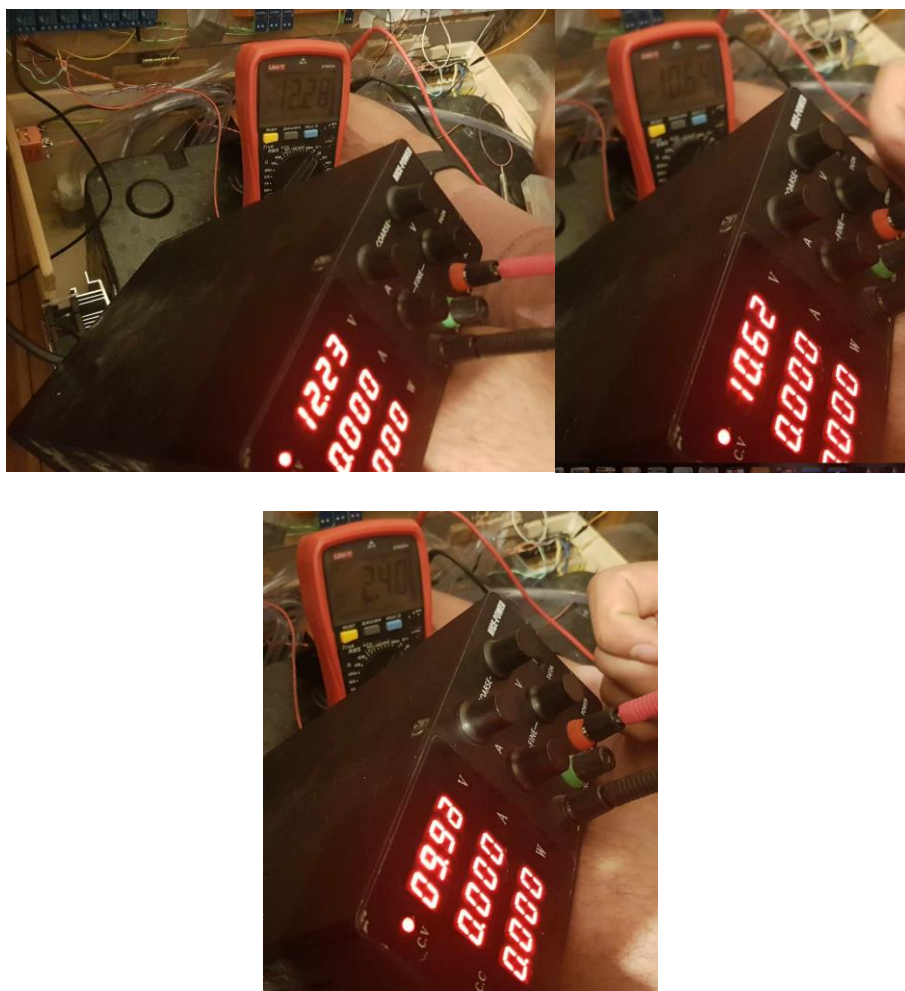
Napięcie

Prąd



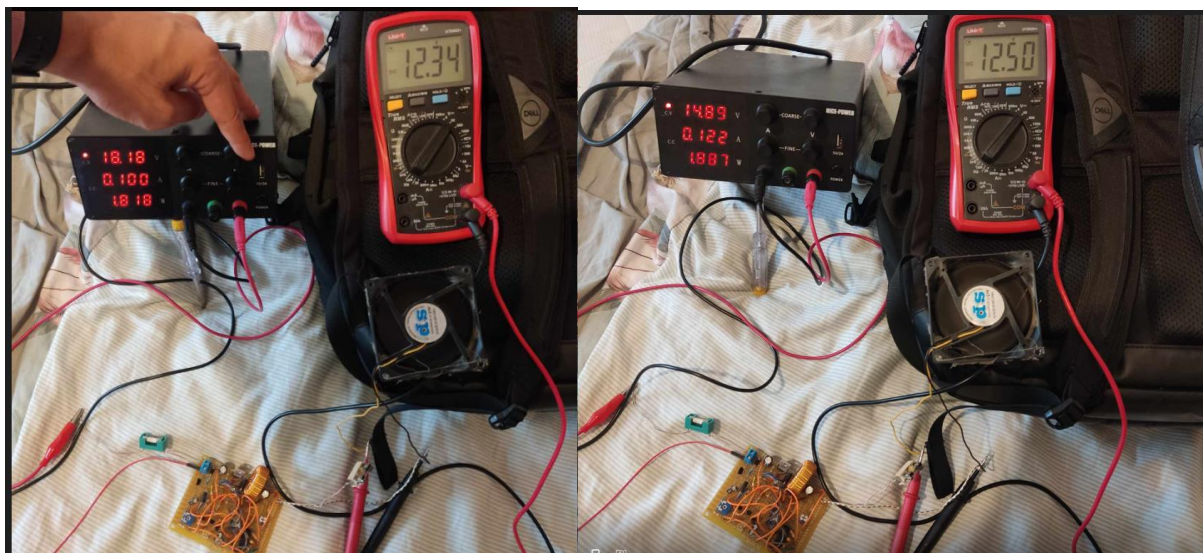
Rys 22. Test ograniczenia prądowego

- Testy układu zabezpieczającego przed rozładowaniem akumulatora.

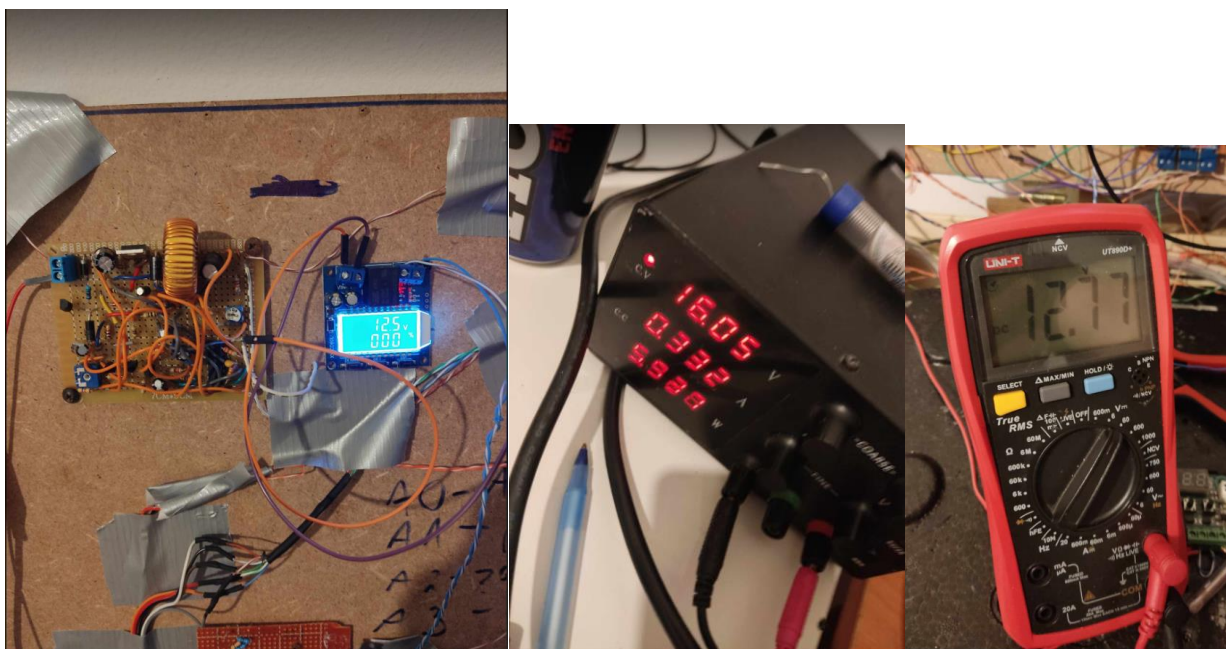


Rys 23. Test układu zabezpieczającego przed rozładowaniem akumulatora

- Testy układu MPPT

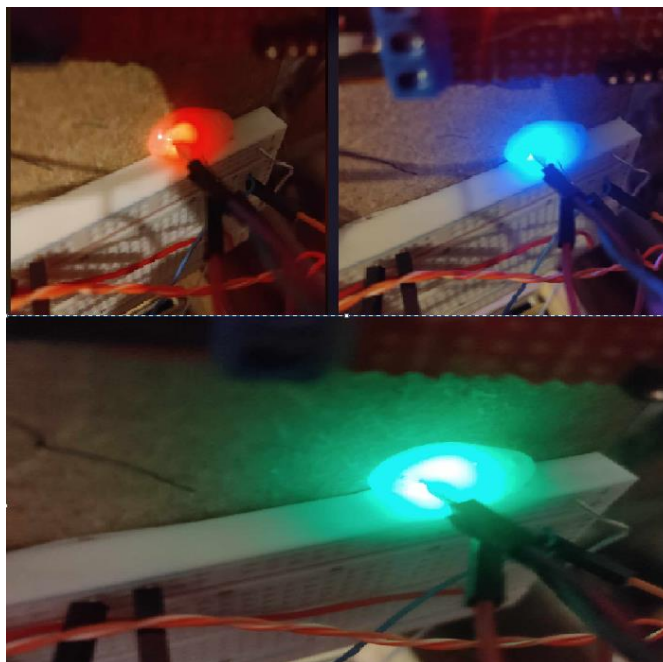


Napięcie wejściowe 16 V, za modułem MPPT 12,77 V



Rys 24. Test układu MPPT

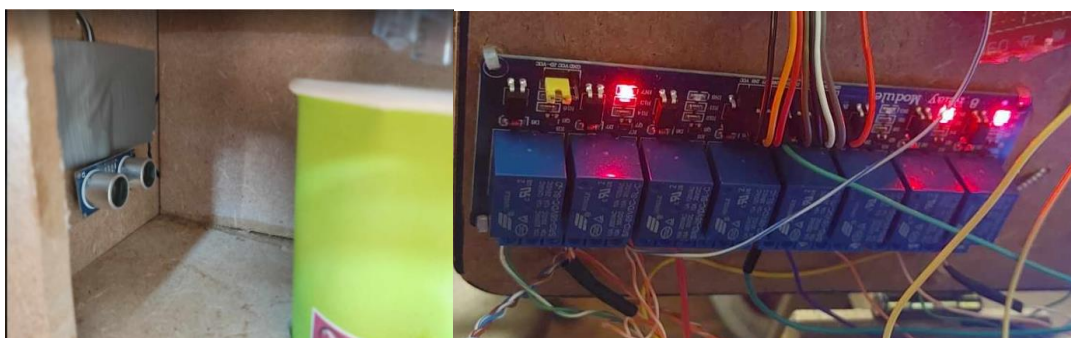
- Test układu wskaźnika diodowego przy różnych napięciach akumulatora



Rys 25. Test układu wskaźnika diodowego

- Test czujnika HC-SR04 (sprawdza czy jest kubek)

Test (kubek jest w automacie)



Test (kubek nie jest w automacie)



Rys 26. Czujnik HC-SR04

- Test pomiarów w aplikacji.

Wykonaliśmy pomiary dla kilku możliwości na Rys 26. Widać odczytaną wartość napięcia akumulatora (Uaku) oraz Up1 czyli napięcia panującego na pompkach prądu mierzy z dokładnością ± 150 mA dlatego nie udało się nam zrealizować założenia odnośnie tego czy w danym pojemniku jest napój, ponieważ różnica w poborze wynosi mniej więcej taką samą wartość co dokładność i nie jest możliwe zrealizowanie tego założenia przy tych niskobudżetowych modułach (ogólnie nie polecam), co do pozostałych napięć mierzone są poprawnie, jest również miejsce na przykładowy pomiar napięcia, więc nie dość że posiadamy automat do napojów, lodówkę to jeszcze prosty multimetr do pomiarów.

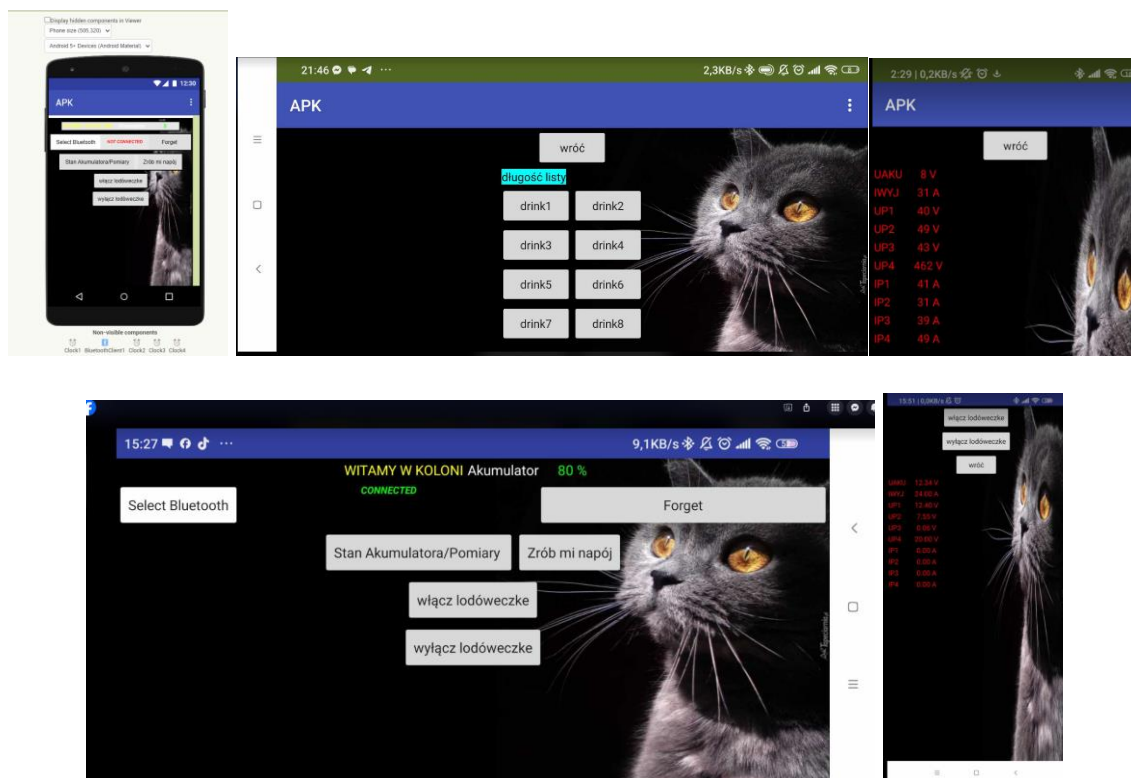
7 Aplikacja mobilna

Funkcjonalność aplikacji

Aplikacja mobilna umożliwia monitorowanie stanu naładowania akumulatora, kontrolę pracy pomp i ogniw Peltiera oraz odbieranie pomiarów napięć i prądów

Interfejs użytkownika

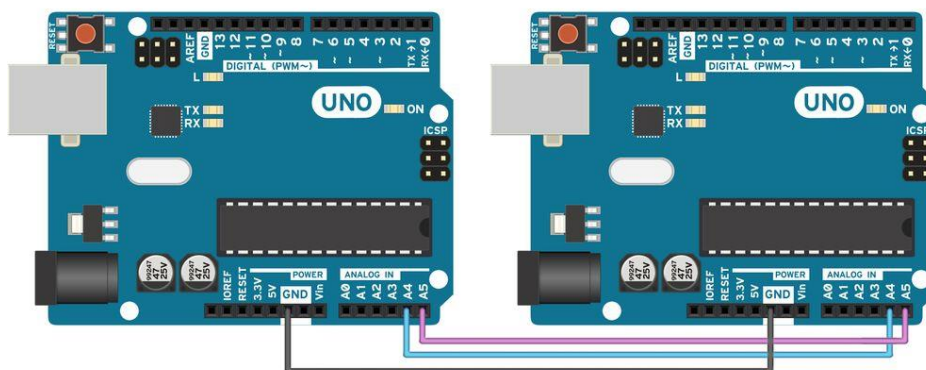
Interfejs użytkownika został zaprojektowany z myślą o prostocie i intuicyjności. Menu aplikacji zawiera opcje monitorowania i kontroli poszczególnych elementów systemu.



Rys 27. Interfejs użytkownika aplikacji

8. Komunikacja z układem

Komunikacja między aplikacją a układem odbywa się za pomocą modułów Bluetooth HC-05 oraz interfejsu I2C, co zapewnia niezawodność i szybki przesył danych.



Rys 28. Komunikacja I2C



Kod aplikaciji

```
when forget .Click  
do call BluetoothClient1 .Disconnect
```

```
when Screen1 .BackPressed  
do call BluetoothClient1 .Disconnect  
close application
```

```
when ListPicker1 .BeforePicking  
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```

```
when ListPicker1 .AfterPicking  
do if call BluetoothClient1 .Connect  
address ListPicker1 . Selection  
then set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```

```
when Clock1 .Timer  
do if BluetoothClient1 . IsConnected  
then set Result_of_Bluetooth . Text to " CONNECTED "  
set Result_of_Bluetooth . TextColor to   
set Clock3 . TimerAlwaysFires to true  
else set Result_of_Bluetooth . Text to " NOT CONNECTED "  
set Result_of_Bluetooth . TextColor to 
```

```

when Clock1.Timer
do
  if BluetoothClient1.IsConnected
  then
    set Result_of_Bluetooth.Text to "CONNECTED"
    set Result_of_Bluetooth.TextColor to green
    set Clock3.TimerAlwaysFires to true
  else
    set Result_of_Bluetooth.Text to "NOT CONNECTED"
    set Result_of_Bluetooth.TextColor to red
  end if
end do

```

```

when stan_aku.Click
do
  set HorizontalScrollArrangement1.Visible to true
  set VerticalScrollArrangement3.Visible to true
  set HorizontalArrangement5.Visible to false
  set HorizontalArrangement2.Visible to false
  set HorizontalArrangement1.Visible to false
end do

```

```

when zrob_napoj.Click
do
  set VerticalScrollArrangement4.Visible to true
  set HorizontalScrollArrangement1.Visible to false
  set HorizontalArrangement5.Visible to false
  set HorizontalArrangement1.Visible to false
  set HorizontalArrangement2.Visible to false
  set VerticalScrollArrangement5.Visible to true
end do

```

```

when Button1.Click
do
  set VerticalScrollArrangement4.Visible to false
  set HorizontalScrollArrangement1.Visible to false
  set VerticalScrollArrangement3.Visible to false
  set HorizontalArrangement5.Visible to true
  set HorizontalArrangement2.Visible to true
  set HorizontalArrangement1.Visible to true
end do

```

initialize global (list) to create empty list

initialize global (input) to "

```

when wroc_zrob_mi_napoj.Click
do
  set VerticalScrollArrangement4.Visible to false
  set HorizontalScrollArrangement1.Visible to false
  set VerticalScrollArrangement3.Visible to false
  set HorizontalArrangement5.Visible to true
  set HorizontalArrangement2.Visible to true
  set HorizontalArrangement1.Visible to true
  set VerticalScrollArrangement5.Visible to false
end do

```

```

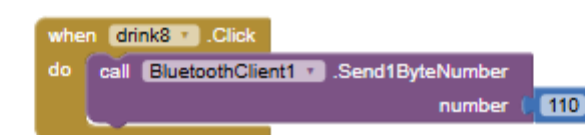
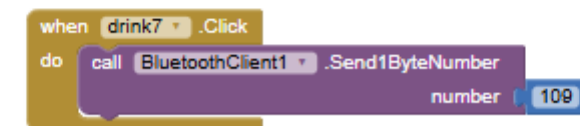
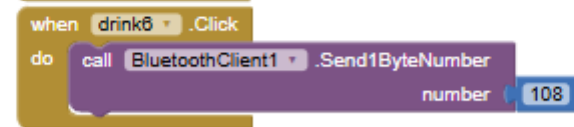
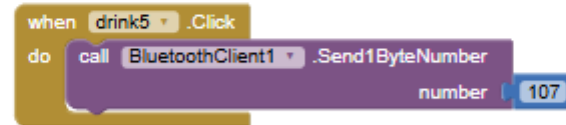
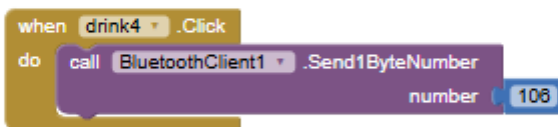
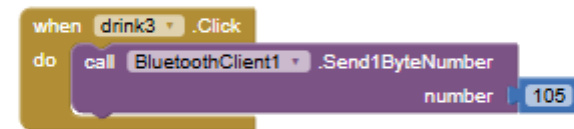
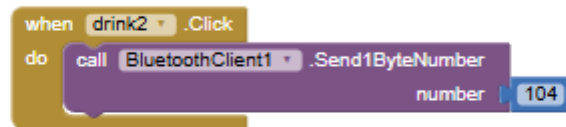
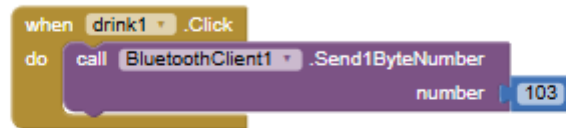
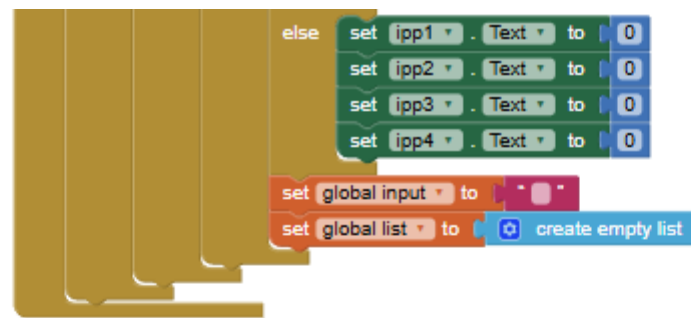
when Clock2.Timer
do
  if Clock2.TimerEnabled
  then
    set Clock2.TimerEnabled to false
  end if
end do

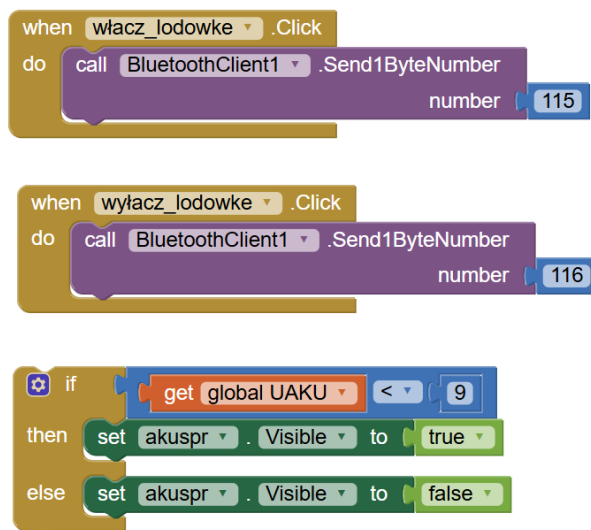
```

```

when Clock3.Timer
do
  if BluetoothClient1.IsConnected
  then
    if call BluetoothClient1.BytesAvailableToReceive > 0
    then
      set global input to call BluetoothClient1.ReceiveText
      numberOfBytes call BluetoothClient1.BytesAvailableToReceive
      set global list to split text get global input
      at " "
      if length of list get global list < 7
      then
        set test.Text to length of list list get global list
        set uaku2.Text to select list item list get global list
        index 1
        set lwyj2.Text to select list item list get global list
        index 2
        set upp1.Text to select list item list get global list
        index 3
        set upp2.Text to select list item list get global list
        index 4
        set upp3.Text to select list item list get global list
        index 5
        set upp4.Text to select list item list get global list
        index 6
        set Label2.Text to select list item list get global list
        index 7
      else
        if length of list list get global list > decimal 11
        then
          set lpp1.Text to select list item list get global list
          index 8
          set lpp2.Text to select list item list get global list
          index 9
          set lpp3.Text to select list item list get global list
          index 10
          set lpp4.Text to select list item list get global list
          index 11
        end if
      end if
    end if
  end if
end do

```



Rys 29. Kod aplikacji

Opis działania aplikacji stworzonej w MIT App Inventor

Aplikacja, którą pokazują dostarczone zrzuty ekranu, jest zaprojektowana do zarządzania i monitorowania systemu zasilania poprzez Bluetooth. Poniżej znajduje się szczegółowy opis działania poszczególnych części aplikacji.

Główne komponenty i funkcje aplikacji:

1. **BluetoothClient1:**
 - Komponent odpowiedzialny za nawiązanie i zarządzanie połączeniem Bluetooth z urządzeniem zewnętrznym (np. Arduino).
2. **ListPicker1:**
 - Umożliwia użytkownikowi wybór urządzenia Bluetooth do połączenia.
3. **Clock1, Clock2, Clock3, Clock4:**
 - Timery używane do wykonywania cyklicznych zadań, takich jak sprawdzanie połączenia Bluetooth oraz aktualizacja danych.
4. **Interfejs użytkownika:**
 - Aplikacja posiada różne elementy interfejsu użytkownika, takie jak przyciski, etykiety i układy przewijania, które umożliwiają monitorowanie stanu akumulatora oraz sterowanie systemem.

Działanie aplikacji:

1. **Zarządzanie połączeniem Bluetooth:**
 - **ListPicker1.BeforePicking:** Przed otwarciem ListPicker1, aplikacja pobiera listę dostępnych urządzeń Bluetooth.
 - **ListPicker1.AfterPicking:** Po wybraniu urządzenia z listy, aplikacja nawiązuje połączenie Bluetooth.
 - **Clock1.Timer:** Timer sprawdza co pewien czas czy połączenie Bluetooth jest aktywne. Jeśli połączenie jest aktywne, wyświetla komunikat "CONNECTED" na zielono, w przeciwnym razie "NOT CONNECTED" na czerwono.
 - **forget.Click:** Rozłącza połączenie Bluetooth, gdy użytkownik kliknie przycisk "Forget".
 - **Screen1.BackPressed:** Rozłącza połączenie Bluetooth i zamyka aplikację, gdy użytkownik naciśnie przycisk "Back".
2. **Aktualizacja danych i stan akumulatora:**
 - **Clock3.Timer:** Timer sprawdza, czy są dostępne nowe dane do odebrania przez Bluetooth. Jeśli tak, odbiera je, przetwarza i aktualizuje odpowiednie pola na ekranie, takie jak napięcie akumulatora, prąd i inne parametry.

- **stan_aku.Click:** Przy kliknięciu przycisku "Stan Akumulatora", aplikacja zmienia widok, aby pokazać szczegółowe informacje o stanie akumulatora.
- **Button1.Click:** Przy kliknięciu przycisku "Button1", aplikacja zmienia widok, aby wrócić do głównego ekranu.

3. Sterowanie napojami:

- **zrob_napoj.Click:** Przy kliknięciu przycisku "Zrób mi napój", aplikacja zmienia widok, aby umożliwić wybór napoju do przygotowania.
- **drinkX.Click:** Każdy przycisk odpowiadający różnym napojom (drink1, drink2, itd.) wysyła odpowiedni numer przez Bluetooth do Arduino, aby uruchomić odpowiednią pompkę do przygotowania napoju.

4. Warunkowe wyświetlanie elementów interfejsu:

- **if get global UAKU < 0 then:** Sprawdza, czy globalna zmienna UAKU (napięcie akumulatora) jest mniejsza niż 0. Jeśli tak, ustawia widoczność elementu "akuspr" na true (widoczny), w przeciwnym razie na false (niewidoczny).

11. Opis części programowej

9.1 Arduino Uno

Główny mikrokontroler Arduino Uno steruje zasilaniem, pompkami oraz pomiarami prądów, a także przesyłaniem oraz odbieraniem informacji z modułu Bluetooth. Szczegółowe działanie kodu zostało opisane poniżej.

```
1 // Dodatkowe biblioteki użyte w projekcie
2 #include <TimerOne.h>
3 #include <Timers.h>
4 #include <Wire.h>
5 #include "ACS712.h"
```

Rys 30. Opis kodu 1

Pierwsza część kodu odpowiada za dołączenie dodatkowych bibliotek wykorzystanych w projekcie.

```
7 // Deklaracja wykorzystanych pinów cyfrowych
8 #define POMPKA1 2 // pin Arduino połączony z przełącznikiem sterującym włączaniem pompki 1
9 #define POMPKA2 3 // pin Arduino połączony z przełącznikiem sterującym włączaniem pompki 2
10 #define POMPKA3 4 // pin Arduino połączony z przełącznikiem sterującym włączaniem pompki 3
11 #define POMPKA4 5 // pin Arduino połączony z przełącznikiem sterującym włączaniem pompki 4
12 #define Akumulator 7 // pin Arduino połączony z przełącznikiem sterującym podłączeniem akumulatora
13 #define Zasilacz 8 // pin Arduino połączony z przełącznikiem sterującym podłączeniem zasilacza
14 #define Trig 12 //pin Arduino połączony z pinem Trigger czujnika
15 #define Echo 11 //pin Arduino połączony z pinem Echo czujnika
16 Timers <8> akcja; // deklaracja 8 wątków
```

Rys 31. Opis kodu 2

Następnie przypisano piny cyfrowe wykorzystane w projekcie oraz zadeklarowano 8 przerwań czasowych.

```

18 // Deklaracja wykorzystanych pinów analogowych
19 ACS712 sensor(ACS712_20A, A1); // pin A1 mierzący pobór prądu ...
20 ACS712 sensor2(ACS712_20A, A2); // pin A2 mierzący pobór prądu ...
21 ACS712 sensor3(ACS712_20A, A3); // pin A3 mierzący pobór prądu ...
22 ACS712 sensor4(ACS712_20A, A6); // pin A4 mierzący pobór prądu ...
23 const int analogInputPin = A0; // pin, na którym mierzone jest napięcie na akumulatorze

```

Rys 32. Opis kodu 3

Następnie przypisano piny analogowe. Piny A1, A2, A3 oraz A6 przypisano z wykorzystaniem gotowej funkcji z biblioteki ACS712.

```

27 // Deklaracja oraz inicjalizacja użytych zmiennych
28 int CM; //odległość w cm z czujnika odległości
29 long CZAS; //długość powrotnego impulsu w uS czujnika odległości
30 int state=0; // zmienna przechowująca aktualny stan działania programu
31 int time=0; // zmienna przechowująca wartość czasu jaka upłynęła od startu lania napoju
32 int time_end1=0; // zmienna przechowująca czas po którym wyłączy się 1 pompka
33 int time_end2=0; // zmienna przechowująca czas po którym wyłączy się 2 pompka
34 int time_end3=0; // zmienna przechowująca czas po którym wyłączy się 3 pompka
35 int time_end4=0; // zmienna przechowująca czas po którym wyłączy się 4 pompka
36 // CHUJCHUJCHUJ //int time_test;
37 float nap; // zmienna przechowująca napięcie akumulatora
38 float Uaku=8.0; // zmienna przechowująca inne napięcie akumulatora
39 int Iwyj=24; // zmienna przechowująca wartość prądu wyjściowego
40 float Up1=33.0; // zmienna przechowująca napięcie na 1 pompce
41 int Up2=42; // zmienna przechowująca napięcie na 2 pompce
42 int Up3=36; // zmienna przechowująca napięcie na 3 pompce
43 int Up4=455; // zmienna przechowująca napięcie na 4 pompce
44 float Ip1=34; // zmienna przechowująca prąd pobierany przez 1 pompke
45 float Ip2=24; // zmienna przechowująca prąd pobierany przez 2 pompke
46 float Ip3=32; // zmienna przechowująca prąd pobierany przez 3 pompke
47 float Ip4=42; // zmienna przechowująca prąd pobierany przez 4 pompke
48 int sensorValue = 0; // zmienna przechowująca wartość odczytaną z pinu A0 (U akumulatora)
49 float napiecie=0.0; // zmienna przechowująca przetworzoną wartość napięcia akumulatora
50 int rozdzielczosc=1024; // zmienna przechowująca rozdzielczość ADC
51 float max=5.0; // Zmienna przechowująca maksymalne napięcie dla pomiaru

```

Rys 33. Opis kodu 4

Linie kodu od 27 do 51 były odpowiedzialne za deklaracje oraz inicjalizacje zmiennych użytych w projekcie. Szczegółowy opis każdej zmiennej przedstawia (Rys 33).

```

53 // Funkcja odczytująca dane przesyłane z 2giego Arduino przez I2C
54 void receiveEvent(int howMany)
55 {
56     while (Wire.available()) { // Sprawdzanie czy dane są przesyłane
57         float skala2=max/rozdzielczosc; // skala dla otrzymanych wartości
58         byte dolnyBajt1 = Wire.read(); // odczytanie 1 połowy 1 wartości
59         byte gornyBajt1 = Wire.read(); // odczytanie 2 połowy 1 wartości
60         int x1 = (gornyBajt1 << 8) | dolnyBajt1; // złączenie otrzymanych danych w 1 wartość typu int
61         Up1 = x1; // przypisanie wartości odczytanej do Up1
62         Up1 = Up1 *skala2*6; // Przeskalowanie wartości Up1 do rzeczywistej wartości napięcia
63
64
65         // To samo dla 2 wartości
66         byte dolnyBajt2 = Wire.read();
67         byte gornyBajt2 = Wire.read();
68         int x2 = (gornyBajt2 << 8) | dolnyBajt2;
69         Up2 = x2;
70         Up2 = Up2 *skala2*6;
71
72         // To samo dla 3 wartości
73         byte dolnyBajt3 = Wire.read();
74         byte gornyBajt3 = Wire.read();
75         int x3 = (gornyBajt3 << 8) | dolnyBajt3;
76         Up3 = x3;
77         Up3 = Up3 *skala2*6;
78
79         // To samo dla 4 wartości
80         byte dolnyBajt4 = Wire.read();
81         byte gornyBajt4 = Wire.read();
82         int x4 = (gornyBajt4 << 8) | dolnyBajt4;
83         Up4 = x4;
84         Up4 = Up4 *skala2*6;
85
86     }
87 }

```

Rys 34. Opis kodu 5

Funkcja receiveEvent() jest odpowiedzialna za komunikację z drugim mikrokontrolerem za pomocą I2C. Dane otrzymane od drugiego Arduino są przekazywane w 2 ośmiobitowych paczkach a następnie łączone w jedną szesnastobitową zmienną, która następnie zostaje przeskalowana na wartość napięcia. Procedura powtarzana jest dla reszty pompek.

```

90 void setup() {
91     // Kalibracja mierników ACS712
92     sensor.calibrate();
93     sensor2.calibrate();
94     sensor3.calibrate();
95     sensor4.calibrate();
96
97     // Konfiguracja pinów pracujących z czujnikiem odległości
98     pinMode(Trig, OUTPUT); // Trig z czujnika odległości
99     pinMode(Echo, INPUT); // Echo z czujnika odległości
100
101     // Konfiguracja pinów sterujących pompkami
102     pinMode(POMPKA1, OUTPUT);
103     pinMode(POMPKA2, OUTPUT);
104     pinMode(POMPKA3, OUTPUT);
105     pinMode(POMPKA4, OUTPUT);
106
107     // Konfiguracja pinów sterujących zasilaniem
108     pinMode(Akumulator, OUTPUT);
109     pinMode(Zasilacz, OUTPUT);
110
111     // Domyślne ustawienie pompki oraz zasilania na HIGH (wyłączenie)
112     digitalWrite(POMPKA1, HIGH);
113     digitalWrite(POMPKA2, HIGH);
114     digitalWrite(POMPKA3, HIGH);
115     digitalWrite(POMPKA4, HIGH);
116     digitalWrite(Akumulator, HIGH);
117     digitalWrite(Zasilacz, HIGH);

```

Rys 35. Opis kodu 6

W pierwszej części funkcji setup() zostają skalibrowane mierniki ACS712 oraz przypisane wartości pinom wyjściowym. Szczegóły przypisanych wartości znajdują się na (Rys 35).

```

119 // Konfiguracja komunikacji I2C
120 Wire.begin(4);
121 Wire.onReceive(receiveEvent);
122
123 // Rozpoczęcie komunikacji szeregowej z prędkością 9600 bps
124 Serial.begin(9600);
125
126 // Inicjalizacja wątków (przerwań czasowych co określony odcinek czasu)
127 akcja.attach(1, 1050, pomiar); // Wątek 1: wcześniej zdefiniowana funkcja o nazwie pomiar wywoływana co ok 1 s
128 akcja.attach(2, 33, pompki); // Wątek 2: wcześniej zdefiniowana funkcja o nazwie pompki, wywoływana co ok 33 ms
129
130 //kalibracja Timera 1
131 TCCR1A = 0;
132 TCCR1B = 0;
133 TCCR1B |= (0 << CS12) | (1 << CS11) | (0 << CS10); //prescaler 8 ~32ms (110 strona dokumentacji ATmegi 128p)
134 TIMSK1 |= (1 << TOIE1);
135
136
137 }

```

Rys

36. Opis kodu 7

W dalszej części funkcji setup() konfigurowane są komunikacje I2C oraz szeregową oraz Timer1, a także deklarowane są przerwania czasowe.

```

139 void loop() {
140
141     akcja.process(); // Uruchomienie wszystkich wątków (przerwań)
142
143     // Odbieranie poleceń z telefonu
144     if(Serial.available() > 0){ // Sprawdzanie czy przesyłana jest informacja przez port szeregowy (Bluetooth)
145         state = Serial.read(); // Zmiana wartości state na tą odczytaną z aplikacji
146     }
147 }

```

Rys 37. Opis kodu 8

W głównej pętli programu uruchamiane są wszystkie wątki oraz sprawdzane jest czy do mikrokontrolera przesyłane są jakieś dane przez port szeregowy. Jeżeli tak to state ustawiany jest na odczytaną wartość.

```

170 void pomiar()
171 {
172     Ip1 = sensor.getCurrentAC(); // Pobranie wartości prądu 1 pompki
173     delay(100);
174     Ip2 = sensor2.getCurrentAC(); // Pobranie wartości prądu 2 pompki
175     delay(100);
176     Ip3 = sensor3.getCurrentAC(); // Pobranie wartości prądu 3 pompki
177     delay(100);
178     Ip4 = sensor4.getCurrentAC(); // Pobranie wartości prądu 4 pompki
179
180     // Ignorowanie wartości poniżej 0.09 dla pompki 1
181     if (Ip1 < 0.09) {
182         Ip1 = 0;
183     }
184
185     // Ignorowanie wartości poniżej 0.09 dla pompki 2
186     if (Ip2 < 0.09) {
187         Ip2 = 0;
188     }
189
190     // Ignorowanie wartości poniżej 0.09 dla pompki 3
191     if (Ip3 < 0.09) {
192         Ip3 = 0;
193     }
194
195     // Ignorowanie wartości poniżej 0.09 dla pompki 4
196     if (Ip4 < 0.09) {
197         Ip4 = 0;
198     }

```

Rys 38. Opis kodu 9

Funkcja pomiar () (która jest wywoływana co ok. 1 s) rozpoczyna się od odczytania wartości prądów na pompkach oraz pominięcia wartości mniejszych niż ustalona wartość.

```

200 nap=nap_aku(); // Przypisanie wartości napięcia akumulatora do zmiennej nap
201
202 if (nap<8.00) // Sprawdzenie czy akumulator jest rozładowany
203 {
204     digitalWrite(Akumulator,HIGH); // Wyłączenie akumulatora
205     digitalWrite(Zasilacz,LOW); // Włączenie zasilacza
206 }
207 else
208 {
209     digitalWrite(Zasilacz,HIGH); // Wyłączenie zasilacza
210     digitalWrite(Akumulator,LOW); // Włączenie akumulatora
211 }
212
213 // Przypisanie do zmiennej procent procentowej wartości naładowania akumulatora
214 int procent=akunal(Uaku);

```

Rys 39. Opis kodu 10

Następnie odczytywana jest wartość napięcia na akumulatorze i w sytuacji, gdy napięcie na akumulatorze jest mniejsze niż 8V (akumulator jest rozładowany) następuje zmiana zasilania z akumulatora na zasilacz. Ponadto odczytywana jest procentowa wartość naładowania akumulatora.

```

216 // Przesłanie portem szeregowym (do modułu Bluetooth) informacji o napięciach oraz prądzie
217 Serial.print(Uaku);
218 Serial.print(" v");
219 Serial.print("|");
220 Serial.print(Iwyj);
221 Serial.print(" A");
222 Serial.print("|");
223 Serial.print(Up1);
224 Serial.print(" v");
225 Serial.print("|");
226 Serial.print(Up2);
227 Serial.print(" v");
228 Serial.print("|");
229 Serial.print(Up3);
230 Serial.print(" v");
231 Serial.print("|");
232 Serial.print(Up4);
233 Serial.print(" v");
234 Serial.print("|");
235 Serial.print(procent);
236 Serial.print(" %");
237 Serial.print("|");

```

Rys 40. Opis kodu 11

Otrzymane wartości napięć i prądu wyjściowego zostały przesłane przez port szeregowy do modułu Bluetooth, skąd dalej przesyłane są na telefon użytkownika.


```

253 if(state==103 or state==104 or state==105 or state==106 or state==107 or state== 108)
254 {
255     Serial.print(Ip1);
256     Serial.print(" A");
257     Serial.print("|");
258     Serial.print(Ip2);
259     Serial.print(" A");
260     Serial.print("|");
261     Serial.print(Ip3);
262     Serial.print(" A");
263     Serial.print("|");
264     Serial.print(Ip4);
265     Serial.print(" A");
266     Serial.print("|");
267 }

```

Rys 41. Opis kodu 12

Jeżeli w danym momencie jest robiony jakiś napój to dodatkowo wysłane zostaną informacje o prądach pobieranych przez poszczególne pompki. Na ich podstawie użytkownik może stwierdzić, że jakiś napój się kończy (mniejszy pobór prądu).

```

280 float nap_aku()
281 {
282     sensorValue = analogRead(analogInputPin); // Odczytanie wartości z miernika napięcia akumulatora
283     float skala=max/rozdzielczosc; // Skala
284     napiecie=(sensorValue*skala)*(1200 / 200); // Zamiana odczytanej wartości na napięcie
285     return napiecie;
286 }

```

Rys 42. Opis kodu 13

Funkcja nap_aku() zwraca wartość napięcia na akumulatorze.

```

288 // Funkcja zwracająca procentowe naładowanie akumulatora
289 float akunal(float napiecie)
290 {
291     if (napiecie>12.75)
292     {
293         return 100;
294     }
295     else if(napiecie<12.75 && napiecie>12.50)
296     {
297         return 90;
298     }
299     else if(napiecie<12.50 && napiecie>12.30)
300     {
301         return 80;
302     }
303
304     else if(napiecie<12.30 && napiecie>12.15)
305     {
306         return 70;
307     }
308     else if(napiecie<12.15 && napiecie>12.05)
309     {
310         return 60;
311     }

```

Rys 43. Opis kodu 42

Na (Rys 43.) pokazano fragment funkcji `akunal()`, która zwraca szacunkowy poziom naładowania w interwałach co 10%.

```
339 // Funkcja konfigurująca oraz odczytująca wartość z czujnika odległości
340 void pomiar_odleglosci ()
341 {
342     digitalWrite(Trig, LOW);           //ustawienie stanu wysokiego na 2 uS - impuls inicjalizujący - patrz dokumentacja
343     delayMicroseconds(2);
344     digitalWrite(Trig, HIGH);          //ustawienie stanu wysokiego na 10 uS - impuls inicjalizujący - patrz dokumentacja
345     delayMicroseconds(10);
346     digitalWrite(Trig, LOW);
347     digitalWrite(Echo, HIGH);
348     CZAS = pulseIn(Echo, HIGH);         // Odczytanie czasu odbitego impulsu w uS
349     CM = CZAS / 58;                     //szerokość odbitego impulsu w uS podzielone przez 58 to odleglosc w cm - patrz dokumentacja
350 }
```

Rys 44. Opis kodu 15

Funkcja `pomiar_odleglosci()` konfiguruje ultradźwiękowy czujnik odległości HC-SR04, dokonuje pomiaru czasu, po którym wraca wysłany impuls i zamienia tę wartość na odległość w cm.

```
355 // Funkcja przerwania wykonująca się co 33 ms sterująca pompkami
356 void pompki(){
357
358     pomiar_odleglosci(); // Sprawdzenie czy jest kubek
359
360     if(CM > 7) // Nie ma kubka
361     {
362         state = 69;
363     }
364
365     switch (state)
366     {
367     case 0: // Stan neutralny - nic się nie dzieje (czeka na zmianę state)
368         break;
369
370     case 103: // 1/3 - napoju z pompki 1 i 2/3 napoju z pompki 2 - konfiguracja
371         time = 0; // wyzerowanie czasu od rozpoczęcia lania napojów
372         time_end1 = 60; // ustawienie ilości przerwań, po których wyłączy się pompka 1
373         time_end2 = 120; // ustawienie ilości przerwań, po których wyłączy się pompka 2
374         state = 11; // Zmiana stanu na stan 11
375         digitalWrite(POMPKA1, LOW); // Włączenie pompki 1
376         digitalWrite(POMPKA2, LOW); // Włączenie pompki 2
377         break;
378     }
```

Rys 45. Opis kodu 16

Funkcja `pompki()`, która jest wywoływana co 33 ms odpowiada za działanie pompek oraz sterowanie zasilaniem. Na początek wywoływana jest funkcja `pomiar_odleglosci()`, która sprawdza, czy w uchwycie znajduje się kubek. Jeżeli nie to stan zmienia się na 69 (Rys 45.). Jeżeli nie jest wykonywany żaden napój to stan wynosi 0. Stan 103 to przykładowy stan przejściowy przed wykonaniem napoju. Na początek zerowany jest czas od rozpoczęcia lania napoju oraz ustawiane są czasy działania poszczególnych pompek. Następnie stan zmienia się na stan nalewania (Rys 45.) oraz włączane są pompki.

```

408 case 11: // Stan robienia napoju 1/3 - pompka 1, 2/3 - pompka 2
409 time++; // Inkrementacja czasu
410 if(time >= time_end1){ // Sprawdzenie czy minął czas po którym wyłączy się 1 pompka
411     digitalWrite(POMPKA1,HIGH); // Wyłączenie pompki 1
412     if(time >= time_end2){ // Sprawdzenie czy minął czas po którym wyłączy się 2 pompka
413         digitalWrite(POMPKA2,HIGH); // Wyłączenie pompki 2
414         time = 0; // Wyzerowanie czasu
415         time_end1=0; // Wyzerowanie czasu lania pompki 1
416         time_end2=0; // Wyzerowanie czasu lania pompki 2
417         state = 0; // Przejście do stanu 0
418     }
419 }
420 break;
421
422 case 69: // Stan gdy nie ma kubka
423     // Reset wszystkich czasów
424     time = 0;
425     time_test = 0;
426     state = 0;
427     // Wyłączenie wszystkich pompek
428     digitalWrite(POMPKA1,HIGH);
429     digitalWrite(POMPKA2,HIGH);
430     digitalWrite(POMPKA3,HIGH);
431     digitalWrite(POMPKA4,HIGH);
432 break;

```

Rys 46. Opis kodu 17

Na Rys 46. znajduje się działanie dla stanu 11 (przykładowy stan robienia napoju). Za każdym razem, gdy dochodzi do przerwania, iterowana jest zmienna time aż do osiągnięcia docelowego czasu lania napoju. Wtedy wyłączana jest pompka, dla której ów czas był wyznaczony. Kiedy wszystkie pomki zostaną wyłączone, wszystkie czasy zostają wyzerowane a state ustawiony na 0. Stan 69 oznacza sytuację, gdy czujnik nie wykrył kubka. W tej sytuacji czasy są zerowane a wszystkie pomki wyłączane.

9.2 Nano

Mikrokontroler Arduino Nano służy do odczytywania napięć na akumulatorze, zasilaczu oraz pompkach, przeskalowanie otrzymanych danych oraz przesyłanie ich na pierwszy mikrokontroler. Ponadto steruje on diodą LED, która informuje o stopniu naładowania akumulatora. Komunikacja między mikrokontrolerami odbywa się za pomocą interfejsu I2C.

```

1 // Piny, na które zbierają dane o napięciu
2 const int analogInputPin = A0;//napięcie akumulatora
3 const int analogInputPin1 = A1;//napięcie pompek
4 const int analogInputPin2 = A2;//napięcie na zasilaczu
5
6 // Piny odpowiedzialne za sterowanie diodą RGB
7 #define NIEBIESKA 3
8 #define ZIELONA 5
9 #define CZERWONA 6

```

Rys 47. Opis kodu 1

Pierwsza część kodu odpowiedzialna jest za przypisanie analogowych pinów odpowiedzialnych za pobór wartości napięć na akumulatorze, pompkach i zasilaczu, oraz cyfrowych pinów PWM odpowiedzialnych za sterowanie diodą RGB.

```
11 #include <Wire.h> // biblioteka pomagająca w przesyłaniu danych przez I2C
12 #include <ACS712.h> // biblioteka wspierająca mierniki ACS712
```

Rys 48. Opis kodu 2

W dalszej części kodu zawarto dodatkowe biblioteki wykorzystane w kodzie.

```
14 // Zmienna przechowująca odczytane napięcie
15 int sensorValue = 0;
16 int sensorValue2 = 0;
17 float napiecie = 0.00;
18 float max = 5.00;
19 int rozdzielczosc = 1023;
20 String wartosc = "100%";
21
22 char buffer[10]; // w bufferze będą przechowywane dane przesyłane przez I2C
23
24 float voltagePerMilliamp = 0.001;
```

Rys 49. Opis kodu 3

Następnie w kodzie zadeklarowano oraz zainicjalizowano wartości zmiennych użytych w dalszej części programu. Na szczególną uwagę zasługuje zmienna buffer, której zadaniem jest przechowywanie danych, które będą przesyłane do głównego mikrokontrolera.

```

27 void oo() { // funkcja pobierające dane o napięciu odczytanym przez ACS712 oraz wysyłająca te dane do Arduino UNO
28 // komendy odczytujące napięcia i zapisujące je do zmiennych
29 int x = analogRead(A1); // napięcie na pompkach
30 delay(100);
31 int x2 = analogRead(A2); // napięcie zasilacza
32 delay(100);
33 int x3 = analogRead(A3); // miernik
34 delay(100);
35
36 // Pobrane dane są zbyt duże aby móc je przesłać przez I2C dlatego dzielimy je na 2 mniejsze zmienne
37 byte gora1 = (byte)(x & 0xFF);
38 byte dol1 = (byte)((x >> 8) & 0xFF);
39
40 byte gora2 = (byte)(x2 & 0xFF);
41 byte dol2 = (byte)((x2 >> 8) & 0xFF);
42
43 byte gora3 = (byte)(x3 & 0xFF);
44 byte dol3 = (byte)((x3 >> 8) & 0xFF);
45
46 // łączenie się z Arduino UNO oraz przesłanie przez I2C zmierzonych danych
47 Wire.beginTransmission(4); // Rozpoczęcie transmisji
48 // Przesyłanie danych
49 Wire.write(gora1);
50 Wire.write(dol1);
51 Wire.write(gora2);
52 Wire.write(dol2);
53 Wire.write(gora3);
54 Wire.write(dol3);
55 Wire.endTransmission(4); // Zakończenie transmisji
56
57 delay(200);
58 }

```

Rys 50. Opis kodu 4

Funkcja oo() ma za zadanie pobrać dane z mierników napięcia oraz przesłać ich do głównego mikrokontrolera. Linie kodu od 29 do 34 pobierają wartość z poszczególnych pinów analogowych oraz zapisują je do wcześniej zainicjalizowanych zmiennych. Zmienne te są typu int, a więc są zbyt duże, aby dało się je przesłać w jednym pakiecie danych przez I2C. W tym każda z otrzymanych zmiennych jest dzielona na dwie mniejsze 8 bitowe zmienne oraz przesyłana do pierwszego Arduino. Za podział zmiennych na mniejsze odpowiadają linie kodu od 37 do 44, natomiast za ich transmisję linie kodu od 47 do 55.

```

63 void setup() {
64 // Inicjalizacja komunikacji szeregowej z prędkością 9600 bps
65 Wire.begin();
66 Serial.begin(9600);
67 }

```

Rys 51. Opis kodu 5

W funkcji setup() inicjalizowana jest komunikacja szeregową z prędkością bitową 9600 bps.

```

void loop() {
  // Odczytanie wartości napięcia na wejściu analogowym
  oo();

  // Odczytanie napięcia na akumulatorze
  sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  float skala = max / rozdzielczosc; // obliczenie skali

  // Wyświetlenie odczytanej wartości na monitorze szeregowym
  Serial.println("Odczytane napięcie: ");
  Serial.println(sensorValue * skala);
  Serial.println("Odczytane napięcie poprawnego: ");

  Serial.println(sensorValue * skala); //*(1200)/200);
  napiecie = (sensorValue * skala) * (1200 / 200); // Przeskalowanie otrzymanego wyniku
}

```

Rys 52. Opis kodu 6

W dalszej części opisu omówiona zostanie funkcja loop(). Na początku każdej pętli wywoływana jest poprzednio opisana funkcja oo(). Następnie odczytywana jest wartość z analogowego pinu A0, która następnie zostaje przeskalowana oraz zamieniona na wartość napięcia (przemnożona przez odwrotność dzielnika napięcia podłączonego do miernika). W ten sposób otrzymujemy wartość napięcia na akumulatorze.

```

93   analogWrite(ZIELONA, 0);
94   analogWrite(CZERWONA, 255);
95   analogWrite(NIEBIESKA, 255);
96   Serial.println("bateria :");
97   wartosc = "100%";
98   Serial.print(wartosc);
99   } else if (napiecie < 12.75 && napiecie > 12.50) {
100   analogWrite(ZIELONA, 50);
101   analogWrite(CZERWONA, 255);
102   analogWrite(NIEBIESKA, 255);
103   Serial.print("bateria :");
104   wartosc = "90%";
105   Serial.println(wartosc);
106   } else if (napiecie < 12.50 && napiecie > 12.30) {
107   analogWrite(ZIELONA, 100);
108   analogWrite(CZERWONA, 255);
109   analogWrite(NIEBIESKA, 255);
110   Serial.print("bateria :");
111   wartosc = "80%";
112   Serial.print(wartosc);
113   } else if (napiecie < 12.30 && napiecie > 12.15) {
114   analogWrite(ZIELONA, 140);

```

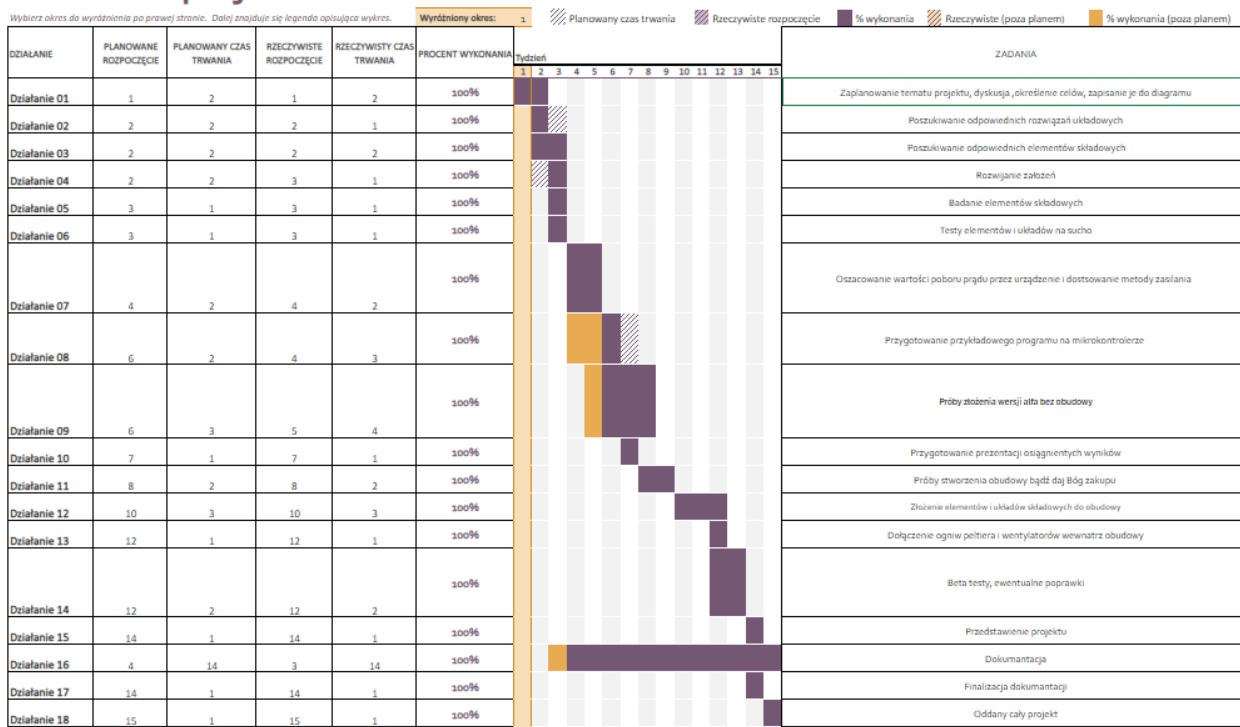
Rys 53. Opis kodu 7

Ostatnią częścią pętli jest wysterowanie diodą RGB w zależności od tego, jakie jest naładowanie akumulatora. Rys X. przedstawia fragment kodu odpowiedzialny za sterowanie diodą dla wartości naładowania akumulatora od 80% do 100%. Ponadto wartość naładowania akumulatora jest wykorzystywana w głównym mikrokontrolerze do przełączania zasilania oraz wyłączania chłodzenia, gdy wartość napięcia na akumulatorze spadnie poniżej wartości krytycznej.

10. Wykres Gantta

Podczas początkowych zajęć z projektu zespołowego wykonano wykres Gantta planowanych czynności projektowych oraz wykończeniowych. Udało się ukończyć całość projektu przed planowaną datą oddania go oraz udało się wykonać większość czynności w planowanym terminie. Drobny wyjątek stanowi opracowanie kodu na mikrokontrolerze, co było spowodowane implementacją dodatkowych funkcji do projektu.

Terminarz projektu



Rys 54. Wykres Gantta

11. Status wykonania założeń projektowych

Wykonane założenia projektowe:

- Sterowanie przez mikrokontroler Arduino – Zgodnie z założeniami projekt sterowany jest za pomocą mikrokontrolerów Arduino. W projekcie wykorzystano dwa Arduina Nano.
- Możliwość automatycznego robienia napojów z max 4 innych napojów - W aplikacji mobilnej można wybrać jeden z wielu napojów, które tworzone są poprzez połączenie do 4 różnych napojów.
- Czujnik czy umieszczono kubek – Zastosowano ultradźwiękowy czujnik odległości HC-SR04, który w razie potrzeby przerywa robienie napojów.
- Zasilanie sieciowe lub bateryjne – W projekcie zastosowano 2 różne rodzaje zasilania – sieciowe oraz akumulatorowe.

Niewykonane założenia projektowe:

- System monitorujący stan uzupełnienia napojów - System miał się opierać na informacji o aktualnie pobieranym prądzie przez pompki, który różnił się w zależności od tego czy dostępny był napój. Niestety zakupione mierniki prądu nie były wystarczająco dokładne, aby wprowadzić to rozwiązanie.

- Ręczne robienie napojów - Uznano, że lepiej będzie, jeżeli będzie można wykonywać z góry ustalone napoje.

Dodatkowo dodane funkcje:

- Aplikacja mobilna do sterowania oraz monitorowania urządzenia - Zaprojektowano aplikację na telefon łączącą się z urządzeniem za pomocą komunikacji Bluetooth. Aplikacja umożliwia wybór napoju do przygotowania oraz możliwość monitorowania prądów oraz napięć w kluczowych fragmentach urządzenia.
- System chłodzenia napojów z regulacją termostatem – Stworzono komorę chłodzącą wykorzystującą ogniwa Peltiera oraz wentylatory, która sterowana jest z użyciem termostatu.
- Zasilanie akumulatorem kwasowym – W urządzeniu zastosowano akumulator kwasowo ołowiowy o napięciu 12 V oraz pojemności 20 Ah.
- Możliwość przełączania między zasilaniem sieciowym a akumulatorowym – Zaimplementowano rozwiązanie, że w momencie rozładowania akumulatora urządzenie automatycznie przełącza się na zasilanie sieciowe.
- System ładowania akumulatora – Akumulator może być ładowany z 2 źródeł: z zasilacza sieciowego lub ze źródła zewnętrznego (np. panelu fotowoltaicznego) poprzez układ MPPT.
- System monitorujący stan naładowania akumulatora – Napięcie na akumulatorze jest stale monitorowane, a jego wartość wysyłana do aplikacji mobilnej oraz wyświetlana za pomocą diody RGB.
- Obudowa zapewniająca mobilność urządzenia - Urządzenie umieszczono w drewnianej skrzynce na kółkach, co ułatwia transport urządzenia.
- System monitorujący napięcia - Napięcia na pompkach, akumulatorze oraz zasilaczu mogą być stale monitorowane w aplikacji mobilnej.
- System monitorujący prąd pobierany przez poszczególne pompki – Zastosowano 4 mierniki ACS712. Niestety uzyskane pomiary obarczone są dość dużym błędem.

12. Ryzyka oraz przebieg prac

RYZYKO	PROCENT RYZYKA	Opóźnienie	Rozwiązanie
Brak ryzyka	0%	0 tygodni	Nie ma ryzyka nie ma rozwiązania.
Wybór nieodpowiednich rozwiązań może prowadzić do marnowania czasu i zasobów na implementację nieefektywnych lub niepraktycznych rozwiązań.	20%	1 tydzień	Należy przeprowadzić staranne badania i analizę dostępnych opcji, skupiając się na ich efektywności, praktyczności i zgodności z wymaganiami projektu. Warto także wykorzystać opinie społeczności technologicznej, aby uzyskać wsparcie i porady dotyczące najlepszych praktyk i rozwiązań.
Błędne założenia mogą prowadzić do nieprawidłowych wyników projektu. Dlatego będziemy regularnie monitorować i aktualizować założenia w miarę postępów projektu, aby uniknąć ewentualnych problemów.	15%	1 tydzień	Aby zminimalizować ryzyko błędnych założeń i nieprawidłowych wyników projektu, warto przeprowadzać regularne spotkania zespołu projektowego, podczas których będą omawiane założenia i postępy pracy. Dodatkowo, należy zachęcać członków zespołu do aktywnego zgłaszania wątpliwości i sugerowania poprawek.
Podczas oszacowania wartości poboru prądu przez urządzenie i dostosowania metody zasilania mogą wystąpić problemy związane z błędnym oszacowaniem poboru prądu.	10%	1 tydzień	Aby zminimalizować ryzyko błędnego oszacowania poboru prądu, należy przeprowadzić dokładne badania i testy, wykorzystując rzeczywiste dane oraz symulacje.
Podczas przygotowywania programu na mikrokontrolerze mogą wystąpić problemy związane z błędami w kodzie, niekompatybilnością z używanym sprzętem lub nieprzewidywanymi ograniczeniami zasobowymi.	5%	0 tygodni	Aby zminimalizować ryzyko wystąpienia problemów związanych z błędami w kodzie, warto przeprowadzać testy i debugowanie kodu, korzystać z dokumentacji sprzętu oraz zawsze uwzględnić specyfikację i ograniczenia mikrokontrolera podczas pisania programu.
Próby złożenia wersji alfa bez obudowy mogą napotkać na problemy związane z niewłaściwym chłodzeniem, ryzykiem uszkodzenia elementów elektronicznych oraz potencjalnym niebezpieczeństwem dla użytkownika związanym z brakiem fizycznej ochrony.	50%	1 tydzień	Ponadto, należy ostrożnie manipulować elementami elektronicznymi, aby uniknąć przypadkowych uszkodzeń podczas testów, oraz zapewnić środki bezpieczeństwa dla użytkowników, takie jak izolacja elementów, aby ograniczyć potencjalne niebezpieczeństwo związane z brakiem fizycznej ochrony.
Brak ryzyka	0%	0 tygodni	Nie ma ryzyka nie ma rozwiązania.
Ktoś może źle zmierzyć przez co urządzenie nie zmieści się do obudowy	12%	0 tygodni	Aby uniknąć tego problemu, należy wykonać dokładne pomiary wszystkich elementów oraz przestrzeni wewnątrz obudowy, z uwzględnieniem wszelkich ograniczeń i tolerancji.
Brak testów lub nieprawidłowe testowanie może prowadzić do pominięcia problemów, które mogą pojawić się w późniejszych etapach projektu. Zadbamy o kompleksowe testowanie wszystkich elementów i układów przed przejściem do następnych faz projektu.	27%	1 tydzień	Aby zmniejszyć ryzyko braku testów lub nieprawidłowego testowania, należy opracować kompleksowy plan testów, który uwzględni wszystkie możliwe scenariusze i przypadki użycia.
Brak ryzyka	0%	0 tygodni	Nie ma ryzyka nie ma rozwiązania.
Ocena mniejsza niż 3,0	100%	1 rok :{(Warunek :{(

Rys 55. Ryzyka oraz rozwiązania rozpoznane na początku projektu

Podczas trwania projektu napotkaliśmy następujące problemy:

- Problemy z komunikacją w zespole - Zdarzały się sytuacje, w których brakowało zgodności co do kierunku, w którym zmierza projekt oraz problemy z dogadaniem terminów spotkań.
- Nierówne zaangażowanie w projekt – Nie wszyscy uczestnicy wykazali się równomiernym zaangażowaniem w projekt.
- Wzrost kosztów projektu – Wraz z implementacją nowych funkcjonalności wzrosły koszty projektu.
- Uszkodzenie dużej części elementów podczas jednego ze spotkań – Podczas jednego ze spotkań doszło do fatalnej pomyłki podczas podłączania czujnika odległości, przez co uszkodzona została spora część projektu.
- Niska jakość zakupionych elementów - Wiele z zakupionych elementów wykazało się niską jakością. Zakupione mierniki prądu cechowała duża niedokładność pomiarów a jeden z zakupionych mikrokontrolerów miał zepsutą sekcję zasilania.

13. Podsumowanie

Projekt systemu zarządzania akumulatorem (BMS) oraz automatu do napojów został zrealizowany zgodnie z założeniami, zapewniając wysoką wydajność i niezawodność. System BMS składa się z kluczowych komponentów takich jak układ MPPT, czujniki prądu ACS7200, moduł Bluetooth HC-05, moduł HC-SR04 (sprawdza czy jest kubek) oraz układ zabezpieczający przed nadmiernym rozładowaniem i przeładowaniem akumulatora. Dodatkowo, automat do napojów został wyposażony w funkcje umożliwiające zasilanie z sieci lub akumulatora, a także monitorowanie i sterowanie za pomocą aplikacji mobilnej automat również autonomicznie steruje pracą lodówki ma ją rozłączyć przy 30 % naładowania akumulatora, żeby można była ciągle robić napoje z akumulatora, bez wykorzystania lodówki.

Główne osiągnięcia projektu:

1. **Efektywne ładowanie akumulatora:**
 - Zastosowanie układu MPPT (Maximum Power Point Tracking) pozwala na optymalne wykorzystanie energii z paneli słonecznych, maksymalizując efektywność ładowania akumulatora.
2. **Zabezpieczenie akumulatora:**
 - System zawiera mechanizmy ochrony przed nadmiernym rozładowaniem i przeładowaniem akumulatora. Transystor MOSFET IRF9540 w połączeniu z diodą Zenera TL431 zapewnia, że akumulator jest odłączany przy napięciu poniżej 9.8V, chroniąc go przed uszkodzeniem.
3. **Kontrola prądu ładowania:**
 - Układ ograniczenia prądowego wykorzystujący wzmacniacz operacyjny LM358 oraz tranzystor darlingtona BDX33C stabilizuje prąd ładowania na poziomie 3A, co zapewnia bezpieczny i efektywny proces ładowania.
4. **Monitorowanie i komunikacja:**
 - Aplikacja mobilna, stworzona w MIT App Inventor, umożliwia monitorowanie stanu naładowania akumulatora oraz kontrolę pracy systemu chłodzenia i pomp. Moduł Bluetooth HC-05 zapewnia niezawodną komunikację między aplikacją a systemem.
5. **Stabilizacja napięcia:**
 - Układ stabilizatora napięcia LM7805 dostarcza stabilne napięcie 5V do zasilania modułów pomiarowych i komunikacyjnych, zapewniając stabilność i niezawodność całego systemu.
6. **Kontrola obecności kubka**

Układ HC-SR04 sprawdza czy jest obecny kubek, jeżeli nie ma nie wlewać napoju do momentu włożenia kubka.
7. **Kontrola poziomu naładowania akumulatora**

Automat ma wyłączyć lodówkę, jeżeli pojemność spadnie poniżej 30 %.
8. **Możliwość włączania i wyłączania lodówki od strony aplikacji**

Aplikacja umożliwia włączenie i wyłączenie lodówki za pomocą przycisków.
9. **Mobilność dzięki zintegrowanym kołom i zasilaniu akumulatorowym**

W pełni przenośne rozwiązanie dzięki zamontowanym kółkom i akumulatorowi.

Rekomendacje

W przyszłych iteracjach projektu warto rozważyć:

☐ Optymalizacja zużycia energii:

- W przyszłych iteracjach projektu warto rozważyć optymalizację zużycia energii, aby zwiększyć efektywność systemu.

☐ Ulepszenie interfejsu aplikacji mobilnej:

- Warto pracować nad ulepszeniem interfejsu aplikacji mobilnej dla lepszego komfortu użytkownika.

☐ Możliwość dodawania napojów od strony aplikacji:

- Warto popracować nad ustawianiem czasów lania danych napojów od strony aplikacji

☐ Ulepszenie konstrukcji obudowy w celu łatwiejszego jej przemieszczania i odporności na warunki atmosferyczne:

- Zamknąć obudowę, żeby była odporna na np. deszcz.

☐ Zwiększenie pojemności akumulatora:

- Wprowadzenie akumulatorów o większej pojemności pozwoli na dłuższy czas pracy urządzenia bez konieczności częstego ładowania.

Wnioski

Projekt automatu do napojów osiągnął zamierzone cele, dostarczając kompleksowe rozwiązanie do zarządzania energią w automacie do napojów. Integracja technologii elektronicznych i mechanicznych oraz możliwość monitorowania i sterowania za pomocą aplikacji mobilnej znacząco podniosły funkcjonalność i wygodę użytkownika urządzenia.

Projektując automat, udało się wprowadzić szereg innowacyjnych rozwiązań, które przyczyniły się do zwiększenia jego efektywności operacyjnej. Zastosowanie układu MPPT (Maximum Power Point Tracking) pozwala na maksymalne wykorzystanie energii słonecznej, co jest szczególnie ważne w kontekście zrównoważonego rozwoju i oszczędności energii. Dodatkowo, mechanizmy ochrony przed nadmiernym rozładowaniem i przeładowaniem akumulatora zapewniają długą żywotność i niezawodność urządzenia.

Kontrola prądu ładowania, stabilizacja napięcia oraz monitorowanie stanu naładowania akumulatora poprzez aplikację mobilną umożliwiają użytkownikom łatwe zarządzanie urządzeniem i natychmiastową reakcję na ewentualne problemy. Moduł Bluetooth HC-05 zapewnia płynne i niezawodne połączenie między automatem a aplikacją, co umożliwia zdalne sterowanie i monitorowanie urządzenia.

Automat do napojów, dzięki zintegrowanym kołom i zasilaniu akumulatorowemu, charakteryzuje się wysoką mobilnością, co umożliwia jego łatwe przemieszczanie w różne miejsca w zależności od potrzeb użytkowników. Mobilny design jest istotnym atutem, szczególnie w miejscach o zmiennym natężeniu ruchu, takich jak festiwale, imprezy masowe czy targi.

Dodatkowe funkcje, takie jak system chłodzenia napojów oraz automatyczne monitorowanie i uzupełnianie zapasów, znacząco zwiększają komfort użytkowania automatu. Możliwość płatności bezgotówkowych oraz zdalne sterowanie i monitorowanie przez Internet to kolejne udogodnienia, które przyczyniają się do zwiększenia satysfakcji użytkowników.

W przyszłości warto rozważyć dalszą optymalizację energochłonności urządzenia, ulepszenie interfejsu aplikacji mobilnej oraz dodanie nowych funkcji, takich jak automatyczne mieszanie napojów czy zaawansowane monitorowanie parametrów środowiskowych. Wprowadzenie takich usprawnień może dodatkowo zwiększyć atrakcyjność automatu oraz jego konkurencyjność na rynku.

Podsumowując, projekt automatu do napojów nie tylko spełnił, ale i przewyższył oczekiwania, dostarczając innowacyjne i efektywne rozwiązanie, które jest łatwe w obsłudze, mobilne i dostosowane do współczesnych wymagań użytkowników.