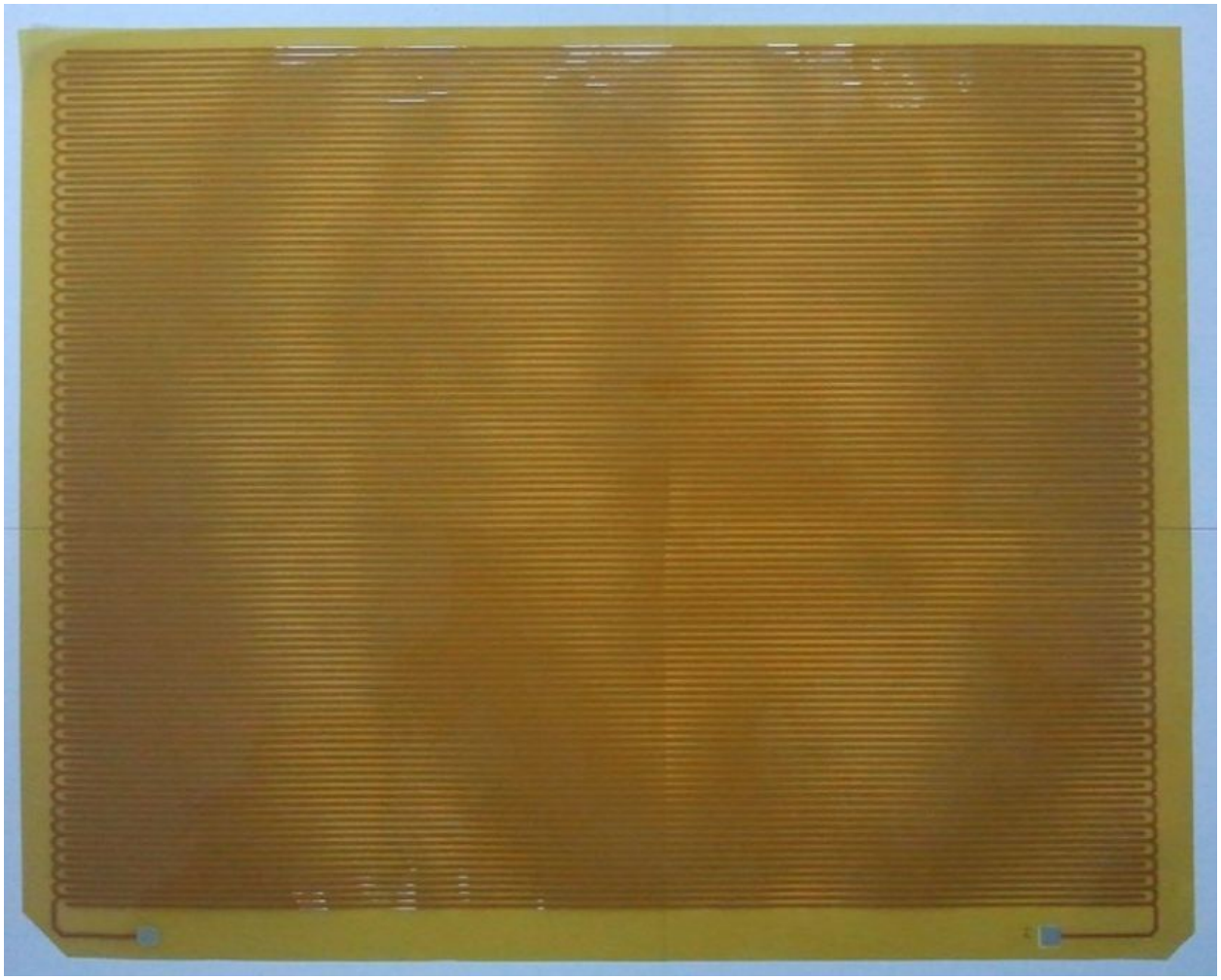
# Projekt Profilowy Robotyka 2

# Urządzenie do generowania i kontroli wysokich napięć na potrzeby robotyki miękkiej

Opiekun projektu: dr. inż. Igor Żubrucki
Michał Szczęsny 233845
Albert Sawiński 233839
Michał Witczewski 233851

## 1. Cel projektu

Celem niniejszego projektu jest zbadanie możliwości zasilania elastycznych ogniw elektrodowych stosowanych w robotyce miękkiej przy wykorzystaniu wysokich napięć. W tym celu zakupiona została przetwornica wysokiego napięcia, której budowa została zbadana, na jej podstawie przeprowadzone zostały symulacje, a końcowo stworzony został układ elektroniczny, którego zadaniem było zabezpieczenie przetwornicy oraz kontrola napięcia wyjściowego podawanego na przetwornicę.



Rys. 1.1 - Elastyczny panel elektrod (interdigital electrodes)

## 2. Przetwornica wysokiego napięcia

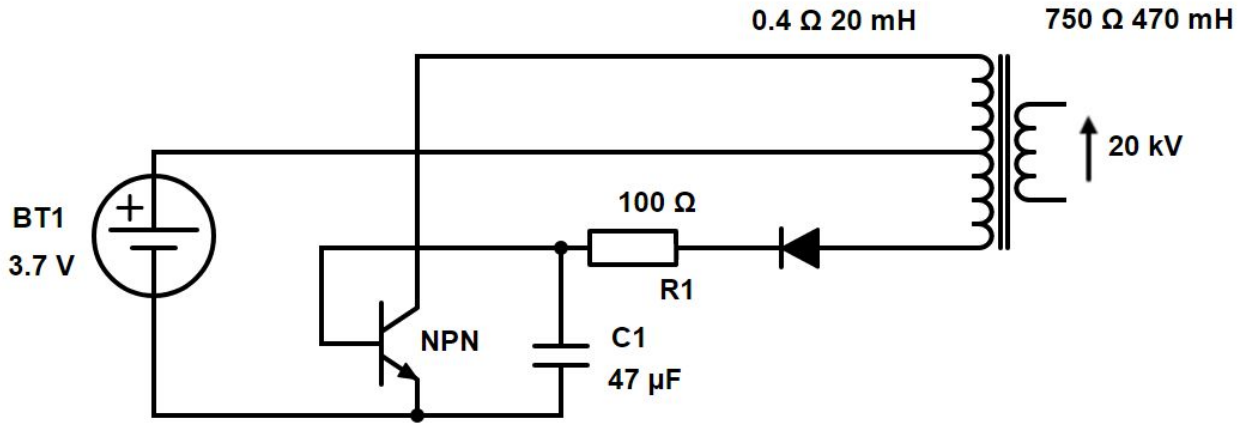
Zakupiona przetwornica charakteryzuje się następującymi parametrami:

Tab. 2.1 - Parametry przetwornicy wysokiego napięcia podane przez sprzedawcę

| | |
|---|---|
| Napięcie zasilania | 3,7 [V] |
| Prąd zasilania | 0 - 1,5 [A] |
| Napięcie wyjściowe maksymalne | 0 - 20 [kV] |
| Prąd wyjściowy maksymalny | 0 - 50 [mA] |

Rys. 2.1 - Przetwornica wysokiego napięcia

Powyższa przetwornica została rozmontowana w celu zbadania jej wewnętrznej budowy. Schemat elektryczny przetwornicy:




Rys. 2.2 - Schemat elektryczny przetwornicy

Poszczególne elementy elektroniczne zostały zbadane, dzięki czemu uzyskaliśmy dokładne wartości tych elementów, podane w poniższej tabeli:

Tab. 2.2 - Uzyskane poprzez testy parametry przetwornicy

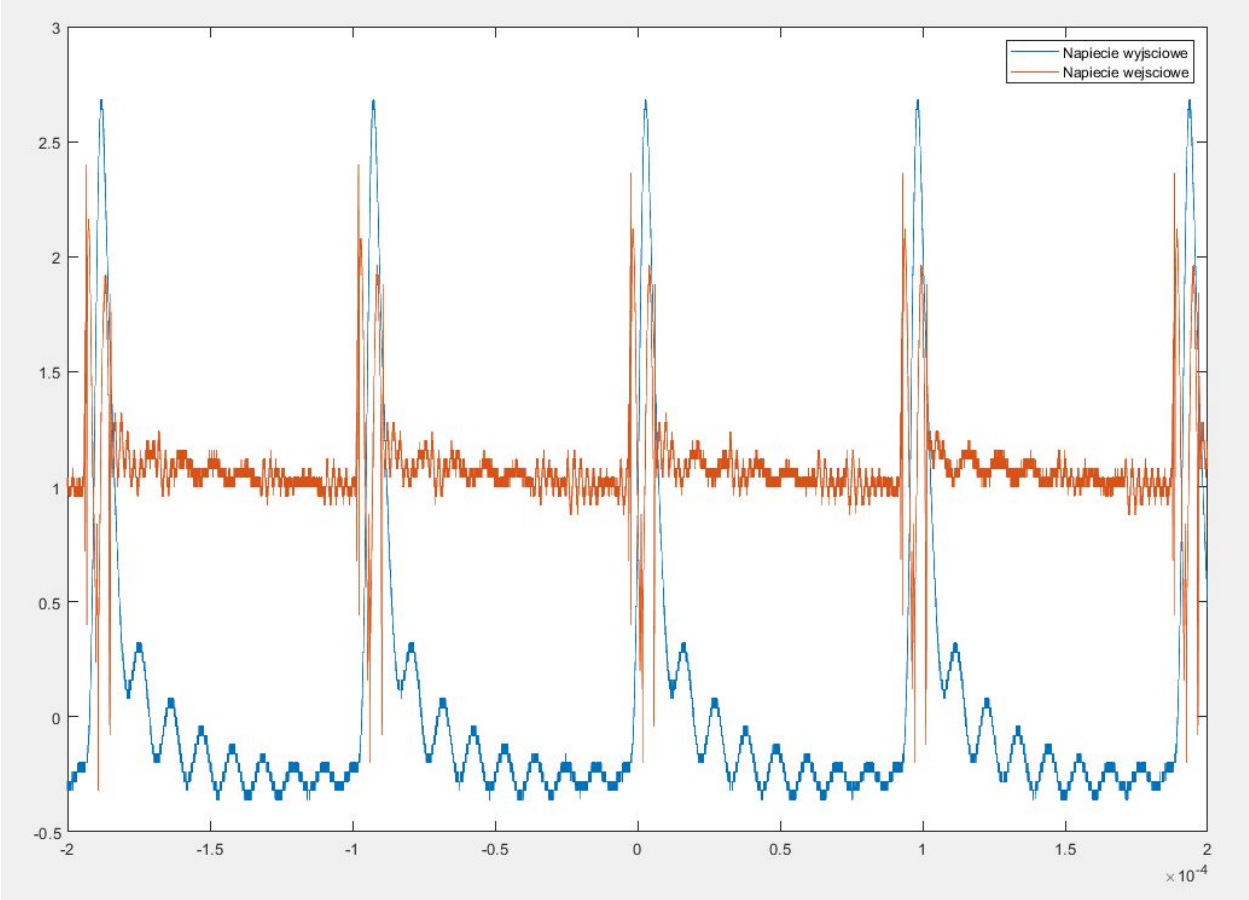
| | |
|---|---|
| Tranzystor NPN | $U_f$ = 0,6 V; hFE = 439 |
| Kondensator C1 | C = 47 µF |
| Rezystor R1 | R = 100 Ω |
| Transformator | Uzwojenie pierwotne: 0.4 Ω, 20 µH<br>Uzwojenie wtórne: 75 0Ω, 470 mH |
| Dioda | $U_f$ = 0,7 V; C = 66 pF |

Przetwornica bazuje na transformatorze podnoszącym napięcie. Uzwojenie pierwotne zasilane jest napięciem 3,7 V, przełączanym za pomocą tranzystora NPN. Baza tego tranzystora połączona jest do obowdu rezonansowego składającego się z kondensatora i cewki uzwojenia pierwotnego. W efekcie uzyskiwane są “szpilki” napięcia podawane na bazę tranzystora, co z kolei powoduje podawanie zmiennego napięcia na uzwojenie transformatora. Odpowiednio zwiększone napięcie pojawi się na zaciskach uzwojenia wtórnego.

Następnie zbadane zostały przebiegi napięcia wyjściowego za pomocą oscyloskopu. W tym celu konieczne było użycie sondy oscyloskopowej przystosowanej do mierzenia wysokich napięć - użytą przez nas sondą była HVP-15HF. Sonda ta skaluje mierzone napięcie w stosunku 1000:1. Przeprowadzone zostały pomiary dla różnych napięć zasilających, natomiast wyjściowe zaciski zostały rozwarte.

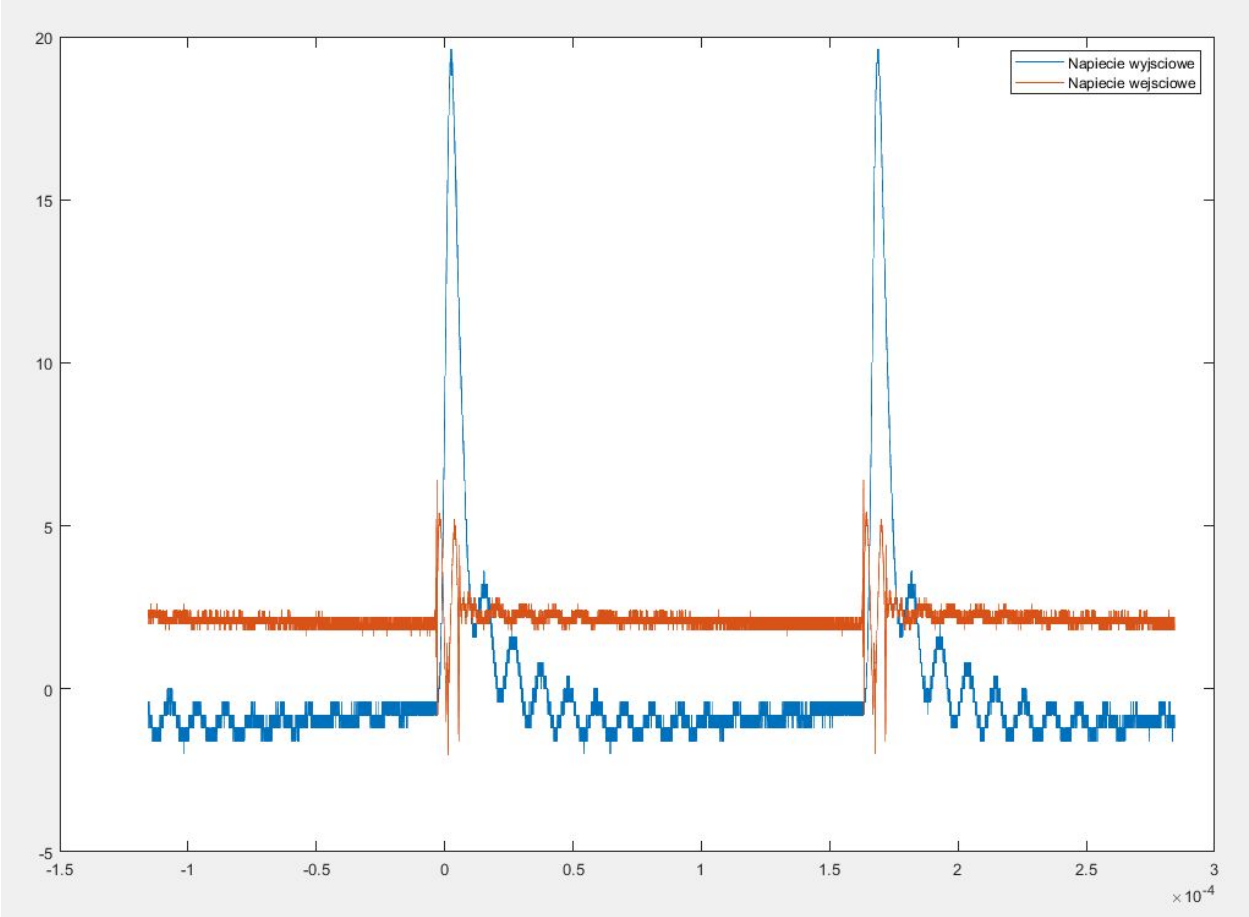
Wyniki pomiarów:

- Dla zasilania $U_{WE}$ = 1V



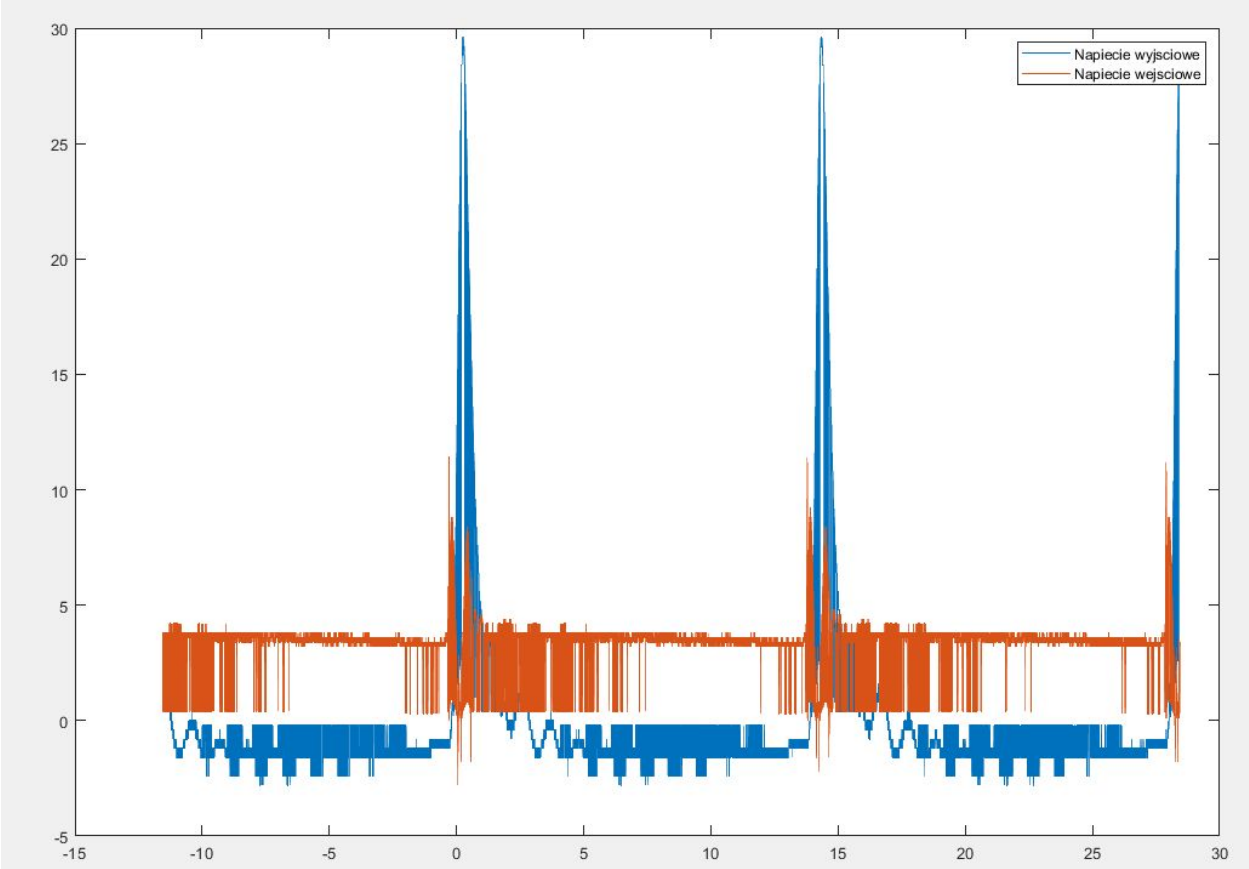

Rys. 2.3 - Przebiegi napięć dla zasilania 1V

- Dla zasilania $U_{WE}$ = 2,1V




Rys. 2.4 - Przebiegi napięć dla zasilania 2,1V

- Dla zasilania $U_{WE}$ = 3.7V (Napięcie wyjściowe dodatkowo przeskalowane o 10 dla czytelności, wynikowa skala: 10000:1)

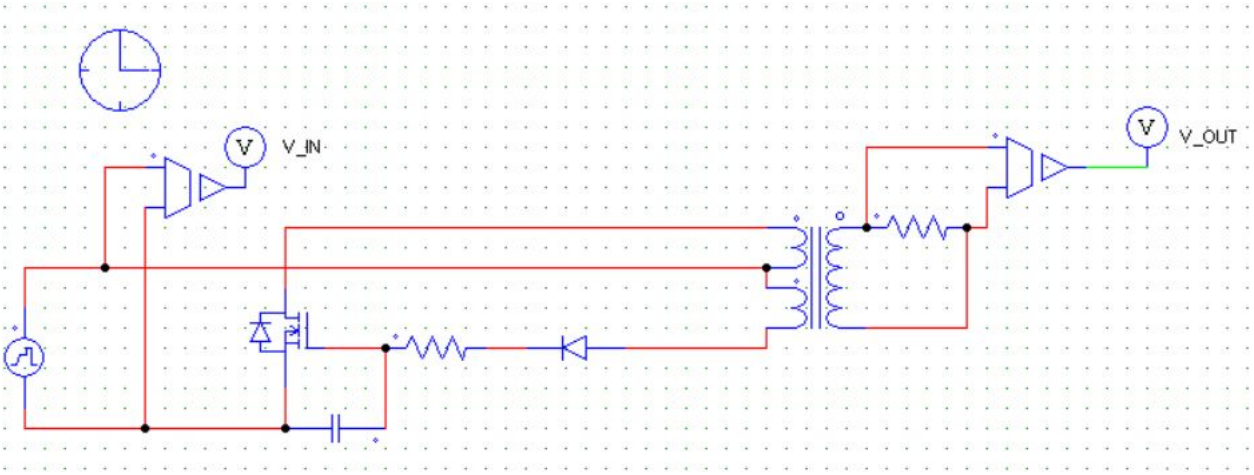



Rys. 2.5 - Przebiegi napięć dla zasilania 3,7V

Z powyższych wykresów widać, że poziom napięcia zasilającego wpływa na maksymalne napięcia generowane przez przetwornice, zatem można w ten sposób kontrolować napięcie wyjściowe przetwornicy.

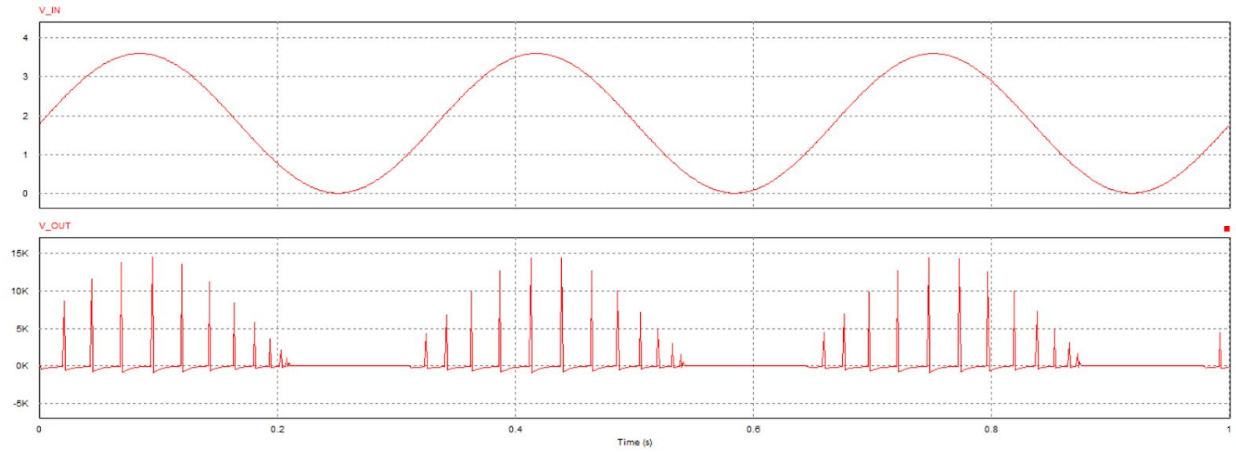
## 3. Symulacja

Po zidentyfikowaniu parametrów wykonaliśmy symulacje układu w oprogramowaniu PSIM. Konieczne jednak było zamienienie tranzystora NPN tranzystorem MOSFET, ponieważ zastosowanie tranzystora NPN powodowało, że symulacja zwracała błędy. Układ symulacyjny powinien jednak dość blisko odwzorowywać zachowanie rzeczywistego układu.

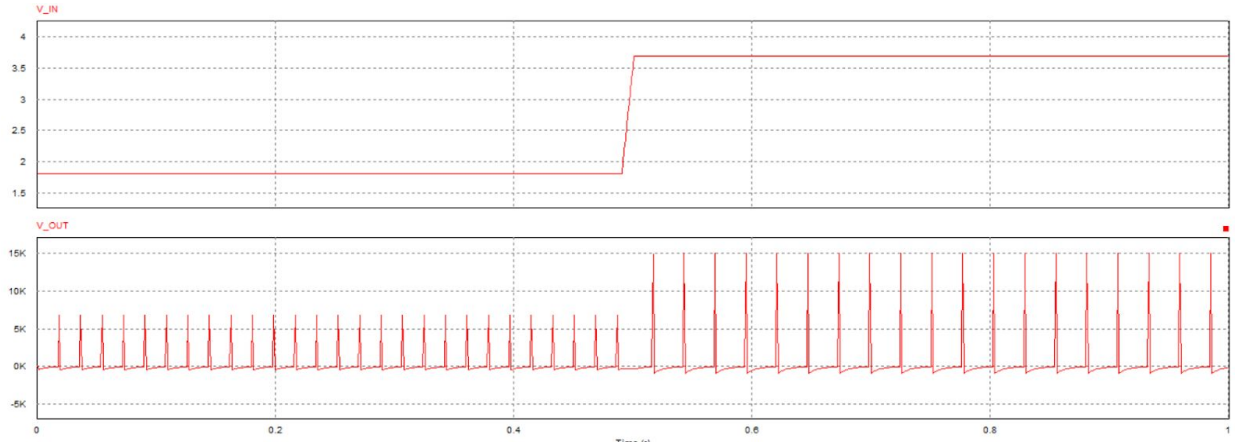



Rys. 3.1 - Schemat przetwornicy wykonany w programie PSIM

Po zbudowaniu układu badany był wpływ napięcia wejściowego przetwornicy na jej napięcie wyjściowe:




Rys. 3.2 - Przebiegi napięcia wejściowego oraz wyjściowego przetwornicy - zasilanie falą sinusoidalną

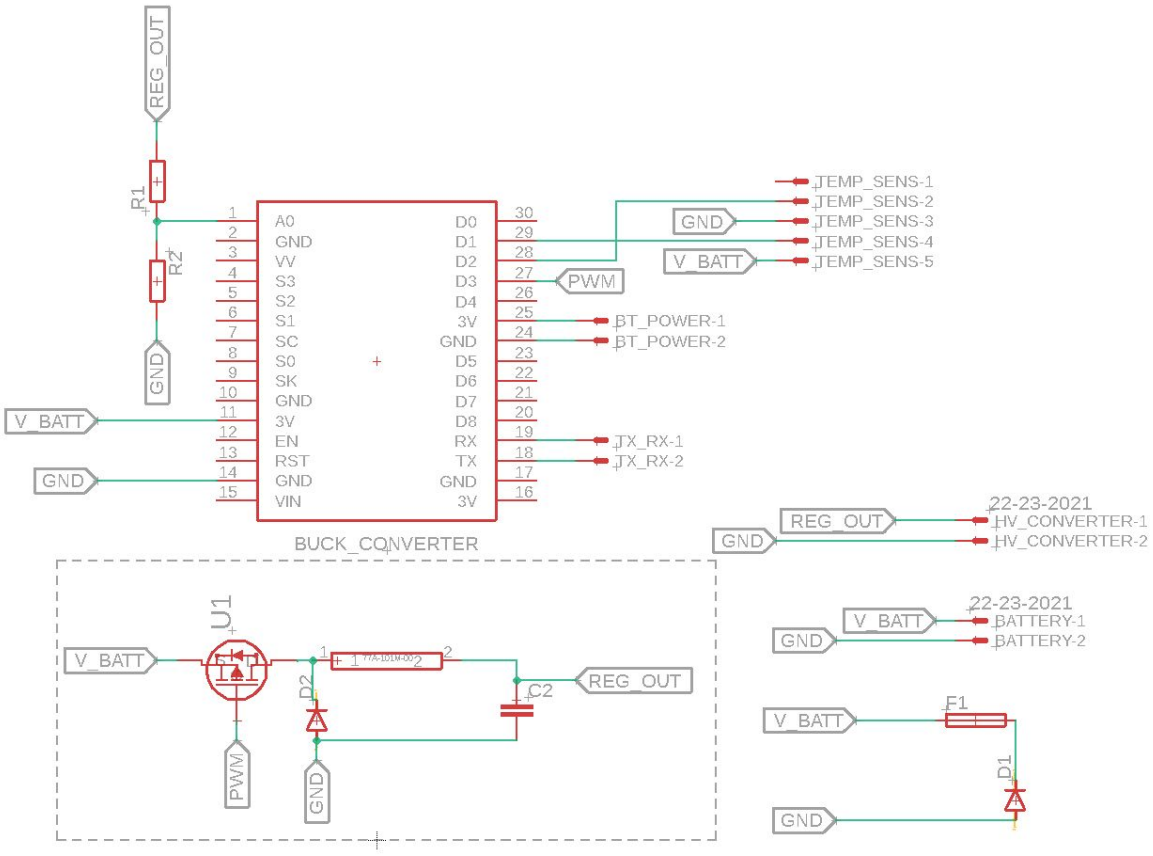



Rys. 3.3 - Przebiegi napięcia wejściowego oraz wyjściowego przetwornicy - różne wartości napięcia stałego

Symulacja pokrywa się z zmierzonymi wartościami i potwierdza możliwość sterowania napięciem wyjściowym poprzez formowanie kształtu napięcia wejściowego.

## 4. Projekt obwodu elektronicznego

Obwód elektroniczny zaprojektowany był w oprogramowaniu EAGLE. Schemat urządzenia:




Rys. 4.1 - Schemat urządzenia wykonany w programie Eagle

Spis elementów elektronicznych użytych do projektu:

Tab. 4.1 - Spis elementów elektronicznych.

| | |
|---|---|
| Cewka L1 | 77A-101M-00 |
| Kondensator elektrolityczny C2 | 470uF |
| Tranzystor U1 | IRF9510PBF |
| Dioda D1, D2 | 1N4007 |
| Bezpiecznik F1 | 021802.5TXP |

| | |
|---|---|
| Gniazdo na bezpiecznik | ZHL76A |
| Rezystory R1, R2 | 10kΩ |
| Czujnik temperatury | TC74A0-5.0VAT |
| Listwy zaciskowe na zasilanie i wyjście | DG308-2.54-02P |
| Moduł bluetooth | HC06 |
| Mikrokontroler | NodeMCU v3 |

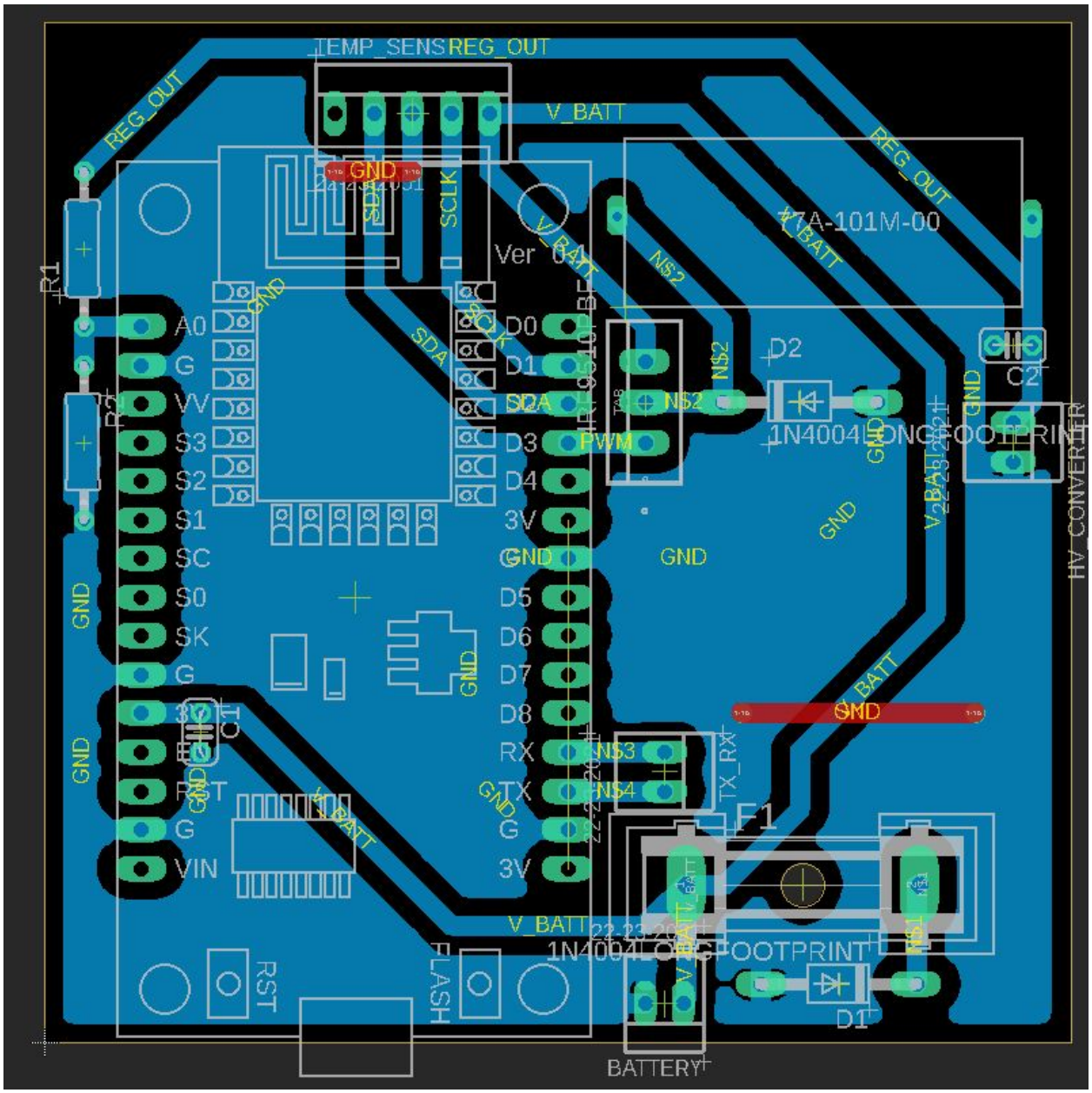
Urządzenie zapewnia zabezpieczenie przed odwrotną polaryzacją oraz przekroczeniem maksymalnego prądu przez obwód składający się z diody i bezpiecznika. W przypadku odwrotnego podłączenia zasilania źródło zasilania zostanie zwarte przez diodę, co wymusi przepływ dużego prądu w efekcie spalając bezpiecznik. Układ wyposażony jest również w czujnik temperatury, który można zamontować na obudowę przetwornicy. Pozwala to na zastosowanie zabezpieczenia termicznego.

Układ sterujący napięciem bazuje na topologii typu buck. Można dzięki temu obniżać napięcie wejściowe od 0 V do Vcc. Cały układ zasilany jest z napięcia 3,7 V, czyli znamionowego napięcia akumulatorów typu li-po.

Istnieje sprzężenie zwrotne pod postacią pomiaru napięcia wyjściowego podawanego na przetwornice.

Jednostką sterującą jest mikrokontroler NodeMCU v3.

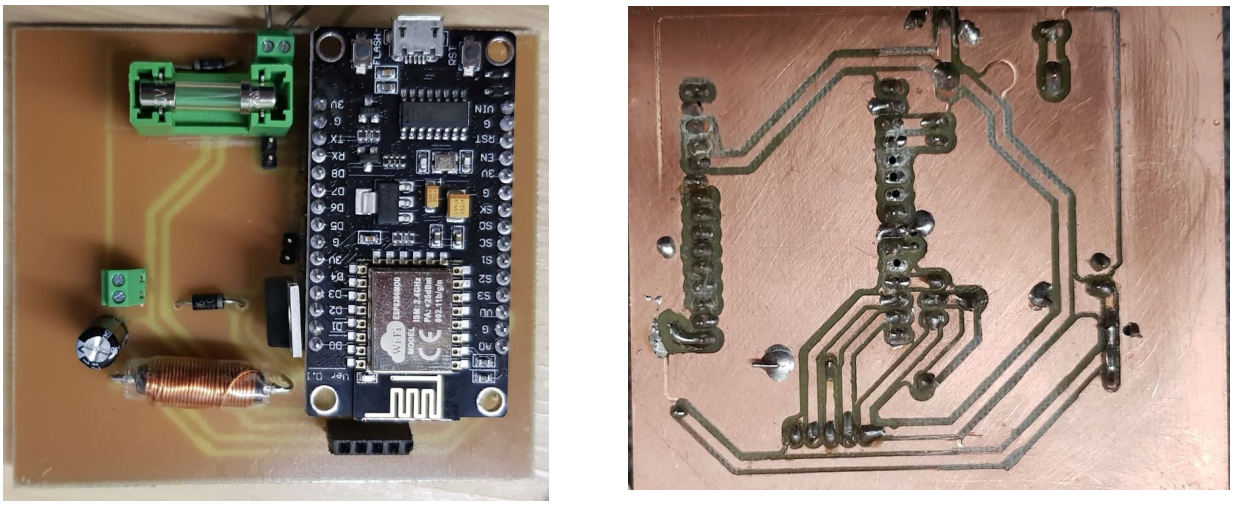
Następnie stworzony został projekt płytki PCB:




Rys. 4.2 - Projekt płytki PCB

Płytka jest jednostronna, wykonana została na frezarce CNC. Niebieskim kolorem zaznaczona jest miedź na płytce, kolor czerwony obrazuje przewody prowadzone po drugiej stronie. Wykorzystany został frez grawerski w kształcie litery V, o kącie 20°, o efektywnej średnicy wycinania równej około 0,85 mm. Grubość warstwy miedzi laminatu wynosi 35 µm, natomiast zastosowana została głębokość wycinania równa 200 µm, aby wziąć pod uwagę nierówności na powierzchni laminatu.

Następnie ręcznie wywiercone zostały otwory w płytce za pomocą wiertła o średnicy 0,5 mm oraz 1 mm, oraz wlutowane zostały elementy elektroniczne. Końcowy efekt:



Rys. 4.3 - Wykonany układ elektroniczny

## 5. Program

Układ NodeMCU v3 można programować za pomocą Arduino IDE poprzez złącze micro USB umieszczone na płytce. Zanim programowanie będzie możliwe należy odłączyć moduł bluetooth od pinów RX i TX układu.

Kod programu zaczyna się od dołączenia odpowiednich bibliotek. Biblioteki te odpowiedzialne za ułatwienie komunikacji przez port szeregowy oraz $I^2C$.
Następnie inicjalizowana jest komunikacja szeregowa poprzez piny RX i TX w celu obsługi modułu Bluetooth.

Następnie definiowane są odpowiednie stałe czasowe określające częstotliwości działania modulacji odpowiednich sygnałów na wejście przetwornicy oraz częstotliwość pomiaru temperatury, a także liczbę kroków występujących podczas modulowania danego kształtu przebiegu sygnału. Kolejnym etapem programu jest przypisanie odpowiednich nazw pinów (analogowy do odczytu aktualnego napięcia na wyjściu NodeMCU oraz cyfrowy do zadawania sygnału PWM). Następnie deklarowane i inicjalizowane są zmienne odpowiadające za: obsługę czujnika temperatury, liczniki, bufor służący w komunikacji Bluetooth oraz za parametry otrzymywane z aplikacji Android.

Kolejnymi elementami programu są funkcje. Pierwsza z nich - *calculatePWM()* - przekształca wartość współczynnika wypełnienia z wartości procentowej na zakres analogowy 0 - 1023 ($2^{10}$), aby ułatwić późniejsze zadawanie tej wartości za pomocą funkcji *analogWrite()*. Kolejna z nich - *readData()* - odpowiada za odczyt danych otrzymanych przez urządzenie mikrokontrolera z aplikacji mobilnej. Następuje w niej odczyt danych bajt po bajcie i czyszczenie bufora, jeśli otrzymany bajt zawiera znak “@” (ASCII 64), oraz czyszczenie bufora samego urządzenia HC-06. Następna funkcja - *calculatePeriod()* - zwraca okres obliczany na podstawie otrzymanej z aplikacji częstotliwości sygnału.

Następnie w programie zawarte są dwie klasyczne funkcje *void* programu: funkcja konfiguracyjna oraz funkcja odpowiadająca za nieskończoną pętlę działania programu. W części konfiguracyjnej rozpoczynana jest komunikacja poprzez port szeregowy, interfejs $I^2C$ oraz moduł Bluetooth. Parametr szybkości transmisji portu szeregowego ustawiony jest na wartość 9600. Skonfigurowane w omawianej funkcji są także tryby pracy pinów.

W głównej pętli programu na samym jej początku sprawdzana jest dostępność urządzenia HC-06. Jeśli komunikacja jest aktywna, to następuje odczyt danych za pomocą wyżej opisanej funkcji *readData()*. Przypisywany jest także aktualny czas w mikrosekundach do odpowiedniej zmiennej mającej ten czas przechowywać. Jest to

pomocne w zadawaniu odpowiedniej częstotliwości działania modulowania sygnałów oraz pomiaru temperatury.

Upływ czasu mierzony jest za pomocą instrukcji warunkowej, która porównuje czas, który upłynął ze stałą wartością interwału. Pierwsza funkcjonalność odpowiada za modulację sygnałów i pracuje z częstotliwością 10 kHz. Odczytywane oraz przeliczane w niej są odpowiednie wartości potrzebne do pracy modulacji sygnału, które zostały odebrane z aplikacji mobilnej. W zależności od wybranego w aplikacji typu sygnału program na wyjściu generuje:

- Sygnał stały - realizowany w prosty sposób za pomocą funkcji *analogWrite()* o wypełnieniu zależnym od wartości otrzymanej z aplikacji.
- Sygnał sinusoidalny - realizowany przy pomocy określenia wartości funkcji w każdym z kroków. Dla każdego kroku sygnał ten realizowany jest za pomocą wzoru:

$$y = A \cdot sin(2 \cdot \pi \cdot n) / n_{max} + A ,$$

gdzie:
  - $A$ - amplituda sygnału, w tym wypadku określana współczynnikiem wypełnienia PWM,
  - $n$ - aktualny krok,
  - $n_{max}$ - całkowita liczba kroków.

Taka realizacja pozwala na zadawanie wyłącznie dodatnich wartości napięcia o kształcie sinusoidalnym. Dla każdego kroku liczone jest wypełnienie PWM, które jest wysterowywane na wyjściu cyfrowym. W efekcie otrzymujemy sygnał prostokątny, którego współczynnik wypełnienia zmienia się wraz ze zmianą wartości funkcji sinus.

- Sygnał trójkątny - realizowany za pomocą funkcji liniowych o dodatnim i ujemnym znaku. Przez połowę kroków jednego okresu wartość współczynnika PWM rośnie liniowo, a przez drugą połowę maleje liniowo. Wzór:

  Dla pierwszej połowy okresu:

$$y = (A \cdot n) / (n_{max} / 2)$$

  Dla drugiej połowy okresu:

$$y = (A \cdot (100 - n)) / (n_{max} / 2)$$

Oznaczenia jak wyżej.

- Sygnał prostokątny - dla pierwszej połowy okresu zadawana jest wartość na wyjściu tak samo jak w przypadku sygnału stałego, natomiast dla drugiej połowy okresu zadawana jest wartość zerowa.

Druga funkcjonalność głównej pętli programu to odczyt danych z czujnika temperatury, którego cechuje częstotliwość 1 Hz. Sprawdzana jest aktywność urządzenia w komunikacji $I^2C$. Jeśli urządzenie jest dostępne następuje odczyt temperatury, której wartość następnie jest zapisywana do odpowiedniej tablicy, dzięki czemu będzie mogła został wysłana do aplikacji.

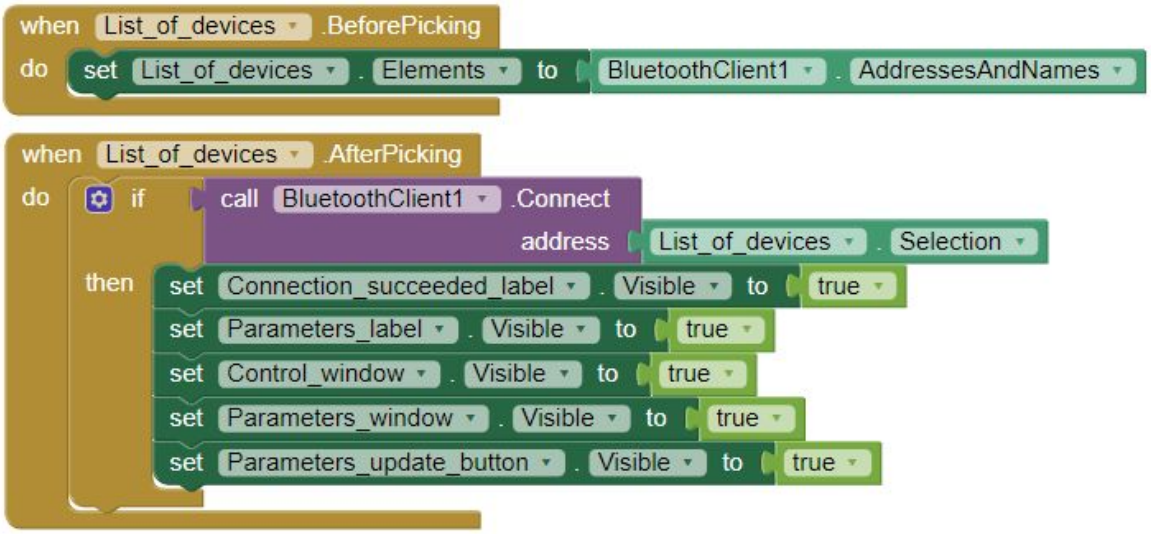
Ostatnią funkcjonalnością głównej pętli programu jest odczyt sekwencji danych otrzymanych z aplikacji. Jeśli kontrolna sekwencja danych jest poprawna z założeniami, odczytywane do odpowiednich zmiennych są parametry ustawiane z ekranu aplikacji mobilnej. Na podstawie tych danych wykonywana jest wyżej opisana modulacja sygnałów.

## 6. Aplikacja

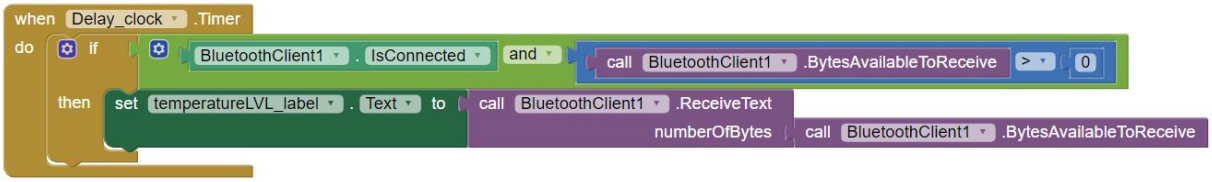
Interfejs użytkownika stanowi aplikacja mobilna pozwalająca na wybór kształtu napięcia podawanego na przetwornicę, współczynnika wypełnienia PWM, oraz częstotliwości sygnału (dla sygnałów innych niż napięcie stałe). Aplikacja wyświetla również odczytaną na urządzeniu i przesłaną na smartfon temperaturę z czujnika.

Aplikacja została napisana w darmowym środowisku programistycznym Mit App Inventor które oferuje proste tworzenie aplikacji na platformy Android oraz IOS. Program jest tworzony graficznie przy pomocy bloków funkcyjnych co sprawia, że program napisany w tym środowisku jest prosty do zrozumienia oraz łatwy w modyfikacji. Jedną z zalet używania Mit App Inventor jest możliwość instalowania aplikacji na telefon przez generowanie i odczytywania kodu QR prosto z monitora, co usprawnia wprowadzanie poprawek w kodzie.

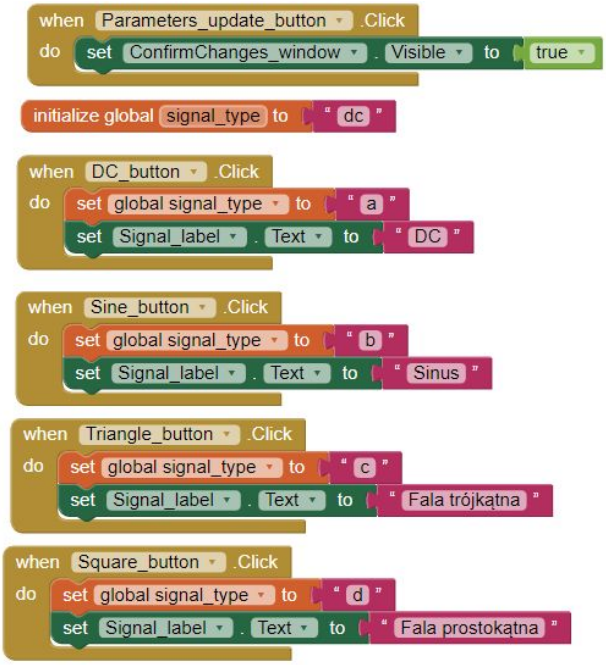
Poniższe bloki funkcyjne odpowiadają za wyświetlanie na ekranie smartfonu listy dostępnych urządzeń bluetooth gotowych do połączenia. Jeżeli zostanie wybrane jedno z nich to następuje wyświetlenie się ekranu głównego aplikacji który jest przedstawiony na rysunku 6.1.



Na początku sprawdzane jest przy pomocy bloku logicznego (AND) czy bluetooth jest podłączony z urządzeniem oraz czy aplikacja odbiera dane. Jeżeli spełnione są te warunki, to następuje przypisanie odebranych danych do etykiety *temperatureLVL_label* związanej z wyświetlaną temperaturą.

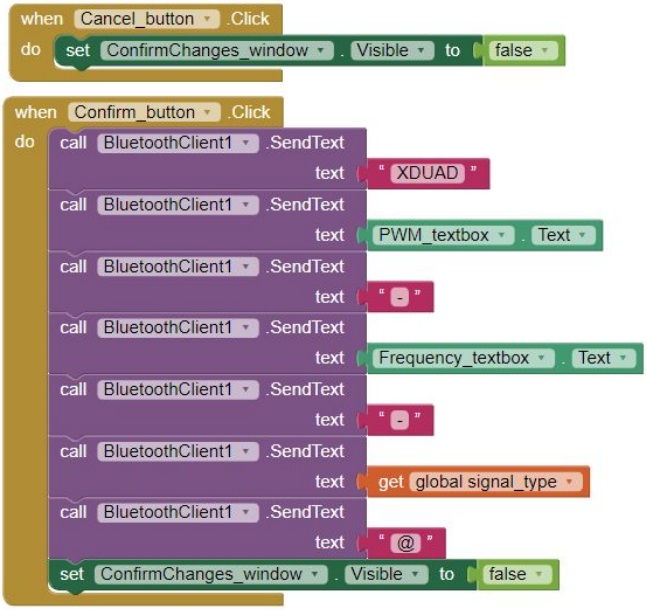


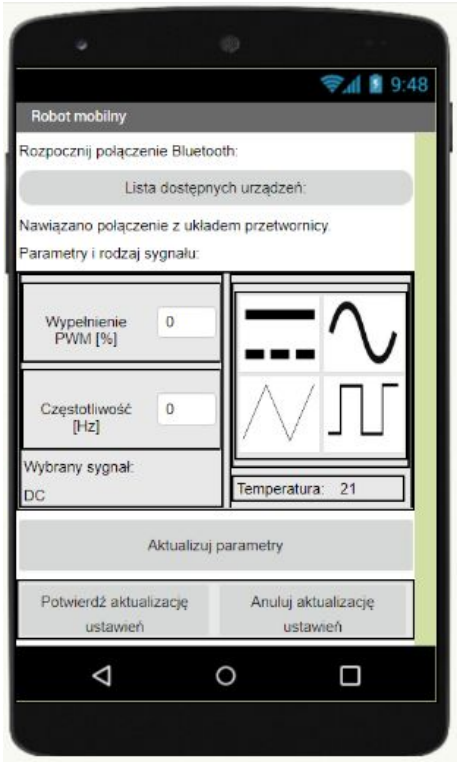

Wybór rodzaju sygnału jaki zostanie podany na wejście przetwornicy dokonywany jest poprzez przyciski zaprogramowane w taki sposób, aby po kliknięciu na odpowiedni rysunek ustawiana była zmienna globalna na odpowiednią wartość.




Po naciśnięciu przycisku *aktualizuj parametry* oraz *potwierdź aktualizację ustawień* wysyłana jest ramka. Przyciski umożliwiające potwierdzenie wysłania danych lub ich odrzucenie pojawiają się dopiero po naciśnięciu przycisku aktualizacji. Służy to zabezpieczeniu przeciw przypadkowym wysłaniem na przetwornicę parametrów, co w pewnych sytuacjach mogłoby okazać się groźne w danym zastosowaniu urządzenia (np. nagła zmiana napięcia lub zmiana kształtu sygnału). Początek ramki stanowi ustalona sekwencja znaków, której pełne i poprawne odczytanie na urządzeniu ma potwierdzać prawidłowość transmisji danych. Następnie wysyłane są odpowiednio: wartość współczynnika wypełnienia [%], częstotliwość [Hz] oraz wybrany typ sygnału.

Koniec wysyłania ramki sygnalizowany jest znakiem ”@” (ASCII 64). Potwierdzenie lub anulowanie wysyłania powoduje ponowne ukrycie się przycisków za te działania odpowiedzialnych.




Rys. 6.1 - Interfejs aplikacji

## 7. Efekt końcowy

Wykonany obwód elektroniczny jest sprawny, zabezpieczenia przed odwrotną polaryzacją oraz przekroczeniem prądu działają. Czujnik temperatury jest poprawnie obsłużony i wysyła informacje do aplikacji mobilnej. Testy aplikacji przeprowadzone były na telefonie Samsung Galaxy S8, z wersją androida 8.0.0. Aplikacja poprawnie komunikuje się z mikrokontrolerem poprzez moduł bluetooth i wysyła ramki danych.

Nie udało się jednak poprawnie zadawać wartości napięcia zasilającego przetwornicę. Część kodu mikrokontrolera odpowiedzialnego za odbiór danych działa tylko raz - gdy aplikacja wyśle informację o nowych parametrach zasilania program działa poprawnie, lecz wysłanie danych ponownie nie powoduje zaktualizowania wartości. Ponadto, program zacina się w nieokreślonym momencie jego działania, co powoduje zatrzymanie się komunikacji z aplikacją oraz zablokowanie zadanej wartości napięcia wyjściowego na wartość maksymalną. Nie udało się dojść do przyczyny tych błędów.

Wszystkie pliki użyte do wykonania projektu znajdują się na repozytorium: https://github.com/michal6271/robotyka2