Politechnika Łódzka

PROJEKT KOMPETENCYJNY

“System zdalnego sterowania instalacją budynkową z wykorzystaniem aplikacji mobilnej”

Prowadzący: dr inż. Jacek Rymaszewski

Autor: Piotr Gałecki

Kierunek: Elektrotechnika

Semestr V

Specjalność: Przetworniki elektromechaniczne

Rok akademicki 2017/18

Spis treści

1. Wprowadzenie 3
1.1. Cel projektu 3
1.2. Poruszana tematyka 3
2. Projekt 4
2.1. Założenia projektowe 4
2.2. Struktura projektu 5
2.2.1. Warstwa sprzętowa 5
2.2.2. Skrypt wykonawczy 8
2.2.3. Aplikacja sterująca 12
2.3. Wykorzystane podzespoły 15
2.3.1. NodeMCU 15
2.3.2. Termometr cyfrowy DS18B20 16
2.3.3. Czujnik Halla TLE4905L 17
2.3.4. ThingSpeak 18
3. Określenie prędkości obrotowej z wykorzystaniem czujnika Halla 19
4. Prezentacja działającego projektu 22
5. Zakończenie 28

1. WPROWADZENIE

1.1. Cel projektu

Celem projektu jest zaprojektowanie, a następnie stworzenie systemu zdalnego sterowania modelem instalacji elektrycznej za pomocą urządzenia mobilnego. Projekt ma na celu poszerzenie wiedzy związanej z automatyką budynkową, bezprzewodową komunikacją urządzeń mikroprocesorowych oraz sposobami pomiaru prędkości obrotowej w maszynach elektrycznych.

1.2.Poruszana tematyka

Tematyka projektu dotyczy sterowania urządzeniami automatyki budynkowej poprzez system zarządzania budynkiem (ang. Building Management System). Systemy tego typu (zwane skrótowo BMS) umożliwiają integrację, kontrolę oraz monitorowanie wszystkich instalacji w budynku (począwszy od instalacji elektrycznej, a kończąc na instalacji HVAC (ang. *Heating, Ventillating, Air Cooling*). Systemy BMS są powszechnie wykorzystywane w biurach, przykładowy system BMS przedstawiony został na rysunku 1.

Rysunek 1. Przykładowy system BMS.

Sterowanie inteligentnym budynkiem może być realizowane z poziomu komputera PC lub tak jak w projekcie, z poziomu urządzenia mobilnego.

2. PROJEKT

2.1. Założenia projektowe

Projekt ma umożliwiać zdalne sterowanie modelem instalacji budynkowej poprzez urządzenie mobilne obsługujące system operacyjny Android. Funkcję modelu instalacji budynkowej pełnił będzie wentylator, którego prędkość obrotowa będzie regulowana poprzez napięcie podawane na zaciski wentylatora. Projektowany system oferował będzie dwa tryby pracy: tryb AUTO – w którym prędkość obrotowa wentylatora będzie dobierana w taki sposób, aby temperatura cyfrowego termometru osiągnęła wartość równą bądź mniejszą od zadanej, oraz tryb MANUAL – w którym użytkownik nastawiał będzie wartość prędkości obrotowej wentylatora. Sterowanie napięciem wentylatora realizowane poprzez zmianę współczynnika wypełnienia sygnału metodą PWM (ang. *Pulse Width Modulation*). Komunikacja między aplikacją a jednostką mikroprocesorową odbywać się będzie poprzez sieć bezprzewodową Wi – Fi.

2.2. Struktura projektu

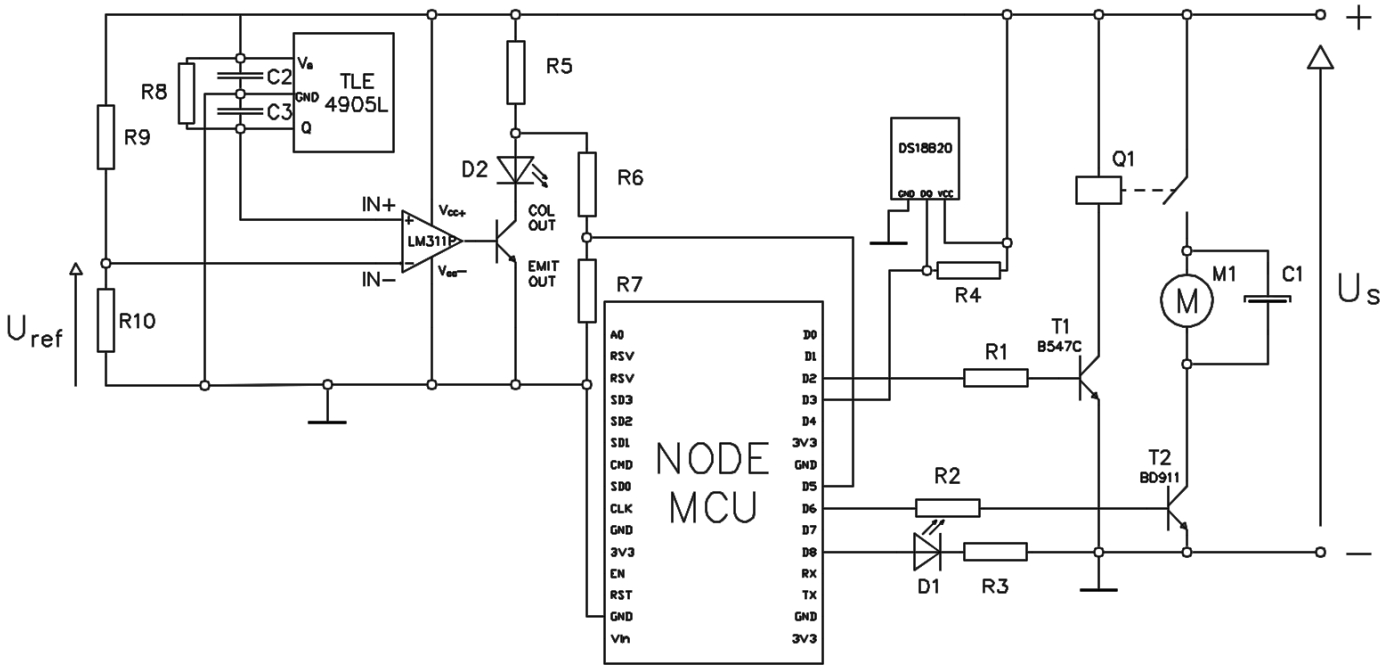
W projekcie wyróżnić można trzy odrębne części, takie jak:

- warstwa sprzętowa,
- skrypt wykonawczy,
- aplikacja sterująca.

W kolejnych podrozdziałach przedstawione zostaną kolejno każda z wymienionych części projektu.

2.2.1. Warstwa sprzętowa

Schemat ideowy warstwy sprzętowej projektu przedstawiony został na rysunku 2. W przypadku układów scalonych zastosowane zostały oznaczenia stosowane w kartach katalogowych producentów.



Rysunek 2. Schemat ideowy warstwy sprzętowej projektu.

W schemacie ideowym warstwy sprzętowej wyróżnić można następujące części składowe

- czujnik Halla

- komparator LM311P
- cyfrowy termometr DS18B20
- układ sterowania wentylatorem

Poniżej opisane zostaną skrótowo wszystkie wymienione części składowe układu.

Czujnik Halla

Czujnik pola magnetycznego TLE4905L został zastosowany zgodnie z zastosowaniem producenta podanym w karcie katalogowej produktu, a mianowicie z wykorzystaniem kondensatorów ceramicznych C2 i C3 wpiętych między wszystkie wyprowadzenia układu. Wyprowadzenie Q (tj. Output) zostało podciągnięte do napięcia zasilania poprzez rezystor R8, co zapewnia utrzymywanie się stanu wysokiego na wyjściu, kiedy indukcja pola magnetycznego nie przekracza wartości załączenia (ang. *Turn-ON induction*).

Komparator LM311P

Komparator LM311P porównuje napięcie z wyprowadzenia Q czujnika Halla z napięciem odniesienia U_{ref} wynoszącym 9V. Jeśli napięcie na wejściu nieodwracającym IN+ osiągnie wartość mniejszą niż U_{ref}, to wbudowany w komparator tranzystor NPN zostaje otwarty, co generuje zero logiczne na jego kolektorze. Sygnał logiczny z kolektora jest doprowadzony poprzez dzielnik napięcia R6/R7 do pinu D5 modułu NodeMCU. Dzielnik napięcia R6 i R7 jest zastosowany w celu dopasowania napięcia stanu wysokiego na kolektorze do poziomu 3.3V stosowanego w module NodeMCU (jego niezastosowanie mogłoby doprowadzić do pojawienia się napięcia 12V na pinie D5, co spowodowałoby uszkodzenie modułu).

Dioda D2 ma na celu wizualną sygnalizację działania czujnika Halla.

Cyfrowy termometr DS18B20

Układ scalony DS18B20 został wyposażony w rezystor R4 pełniący funkcję rezystora pull-up (podobnie jak R8 w przypadku TLE4905L). Sygnał cyfrowy z wyprowadzenia DQ jest doprowadzony do pinu D5 modułu NodeMCU. Termometr jest zasilony z zewnętrznego źródła zasilania.

Układ sterowania wentylatorem

Sterowanie wentylatorem oparte zostało na dwóch tranzystorach: T1 oraz T2. Tranzystor T1 służy do podawania napięcia na cewkę napędową przekaźnika, jest wykorzystany w celu odciążenia modułu NodeMCU z zasilania cewki, co wiązałoby się z nieuzasadnionym obciążeniem prądowym układu mikroprocesorowego. Zastosowany został tranzystor NPN o oznaczeniu B547C, który zapewnia duże wzmocnienie prądowe (wg danych katalogowych wartość typowa wzmocnienia to 600).

Tranzystor T2 służy do wzmacniania prądowego sygnału PWM z pinu D6. Biorąc pod uwagę prąd znamionowy wentylatora wynoszący 144 mA, zastosowany został tranzystor NPN o symbolu BD911 w obudowie TO-220, która zapewnia znacznie większą moc możliwą do rozproszenia. Niska częstotliwość kluczowania (wynosząca 1kHz) uzasadnia zastosowanie tranzystora NPN zamiast tranzystorów przeznaczonych do szybkiego kluczowania.

Kondensator elektrolityczny C1 ma za zadanie uśrednić napięcie podawane na zaciski wentylatora M1oraz rozszerzyć możliwości sterowania jego prędkością obrotową „w dół”.

Dioda D1 pełni funkcję wizualnego potwierdzenia przyjęcia wiadomości przez moduł NodeMCU.

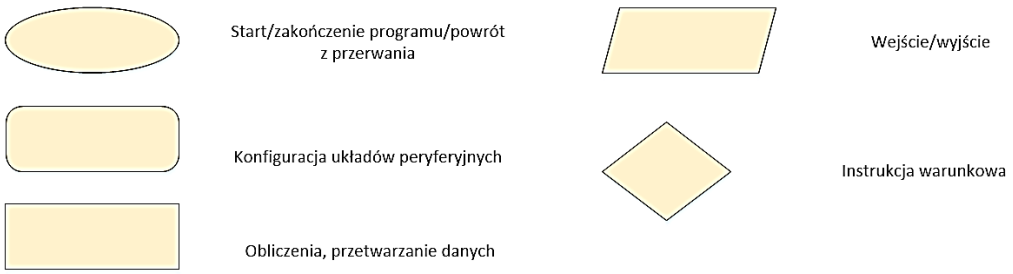
Układ został zaprojektowany do zasilania napięciem stałym wynoszącym 12 V.

2.2.2. Skrypt wykonawczy

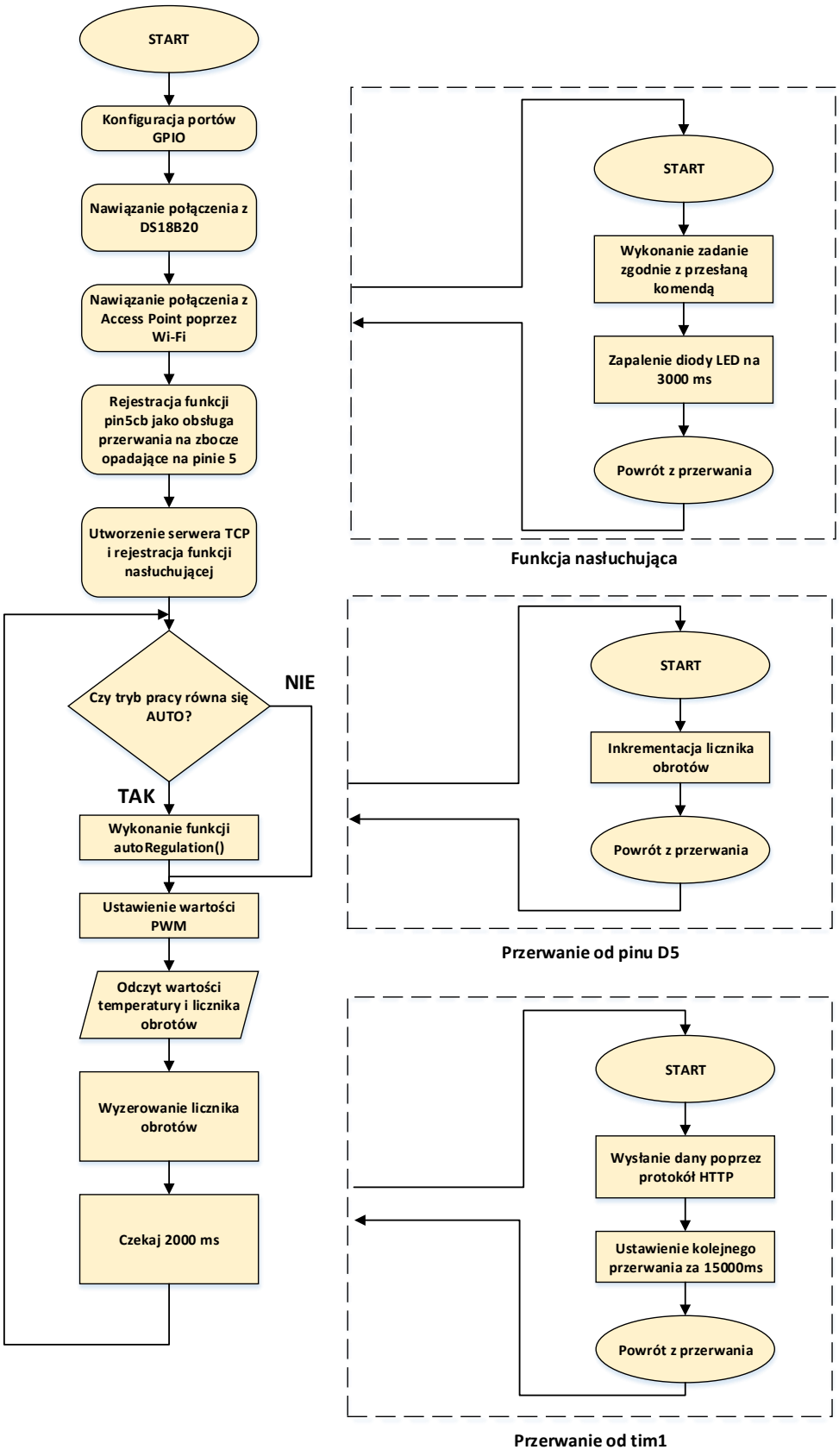
Skrypt wykonywany przez moduł NodeMCU został zaimplementowany w języku skryptowym Lua. Należy zaznaczyć, że moduł ten bazuje na wykorzystaniu przerwań, z których najczęstszymi są przerwania od tak zwanych *timerów* (oznaczanych jako *tim1/tim2/..*). Rysunek 2 zawiera oznaczania, które zostały zastosowane w celu przedstawienia struktury skryptu w postaci algorytmów na rysunku 3 – dotyczącym całego skryptu wykonawczego i na rysunku 4 – przedstawiającym strukturę funkcji *autoRegulation()*.

Na rysunku 3 linią przerywaną objęte zostały trzy przerwania:

- funkcja nasłuchująca – wykonywana gdy serwer TCP zgłosi nadejście danych. Jej zadaniem jest wykonanie odpowiedniej czynności w zależności od komendy.,
- przerwanie od pinu D5 – wykonywane jest gdy na pinie D5 zostanie wykryte zbocze opadające sygnału napięciowego z czujnika Halla. Reakcją układu jest inkrementacja licznika obrotów (zbocze opadające oznacza, że czujnik Halla wykrył pole magnetyczne),
- przerwanie od *tim1* – wykonywane co 15 000, służy do wysłania danych zgromadzonych przez moduł na serwer Thingspeak poprzez protokół http

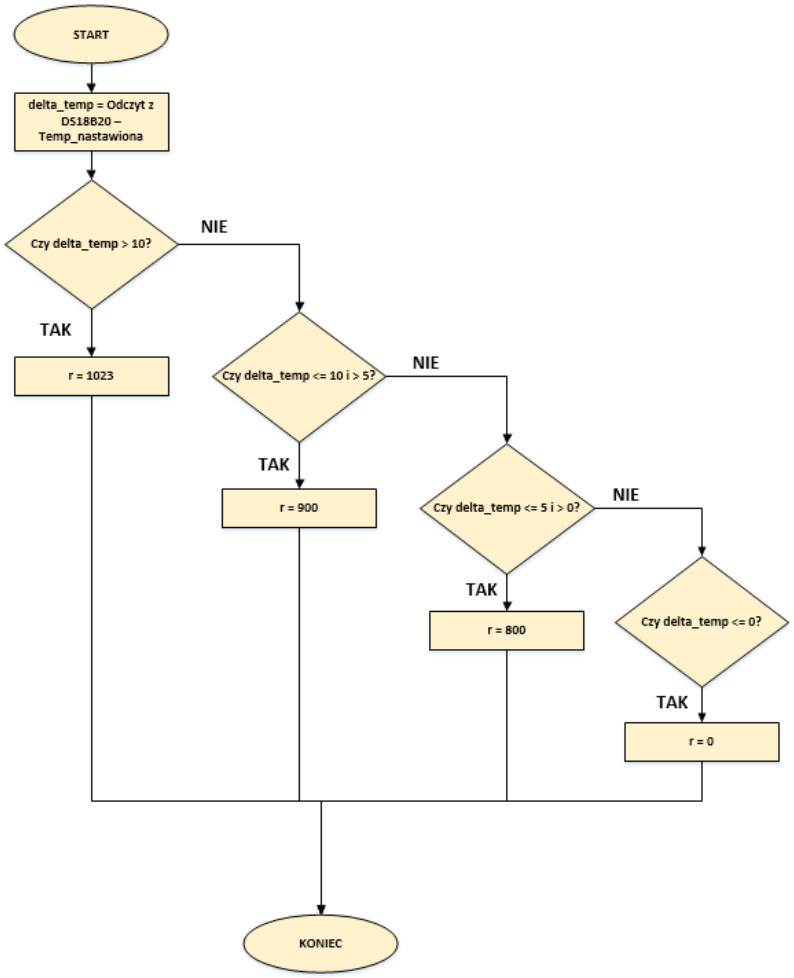


Rysunek 3. Zastosowane oznaczenia.



Rysunek 4. Algorytm skryptu wykonawczego.

Na rysunku 4 przedstawiony został diagram UML funkcji *autoRegulation()*. Za pomocą prostych instrukcji warunkowych współczynnik wypełnienia równy $D = \frac{r}{1023}$ jest ustawiany zgodnie z odczytem temperatury z cyfrowego termometru DS18B20 oraz nastawionej przez użytkownika wartości temperatury.



Rysunek 5. Funkcja autoRegulation().

W projekcie zastosowane zostały trzy komendy służące do sterowania. W standardzie komunikacji przyjęte zostało założenie, że wyraz z dużych liter oznacza komendę, zaś wartość (oznaczona jako val) następująca po znaku „#” oznacza wartość nastawy i jest interpretowana w zależności od rodzaju komendy.

- **RELAY#val**

Służy do podania napięcia na bramkę tranzystora sterującego przekaźnikiem. Możliwe stany val to: „ON” oraz „OFF”.

Przykład:

RELAY#ON – powoduje załączenie przekaźnika

- **AUTO#val**

Służy do nastawienia wartości temperatury do regulacji i jednoczesnej zmiany tryb pracy na AUTO.

Przykład:

AUTO#20 – powoduje dążenie regulatora do osiągnięcia temperatury równej 20 stopni Celsjusza

- **MANUAL#val**

Służy do nastawienia wartości współczynnika wypełnienia sygnału PWM i jednoczesnej zmiany trybu pracy na MANUAL.

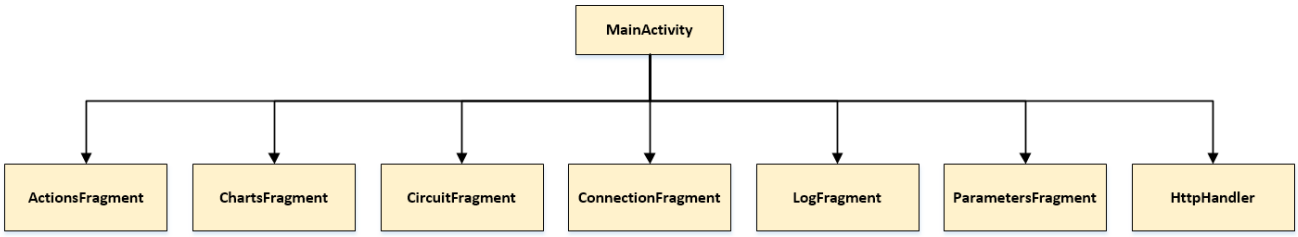
Przykład:

MANUAL#500 – powoduje nastawienie współczynnika wypełnienia sygnału $D = \frac{500}{1023}$

2.2.3. Aplikacja sterująca

Aplikacja na urządzenia mobilne obsługiwane przez system operacyjny Android została zaprogramowana w środowisku programistycznym Android Studio rozwijanym przez firmę Google. Zastosowanie języka angielskiego w GUI aplikacji umożliwia jej użycie przez użytkowników niewładających językiem polskim.

Struktura aplikacji przedstawiona została na rysunku 6.



Rysunek 6. Struktura aplikacji.

Poniżej opisane zostały krótko wszystkie zastosowane klasy:

- MainActivity – główna aktywność[1] aplikacji, obsługująca połączenie z jednostką sterującą i nadzorująca wszystkie fragmenty w programie
- ActionsFragment – fragment[2] służący do nastawy zadanej wartości temperatury bądź współczynnika wypełnienia
- ChartsFragment – fragment służący do sporządzania wykresów ze zgromadzonych danych
- CircuitFragment – fragment wykorzystywany do graficznego przedstawienia instalacji
- ConnectionFragment – fragment pozwalający na nawiązanie połączenia z modułem NodeMCU
- LogFragment – fragment służący do wyświetlania informacji o stanie połączenia
- ParametersFragment – fragment prezentujący dane odebrane z serwera Thingspeak

[1] Aktywność to charakterystyczna dla systemu operacyjnego Android struktura, która pozwala na interakcję użytkownika z aplikacją

[2] Fragmenty są wykorzystywane przez aktywności w celu nadania kodowi budowy modułowej. Fragment stanowi część aktywności, w jednej aktywności może być wykorzystanych wiele fragmentów

- HttpHandler – klasa służąca do komunikacji z serwerem Thingspeak

Aplikacja została napisana w języku Java, który jest przyjęty za główny język programowania w środowisku operacyjnym Android. Przy implementacji ChartsFragment wykorzystana została biblioteka MPAndroidChart udostępniana za darmo przez jej twórcę na portalu GitHub [3].

Zrzuty ekranu pokazujące wykonaną aplikację zostaną zawarte w dalszych rozdziałach.

Projekt uwzględnia dwustronną komunikację między aplikacją a modułem NodeMCU. Komunikacja będzie oparta na innych założeniach i protokołach, w zależności od tego, która jednostka wysyła informacje.

– **Komunikacja aplikacji z modułem NodeMCU**

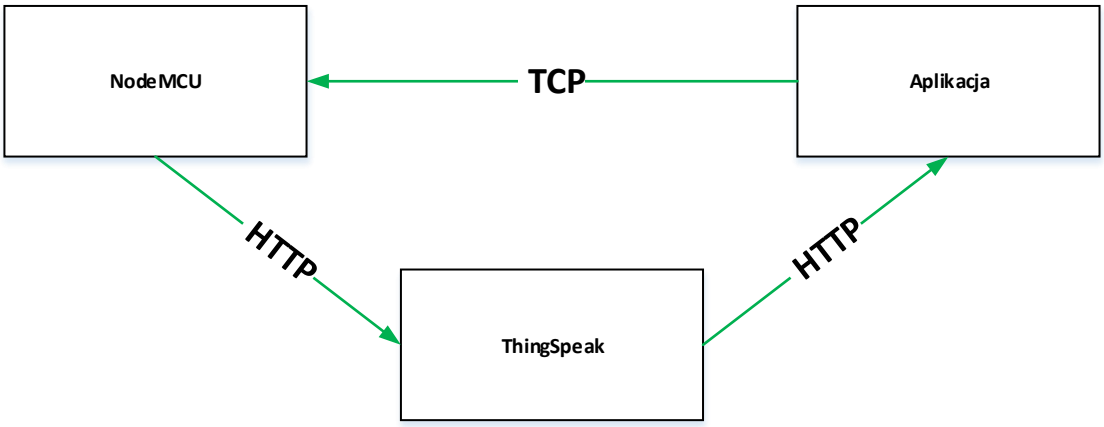
Do komunikacji aplikacji z modułem wykorzystany zostanie protokół TCP. Protokół TCP jest to niezawodny, strumieniowy protokół komunikacyjny, korzystający z protokołu IP do wysyłania i odbierania danych. Jego użycie w projekcie gwarantuje dotarcie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów, wysłanych z urządzenia mobilnego do moduł NodeMCU, co jest kluczowe w przypadku sterowania rzeczywistymi obiektami.

– **Komunikacja modułu NodeMCU z aplikacją**

Komunikacja w tym przypadku wykorzystywać będzie serwer ThingSpeak, który umożliwia gromadzenie i analizowanie danych, co jest jego zaletą w porównaniu do przesyłania informacji przez protokół TCP. Moduł wysyłał będzie dane do serwera za pomocą protokołu http, zaś aplikacja będzie pobierała co określony czas dane z serwera w formacie JSON.

[3]Biblioteka znajduje się pod adresem *https://github.com/PhilJay/MPAndroidChart*

Rysunek 7 przedstawia schemat realizacji komunikacji dwustronnej w projekcie.



Rysunek 7. Komunikacja dwukierunkowa w projekcie.

2.3. Wykorzystane podzespoły

Poniżej zawarty został krótki opis wybranych podzespołów użytych w projekcie.

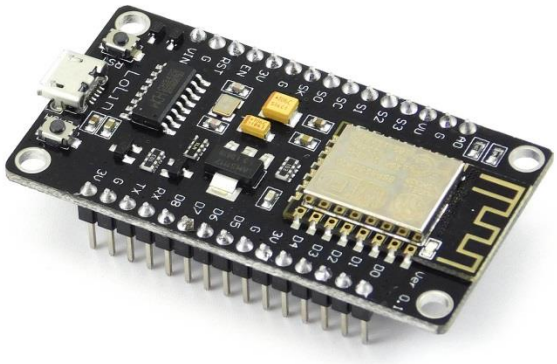
2.3.1. NodeMCU

NodeMCU to moduł WiFi bazujący na popularnym układzie ESP8266. Wyposażony jest w 10 pinów GPIO (z których każdy może być źródłem sygnału PWM), magistralę I^2C, 1-Wire, 10-bitowy przetwornik A/C oraz antenę PCB. Moduł może być programowany bezpośrednio przez port USB dzięki wbudowanemu konwerterowi USB-UART. Posiada 4 MB pamięci Flash oraz procesor RISC 80 MHz. NodeMCU posiada wgrane oprogramowanie umożliwiające programowanie za pomocą języka Lua.

Opisywany moduł cechuje się niską ceną wynoszącą około 20 zł, dzięki czemu jest chętnie wykorzystywany w aplikacjach *IoT (ang. Internet of Things).* Przykładowy egzemplarz modułu NodeMCU przedstawiony został na rysunku 5.

Dane techniczne:

- Moduł zbudowany w oparciu o układ ESP8266-12F z anteną PCB
- Łączność Wi-Fi w standardzie 802.11 b/g/n
- Działa w trybach AP (Access Point), STA (Standalone), AP+STA
- Zasilanie: 3.3V (lub 5V przez port USB)
- Procesor RISC 80MHz
- 10 portów GPIO - PWM / I2C / SPI / 1-Wire
- Maksymalne natężenie na pinach I/O: 12mA
- Konwerter USB-UART - CH340
- Konwerter ADC - 10-bitowy
- Złącze micro USB



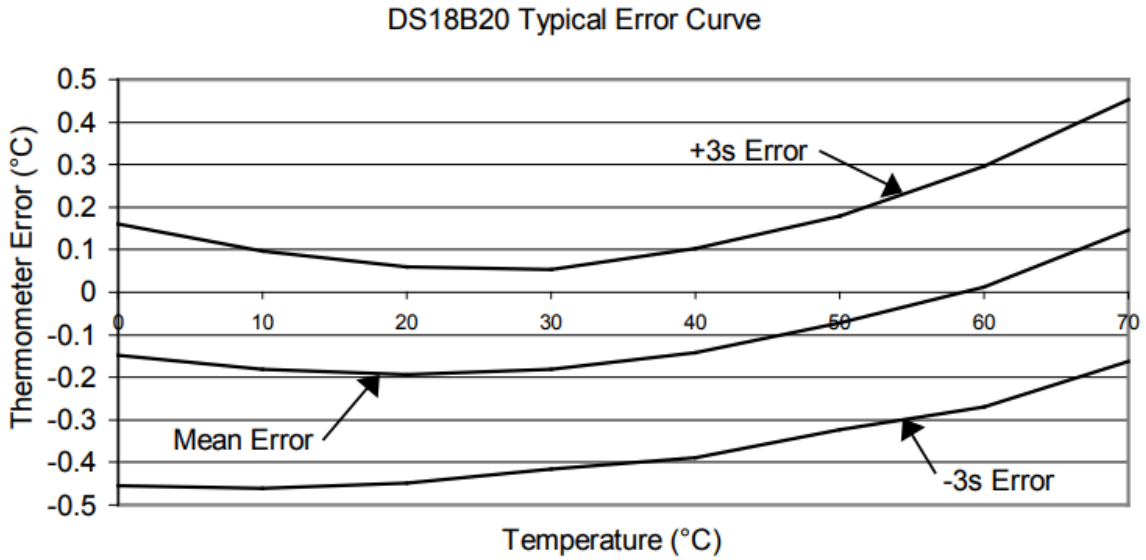
Rysunek 8. Moduł NodeMCU.

2.3.2. Termometr cyfrowy DS18B20

Termometr cyfrowy DS18B20 zapewnia cyfrowy pomiar temperatury w rozdzielczości od 9 do 12 bitów (możliwa regulacja rozdzielczości). Zakres temperatur, dla których możliwe jest stosowanie DS18B20 wynosi od $-55°C$ do $+125°C$, a jego dokładność dla zakresu temperatur $-10°C$ do $+85°C$ wynosi $\pm 0{,}5°C$. Czujnik temperatury komunikuje się z jednostką mikroprocesorową poprzez szynę 1-Wire. Układ może być zasilany z zewnętrznego źródła zasilania bądź pracować w trybie zasilania pasożytniczego (*ang. „parasite power"*), w którym zasilany jest poprzez linię danych.

DS18B20 dzięki posiadaniu unikatowego identyfikatora może być używany w jednym układzie mikroprocesorowym w wielu sztukach, co umożliwia pomiar temperatury z wielu miejsc jednocześnie.

DS18B20 zapewnia stosunkowo dokładny pomiar wartości temperatury dla warunków występujących w normalnym środowisku. Charakterystyka błędu w funkcji temperatury układu scalonego DS18B20 przestawiona została na rysunku 2 (zaczerpnięta z karty katalogowej udostępnianej przez producenta).



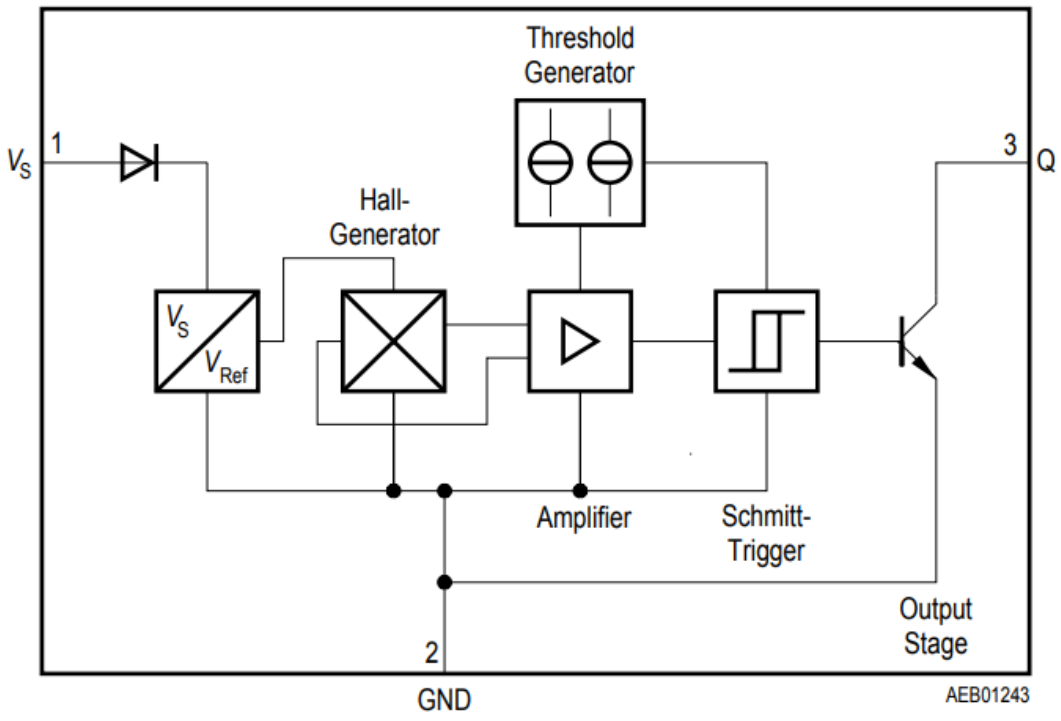
Rysunek 9. Charakterystyka błędu w funkcji temperatury termometru DS18B20.

Koszt cyfrowego termometru DS18B20 wynosi około 6 zł.

2.3.3. Czujnik Halla TLE4905L

Zastosowany w projekcie czujnik Halla TLE4905L jest czujnikiem unipolarnym pola magnetycznego o cyfrowym sygnale wyjściowym. Jego zasada działania jest następująca: Kiedy strumień magnetyczny o określonej polaryzacji przenika przez czujnik i wartości indukcji magnetycznej osiąga wartość 5mT, na czujniki Halla generowany jest stan niski. W przypadku zmniejszenia indukcji pola magnetycznego do wartości poniżej 5mT czujnik Halla generuje stan wysoki.

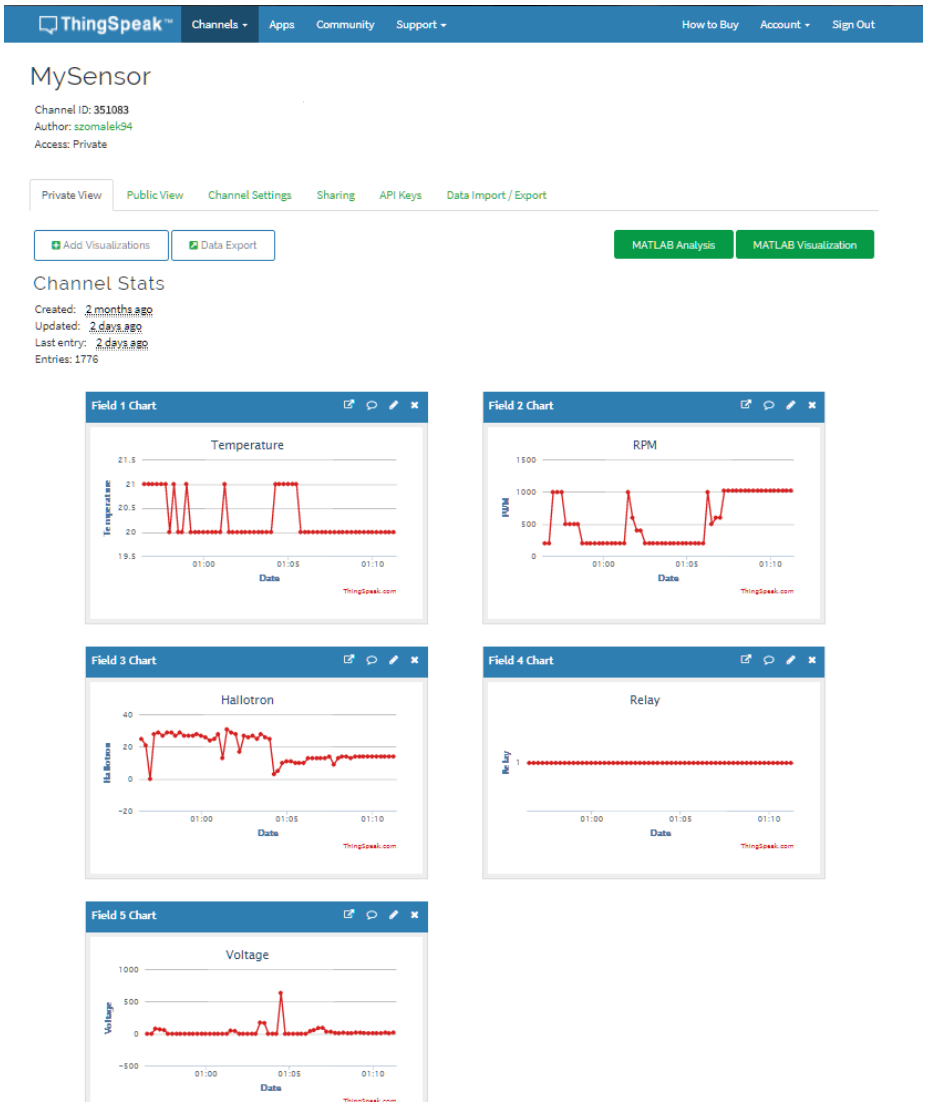
Wewnętrzna budowa czujnika przedstawiona została na rysunku 3 (zaczerpnięto z karty katalogowej producenta). Czujnik zawiera generator Halla, wzmacniacz oraz przełącznik Schmitta. Pole magnetyczne o określonej biegunowości indukuje napięcie na sondzie Halla. Napięcie to jest następnie wzmacniane i powoduje zmianę stanu przełącznika Smitta z otwartym kolektorem. Układ zawiera diodę chroniącą przed zasileniem układu napięciem o ujemnej biegunowości.



Rysunek 10. Budowa wewnętrzna czujnika Halla.

2.3.4. ThingSpeak

ThingSpeak to platforma analityczna, która umożliwia gromadzenie, wizualizację oraz stałą analizę strumienia danych z urządzeń *Internetu Rzeczy*. ThingSpeak jest rozwijane przez firmę MathWorks. Fakt ten pozwala na wykonywanie kodu z programu Matlab (również rozwijanego przez MathWorks) i bezpośrednie przetwarzanie danych. Rysunek 10 przedstawia zrzut ekranu prezentujący kanał wykorzystywany do gromadzenia danych. Urządzenia mogą w darmowej wersji wysyłać dane do kanału co 15 s, zaś limit roczny danych to 3 miliony wiadomości.



Rysunek 11. Zrzut ekranu z kanału ThingSpeak wykorzystanego w projekcie.

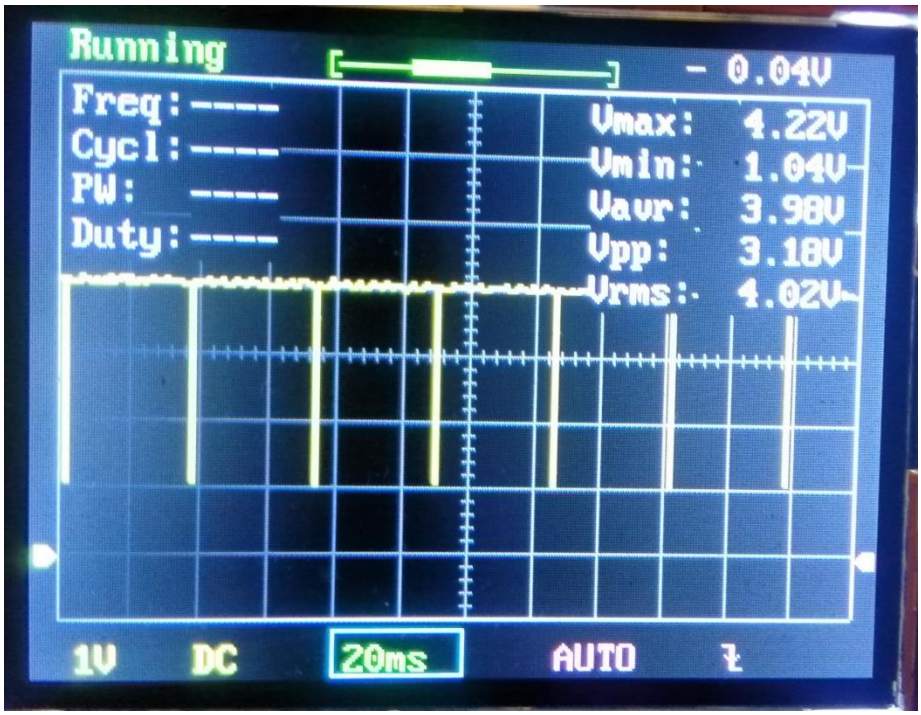
3. OKREŚLENIE PRĘDKOŚCI OBROTOWEJ Z WYKORZYSTANIEM CZUJNIKA HALLA

Zgodnie z informacjami podanymi w podrozdziale 2.3.3 poniższego projektu czujnik Halla po wykryciu pola magnetycznego o odpowiedniej wartości indukcji magnetycznej przechodzi w niski stan logiczny. Okresowa zmiana pola magnetycznego przenikającego czujnik Halla jest realizowana poprzez zamontowanie na jednej z łopatek wentylatora magnesu neodymowego. Realizacja pomiaru prędkości obrotowej przedstawiona została na rysunku 11. Element oznaczony cyfrą 1 to czujnik Halla przymocowany do obudowy za pomocą kleju, 2 to magnes neodymowy powodujący okresową zmianę pola magnetycznego, zaś magnes neodymowy oznaczony cyfrą 3 służy do częściowego wyważenia wentylatora – zamontowanie samego magnesu „2" powoduje duże drgania mechaniczne, które objawiają się również przez hałas.

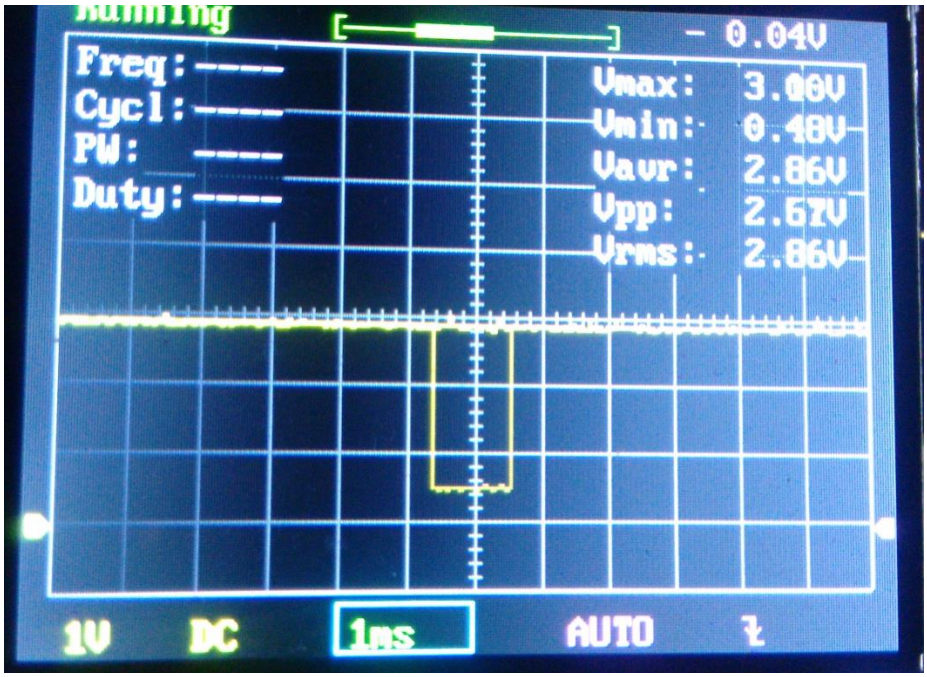
Rysunek 12. Praktyczna realizacja pomiaru prędkości obrotowej.

Na rysunkach 12 i 13 przedstawione zostały oscylogramy napięcia na pinie D5 modułu NodeMCU. Różnica między tymi dwoma oscylogramami polega jedynie na zmianie podstawy czasu.

Zgodnie z wcześniejszymi opisami, moduł NodeMCU reaguje przy wykryciu zbocza opadającego na pinie D5, którego wystąpienie powoduje wykonanie odpowiedniej funkcji obsługującej przerwanie, która inkrementuje wartość licznika obrotów. Zmienna pełniająca rolę licznika obrotów jest wyzerowywana co każdą sekundę. Przy podłączeniu wentylatora do napięcia stałego o wartości 12 V maksymalna liczba obrotów na sekundę wynosi $n = 18\,\frac{obr}{s} = 1080\,\frac{obr}{s}$.



Rysunek 13. Oscylogram napięcia na pinie D5, podstawa czasu 20 ms.



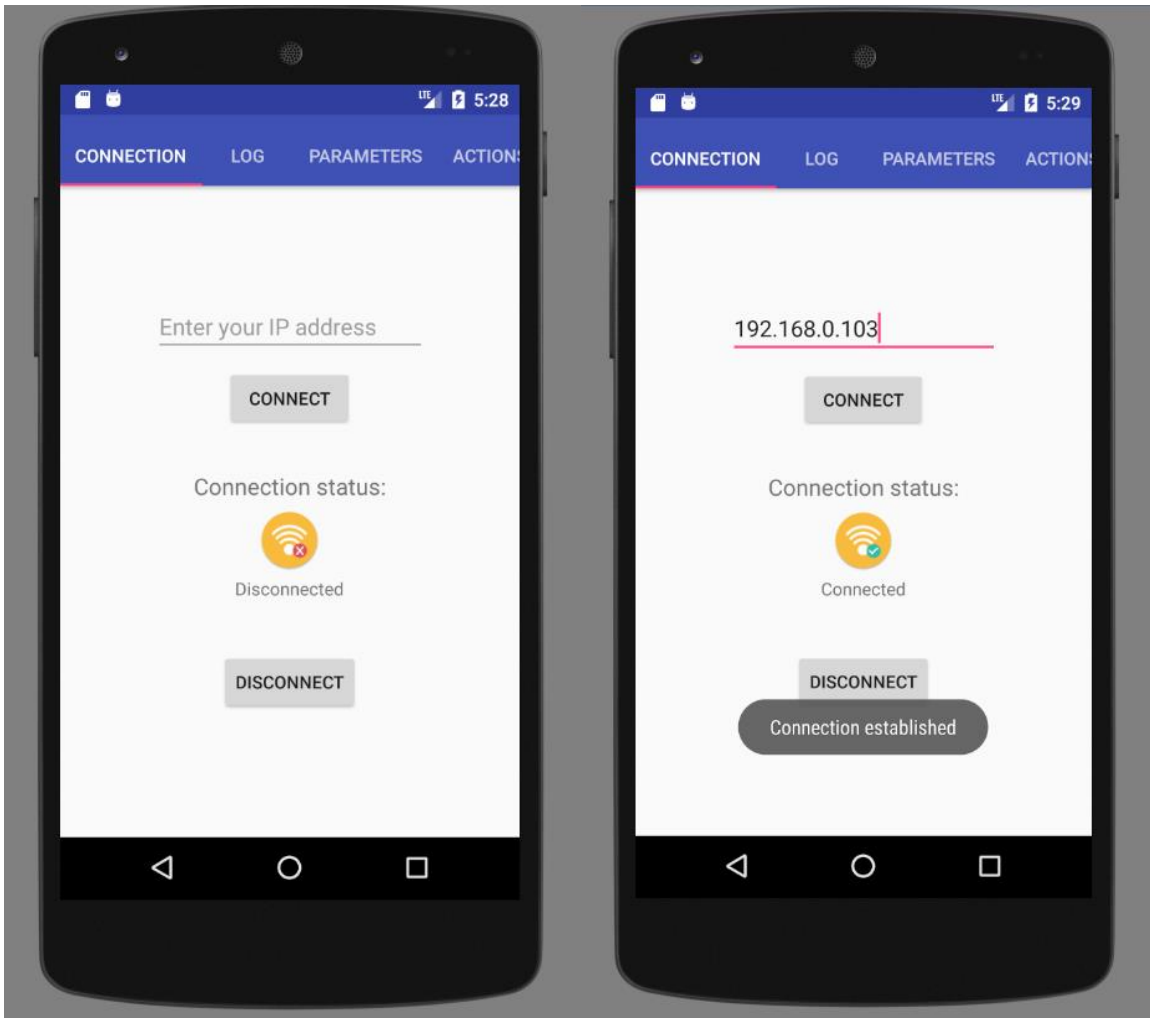
Rysunek 14. Oscylogram napięcia na pinie D5, podstawa czasu 1 ms.

Początkowym zamysłem w projekcie było reagowanie na stan niski na pinie D5 i wprowadzanie opóźnienia (rzędu milisekundy) w celu pozwolenia magnesowi neodymowemu na wyjście spoza obszaru czujnika Halla i na wygenerowanie ponownie stanu wysokiego na czujniku. Takie rozwiązanie angażowałoby w większym stopniu moduł NodeMCU, dlatego zastosowane zostało przerwanie przy wykryciu zbocza opadającego, które stanowi rozwiązanie niemalże idealne dla omawianego przypadku pomiaru prędkości obrotowej.

4. PREZENTACJA DZIAŁAJĄCEGO PROJEKTU

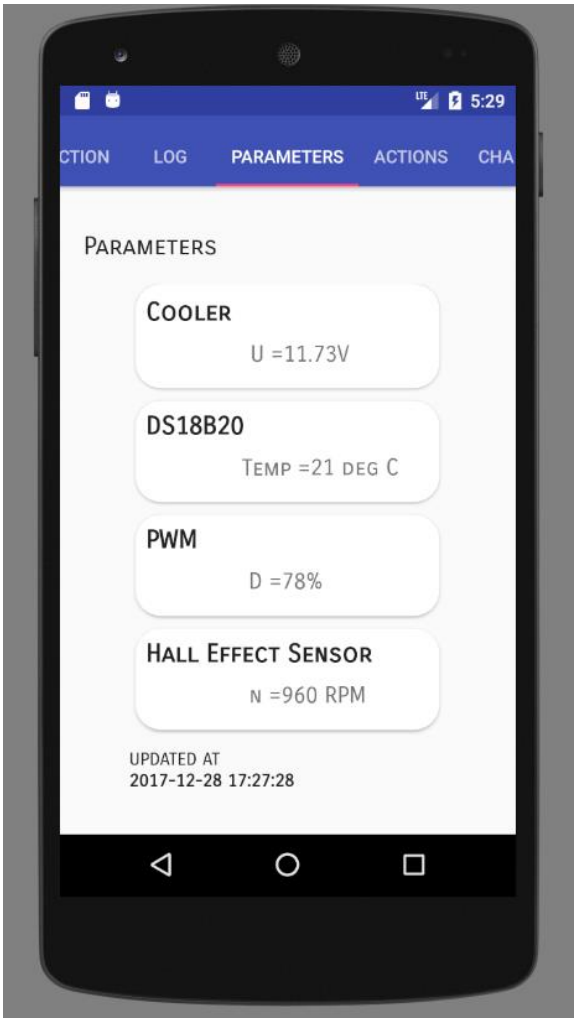
Na poniższych rysunkach zawarte zostały zrzuty ekranu przedstawiające wykonaną aplikację mobilną. Fragmenty opisane w podrozdziale 2.2.3 projektu zostały zgrupowane w pasku na górze ekranu w postaci tak zwanych zakładek. Zmiana zakładki odbywa się poprzez przesunięcie dłonią po ekranie w lewo lub w prawo.

Rysunek 14 przedstawia zakładkę Connection przed i po nawiązaniu połączenia z modułem NodeMCU. W przypadku stworzenia punktu dostępu (tzw. hotspot) na urządzeniu mobilnym należy ustalić adres IP, jaki został nadany modułowi, a następnie wprowadzić do w pole tekstowe. Pomyślne nawiązanie połączenia jest potwierdzone poprzez wyświetlenie odpowiedniego komunikatu i zmianę ikony symbolizującej łączność bezprzewodową.



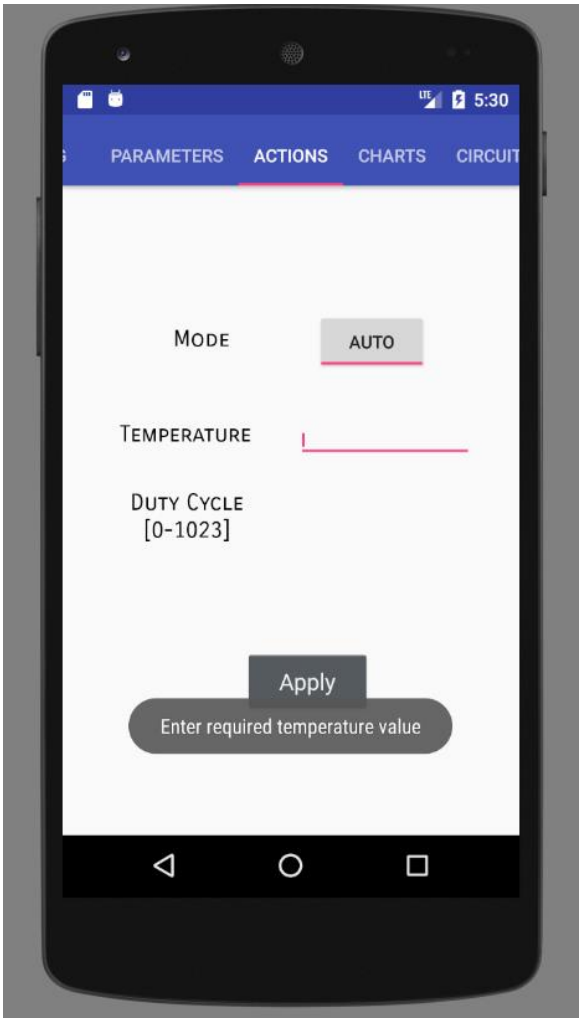
Rysunek 15. Zakładka Connection przed i po nawiązaniu połączenia.

Na rysunku 15 umieszczona została zakładka Parameters, która zawiera informacje takie jak napięcia na zaciskach wentylatora, temperatura wskazywana przez DS18B20, współczynnik wypełnienia sygnału, prędkość obrotową wentylatora, a ponadto moment ostatniej aktualizacji danych w serwerze ThingSpeak.



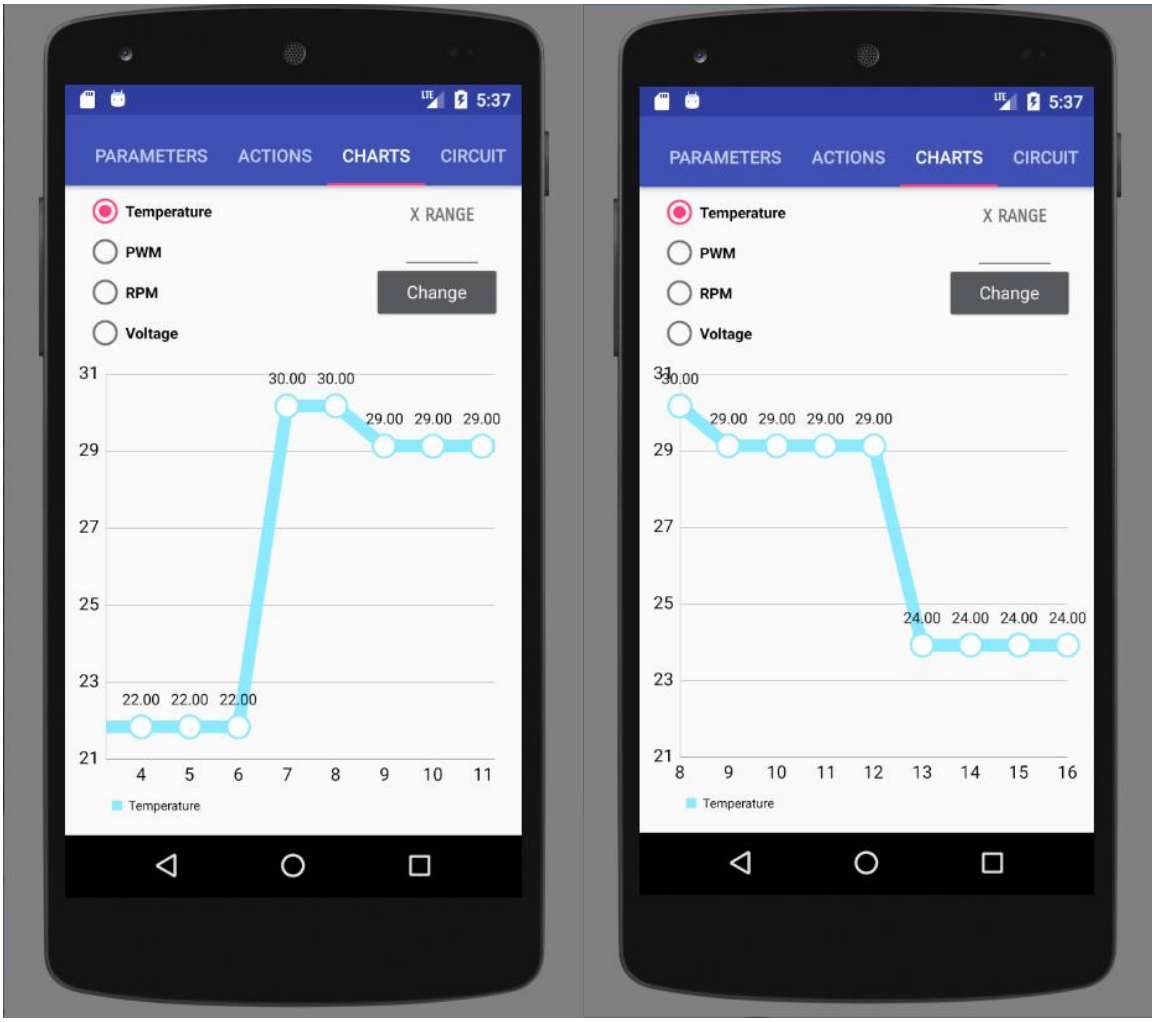
Rysunek 16. Zakładka Parameters.

Zakładka Actions przedstawiona została na rysunku 16. Oferuje ona możliwość regulacji temperatury lub współczynnika wypełnienia sygnału, w zależności od stanu (tj. wciśnięty/niewciśnięty), w którym znajduje się przycisk „Mode”. W celu interakcji z użytkownikiem zastosowane zostały komunikaty proszące o podanie wartości regulowanej, które wyświetlane są po zmianie stanu przycisku.



Rysunek 17. Zakładka Actions.

Aplikacja umożliwia wykreślanie wykresów temperatury, współczynnika wypełnienia, prędkości obrotowej i napięcia w zależności od czasu (dokładniej, w zależności od następujących po sobie danych zgromadzonych w aplikacji od czasu jej uruchomienia). Przykładowy wykres temperatury zamieszczony został na rysunku 17. Zastosowana biblioteka obsługująca wykresy umożliwia przesuwanie wykresu w lewo i prawo, co zostało przedstawione na poniższym rysunku.

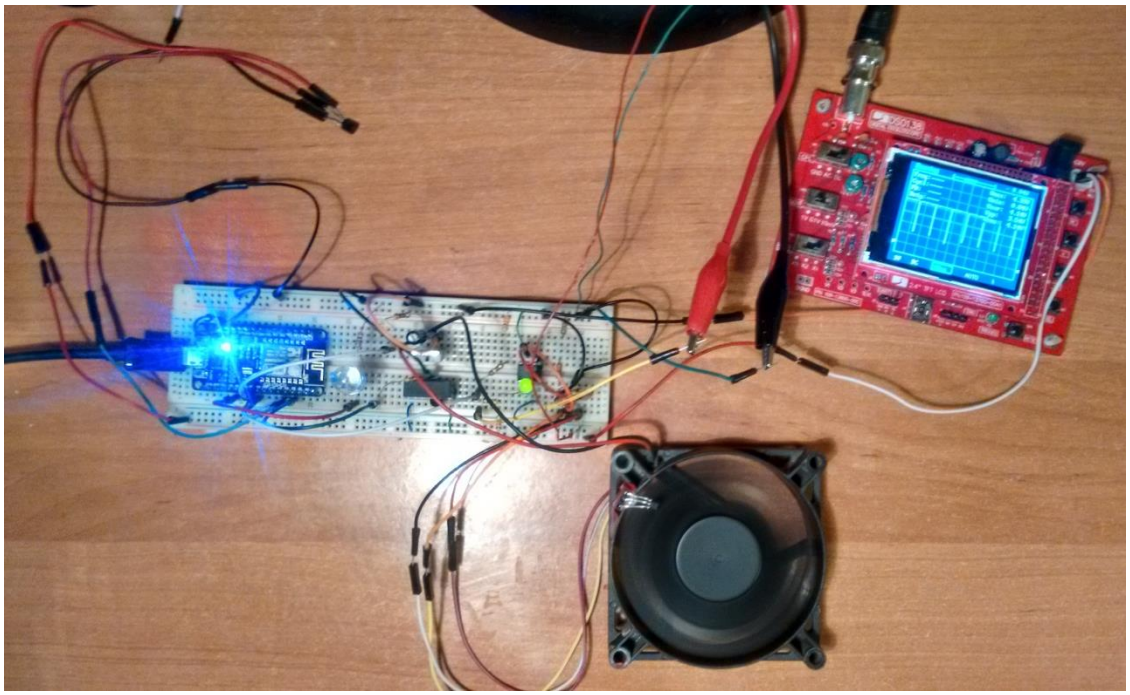


Rysunek 18. Zakładka Charts - przed i po przesunięciu wykresu.

Ostatnią zakładką przeznaczoną dla użytkownika jest Circuit przedstawiony na rysunku 18, na którym zawarte zostały informacje z zakładki Parameters oraz przełącznik umożliwiający sterowanie przekaźnikiem. Łopatki wentylatora przy niezerowej prędkości obrotowej wirują stosownie do wartości prędkości obrotowej, co zostało osiągnięte za pomocą animacji.

Rysunek 19. Zakładka Circuit.

Rysunek 20 przedstawia zmontowany układ będący przedmiotem tego projektu na płytce stykowej. Jak można zauważyć na oscyloskopie, wentylator znajduje się w stanie pracy.



Rysunek 20. Zmontowany układ na płytce stykowej.

5. ZAKOŃCZENIE

Jednym z ważniejszych ograniczeń projektu jest odstęp pomiędzy kolejnymi aktualizacjami danych w serwerze ThingSpeak, który wynosi 15 s w wersji darmowej. Korzystnym rozwiązaniem jest zmniejszenie tego odstępu do np. 5 s lub zastosowanie dwustronnej komunikacji TCP, co jednak posiada jedną wadę – należałoby opracować sposób gromadzenia danych, które potem można by przeglądać i analizować. Gromadzenie danych w rzeczywistych systemach jest ważnym aspektem, ponieważ na podstawie ich analizy możliwe jest wykrycie usterki lub zastanawiające wartości niektórych parametrów (np. skokowa zmiana napięcia na zaciskach wentylatora).

Pomiar prędkości obrotowej może zostać zrealizowany w prosty sposób z wykorzystaniem czujnika Halla i magnesów neodymowych. Rozwiązanie wykorzystujące czujnik Halla są często spotykane w rzeczywistych układach pomiarowych prędkości obrotowej maszyn elektrycznych, jednak zapewniają one zapewne większą dokładność pomiaru niż rozwiązanie przedstawione w projekcie. Zastosowane w projekcie magnesy neodymowe posiadają znaczą masę w stosunku do łopatek wirnika, dlatego koniecznym okazało się ich wyważenie. W rzeczywistych silnikach obciążenie wirnika magnesami nie powinno wprowadzić aż takich zaburzeń jak te zaobserwowane w projekcie.

Cały koszt projektu wyniósł około 50 zł, z czego 40% stanowi cena modułu NodeMCU. Moduł ten pozwala na niezwykle tanią, a zarazem efektywną komunikację bezprzewodową z urządzeniami mobilnymi i na sterowanie instalacją elektryczną.

Projekt przedstawia możliwości zastosowania układów mikroprocesorowych w rzeczywistych układach sterowania i sposoby eliminowania ich ograniczeń – takich jak np. niskie napięcie pracy (5V lub 3,3V) oraz niska obciążalność prądowa wyjść. W rzeczywistych układach rolę przekaźników pełnią styczników, które tak jak w projekcie mogą być sterowane z wykorzystaniem tranzystorów.