

**Politechnika Warszawska**

W Y D Z I A Ł E L E K T R O N I K I  
I T E C H N I K I N F O R M A C Y J N Y C H



Instytut Systemów Elektronicznych  
Zakład Układów i Systemów Elektronicznych

# Praca dyplomowa inżynierska

na kierunku Elektronika  
w specjalności Elektronika i Inżynieria Komputerowa

Raspberry Pi jako zoptymalizowany pod względem kosztu  
sieciowy system akwizycji danych

**Krzysztof Wasilewski**  
nr albumu 265956

promotor  
dr inż. Wojciech Zabołotny

Warszawa 2018



# Raspberry Pi jako zoptymalizowany pod względem kosztu sieciowy system akwizycji danych

## Streszczenie

Praca zawiera opis projektu i implementacji sieciowego systemu akwizycji danych z wykorzystaniem platformy Raspberry Pi. Głównym celem projektu było stworzenie systemu umożliwiającego użytkownikowi akwizycję danych pomiarowych i zachowanie jak najdokładniejszej, w granicach możliwości technicznych, informacji o czasie zebrania próbki. Podstawowym problemem w trakcie projektu było zapewnienie wykonania krytycznych czasowo zadań w wielowątkowym systemie operacyjnym Linux. W pracy przedstawione są możliwości różnych rozwiązań programowych. Projekt zakładał, że użytkownik będzie w stanie ustawiać parametry pomiarów zdalnie za pomocą aplikacji internetowej. Dodatkowymi założeniami były optymalizacja pod względem kosztu i łatwa rozszerzalność projektu. W piątym rozdziale zawarto opis testów sprzętu oraz testów funkcjonalnych i prezentację wyników. Ostatnią częścią pracy jest podsumowanie wraz z ukazaniem dalszych możliwości rozwoju projektu.

**Słowa kluczowe:** system akwizycji danych, Raspberry Pi, sieć



# **Raspberry Pi as a cost-optimized network data aquisition system**

## **Abstract**

Thesis includes project description of the network data aquisition system using Raspberry Pi platform. The main aim of the project was to build a system which can be used to aqisite data and save as much as possible accurate information about measurement timestamp. The main issue was to ensure finishing time-critical tasks in multithreading Linux operating system. Thesis describes variety of software solutions to the problem. Project assumed that the user should be able to control the system via Web application. Another assumption was cost optimazation and developability of the project. In the fifth chapter hardware tests, functional tests and results summary are described. The last part of the thesis is the assumption with capabilities for futher development.

**Keywords:** data aquisition system, Raspberry Pi, netowrk



Warszawa, 29 stycznia 2018

POLITECHNIKA WARSZAWSKA  
WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH

### **OŚWIADCZENIE**

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt. Raspberry Pi jako zoptymalizowany pod względem kosztu sieciowy system akwizycji danych:

- została napisana przeze mnie samodzielnie,
- nie narusza niczych praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektroniki i Technik Informacyjnych.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Krzysztof Wasilewski.....



# Spis treści

Streszczenie . . . . .	i
Abstract . . . . .	ii
<b>1. Wstęp . . . . .</b>	<b>1</b>
1.1. Wprowadzenie . . . . .	1
1.2. Cel i zakres pracy . . . . .	1
1.3. Układ pracy . . . . .	2
<b>2. Założenia . . . . .</b>	<b>3</b>
2.1. Przykłady systemów akwizycji danych . . . . .	3
2.1.1. System akwizycji danych NI USB6008 . . . . .	3
2.1.2. Oszczeloskop cyfrowy Digilent Analog Discovery 2 . . . . .	4
2.2. Cechy systemu . . . . .	5
2.2.1. Informacja o czasie pomiaru . . . . .	5
2.3. Struktura systemu . . . . .	6
2.3.1. Platforma Raspberry Pi . . . . .	7
2.3.2. Płytkę pomiarowa . . . . .	8
2.3.3. Założenia oprogramowania . . . . .	8
2.3.4. Zapewnienie niezawodności systemu . . . . .	8
2.3.5. Protokół sieciowy MQTT . . . . .	9
<b>3. Projekt . . . . .</b>	<b>10</b>
3.1. Płytkę pomiarową . . . . .	10
3.2. Przetwornik analogowo-cyfrowy . . . . .	12
3.2.1. Analiza dostępnych układów . . . . .	12
3.2.2. Przetwornik MAX 1202 . . . . .	12
3.2.3. Przetwornik MCP 3424 . . . . .	13
3.3. Interfejs SPI . . . . .	14
3.4. Interfejs I <sup>2</sup> C . . . . .	14
3.5. Porównanie dostępnych interfejsów . . . . .	15
3.6. Sterowanie pinami GPIO . . . . .	15
3.7. Pozostałe czujniki podłączane do systemu przez magistralę I <sup>2</sup> C . . . . .	15
3.7.1. Moduł z czujnikiem ciśnienia i temperatury LPS25H . . . . .	16
3.7.2. Moduł z czujnikiem wilgotności i temperatury z układem HTS221 . . . . .	16
<b>4. Oprogramowanie . . . . .</b>	<b>18</b>
4.1. System operacyjny . . . . .	18
4.2. Środowisko Buildroot . . . . .	18
4.2.1. Interfejs środowiska . . . . .	19
4.2.2. Toolchain . . . . .	19
4.2.3. Dodawanie modułów jądra . . . . .	19
4.3. Komunikacja z Raspberry Pi . . . . .	20
4.4. Sterowniki urządzeń . . . . .	20
4.4.1. Sterownik przetwornika analogowo-cyfrowego . . . . .	20

4.5.	Protokół synchronizacji czasu . . . . .	21
4.5.1.	Znacznik czasu . . . . .	21
4.6.	Drzewo urządzeń . . . . .	22
4.7.	Konfiguracja modułu watchdog . . . . .	22
4.8.	Aplikacje w przestrzeni użytkownika . . . . .	23
4.8.1.	Aplikacja wykorzystująca moduł <i>spidev</i> . . . . .	23
4.8.2.	Aplikacje wykorzystującą moduł <i>i2cdev</i> . . . . .	23
4.8.3.	Programy publikujące i subskrybujące dane przy pomocy protokołu MQTT . . . . .	23
4.8.4.	Aplikacja webowa . . . . .	24
<b>5.</b>	<b>Testy . . . . .</b>	<b>26</b>
5.1.	Przebieg testów . . . . .	26
5.2.	Użyte narzędzia . . . . .	26
5.3.	Testy funkcjonalne . . . . .	26
5.3.1.	Test interfejsu sieciowego . . . . .	26
5.3.2.	Testy niezawodności systemu . . . . .	27
5.4.	Wyniki pomiarów . . . . .	27
5.4.1.	Charakterystyki napięcia spróbkowanego przetwornikiem ADC MAX1202 . . . . .	27
5.4.2.	Analiza FFT sygnału sinusoidalnego . . . . .	29
5.4.3.	Pomiary ciśnienia czujnikiem LPS25H . . . . .	31
5.5.	Analiza wyników . . . . .	32
5.5.1.	Prezentacja i wizualizacja danych pomiarowych . . . . .	32
<b>6.</b>	<b>Podsumowanie . . . . .</b>	<b>33</b>
<b>Bibliografia . . . . .</b>		<b>35</b>

# 1. Wstęp

## 1.1. Wprowadzenie

W dzisiejszych czasach w dobie Internetu i rozwoju Internetu Rzeczy rośnie zapotrzebowanie na wydajne i zoptymalizowane pod względem kosztu sieciowe systemy akwizycji danych. Rozwój technologii pozwolił na miniaturyzację czujników oraz zmniejszenie kosztu urządzeń zbierających dane pomiarowe.

Akwizycja (zbieranie) danych jest pierwszym i bardzo ważnym etapem przetwarzania danych, gdyż jakość systemu akwizycji wpływa na wydajność i dokładność całego systemu pomiarowego. Istotnym aspektem jest, by odpowiednio dobrą parametry techniczne systemu do konkretnego zastosowania, uwzględniając rodzaj transmisji, poziomy napięć logicznych czy wydajność prądową układu zasilania. Dodatkowo system sieciowy musi używać odpowiedniego do danego zastosowania protokołu transmisji.

Główny nacisk pracy dyplomowej był położony na zachowanie jak najdokładniejszej informacji o czasie pomiaru i okresie próbkowania. Dodatkowo system powinien być zoptymalizowany pod względem kosztu. Użycie platformy Raspberry Pi daje dosyć duże możliwości, jednak ma też swoje ograniczenia. Pierwszym i jednocześnie największym sprzętowym ograniczeniem, wynikającym z zastosowania tej platformy jest brak przetwornika analogowo-cyfrowego na płytce. W celu obsługi sygnałów analogowych konieczne jest zastosowanie zewnętrznego układu. Parametry techniczne płytki Raspberry Pi oraz interfejsy wymiany danych jakie posiada pozwalają na podłączenie i pracę wielu układów peryferyjnych jednocześnie.

## 1.2. Cel i zakres pracy

Celem pracy było wykonanie zoptymalizowanego pod względem kosztu sieciowego systemu akwizycji danych z wykorzystaniem platformy Raspberry Pi. Projekt zakładał stworzenie płytki pomiarowej będącej rozszerzeniem do platformy Raspberry Pi umożliwiającej zbieranie danych z zewnętrznych urządzeń przy pomocy dostępnych interfejsów SPI i I<sup>2</sup>C. Płytki powinna zawierać układy umożliwiające doprowadzenie sygnałów z zewnętrznych czujników. Sieciowy system akwizycji danych może być wykorzystany do stworzenia stacji meteorologicznej z funkcją zdalnego dostępu, możliwością sterowania i wyzwalania pomiarów oraz odbioru danych przez interfejs sieciowy.

W zakres pracy wchodziło:

- Opracowanie sterowników jądra i aplikacji użytkownika umożliwiających obsługę przetworników i czujników pomiarowych podłączonych do komputera przez typowe interfejsy sprzętowe, takie jak I<sup>2</sup>C, SPI
- Zapewnienie akwizycji danych z zachowaniem jak najpełniejszej (w granicach możliwości technicznych) informacji o czasie pomiaru, w celu umożliwienia późniejszej analizy off-line danych z różnych źródeł z uwzględnieniem korelacji czasowej.
- Przygotowanie serwerów i/lub programów klienckich umożliwiających udostępnienie zmierzonych danych przez sieć TCP/IP.
- Realizacja interfejsu pozwalającego na zdalne sterowanie procesem pomiaru.

Podstawowym celem projektu było zaprojektowanie i realizacja systemu umożliwiającego użytkownikowi akwizycję danych pomiarowych, z zachowaniem jak najdokładniejszej informacji o znaczniku czasu w momencie zebrania próbki. W wielozadaniowym systemie operacyjnym zastosowanie aplikacji z przestrzeni użytkownika nie daje pewności co do reżimu czasowego wykonania poleceń zawartych w kodzie programu. W przypadku akwizycji danych z zewnętrznych przetworników podłączonych przez magistralę interfejsu szeregowego (np. *spidev* dla interfejsu SPI) informacja o znaczniku czasu zarejestrowana w aplikacji z przestrzeni użytkownika nie jest dokładna i powoduje powstanie nieregularności okresu próbkowania. W celu zapisania jak najdokładniejszej informacji o czasie pomiaru należy znacznik czasu pobrania próbki zarejestrować w przestrzeni jądra systemu.

W pracy przedstawione są możliwości różnych rozwiązań programowych. Niski koszt zbudowania systemu oraz łatwość rozszerzenia funkcjonalności powodują, że urządzenie stanowi konkurencję dla systemów komercyjnych. Projekt zakładał, że użytkownik będzie w stanie ustawiać parametry pomiarów zdalnie za pomocą aplikacji w przeglądarce internetowej. Dodatkowymi założeniami były optymalizacja pod względem kosztu i łatwa rozszerzalność systemu. W końcowej części pracy zawarto przebieg testów sprzętu oraz testów funkcjonalnych i podsumowanie wyników.

### 1.3. Układ pracy

Praca została podzielona na sześć rozdziałów. Po niniejszym rozdziale wstępny, w Rozdziale 2 dokonano przeglądu systemów akwizycji danych, przedstawiono założenia i wymagania postawione dla sieciowego systemu akwizycji danych oraz dla oprogramowania umożliwiającego obsłuszenie urządzeń dołączanych do platformy Raspberry Pi. W Rozdziale 3 przedstawiono projekt układu pozwalającego na realizację zaprojektowanego systemu akwizycji danych. Rozdział 4 zawiera opis różnych rozwiązań programowych. W Rozdziale 5 przedstawiony jest przebieg oraz wyniki testów systemu. W Rozdziale 6 zawarto podsumowanie projektu.

## 2. Założenia

### 2.1. Przykłady systemów akwizycji danych

Na rynku jest dostępna szeroka gama systemów akwizycji danych. Większość dostępnych rozwiązań to układy do profesjonalnych zastosowań laboratoryjnych. Często urządzenia te pracują tylko pod określonym oprogramowaniem co zwiększa koszty. Powoduje to, że brakuje urządzeń, które posiadałyby podstawową funkcjonalność systemu akwizycji danych za cenę mniejszą niż 500zł.

#### 2.1.1. System akwizycji danych NI USB6008

Przykładem jest system akwizycji danych National Instruments USB6008. Urządzenie posiada 8 analogowych wejść i obsługuje interfejs USB. Zawiera 12-bitowy przetwornik analogowo-cyfrowy próbujący z częstotliwością 10kS/s.[1] Urządzenie kosztuje około 1000zł w celu wykonania pomiarów potrzebujemy komputera z zainstalowanym oprogramowaniem producenta, co powoduje generowanie dodatkowych kosztów.



Rysunek 2.1. National Instruments USB6008

### 2.1.2. Oszkloskop cyfrowy Digilent Analog Discovery 2

Kolejnym przykładem systemu akwizycji danych jest przystawka do komputera osobistego Digilent Analog Discovery 2. Urządzenie po podłączeniu do komputera PC oraz doprowadzeniu badanych sygnałów do jego wejść umożliwia wykonywanie pomiarów. Posiada następujące parametry techniczne[3]

- 16-kanałowy analizator logiczny (14-bit, 100MS/s, +-5V)
- 2-kanałowy oscylскоп cyfrowy (14-bitowy, 100MS/s, +-25V)
- wzmacniacz audio-stereo,
- interfejsy SPI, I<sup>2</sup>C, UART, USB
- zasilanie 5V



Rysunek 2.2. Digilent Analog Discovery 2

Cena urządzenia jest zbliżona do poprzedniego przykładu (ok. 1000-1400zł), jednak oprogramowanie, pozwalające na pracę z przystawką jest darmowe, co znaczaco obniża koszt całego systemu. Nie mniej jednak sama przystawka Digilent Analog Discovery 2 nie tworzy w pełni funkcjonalnego systemu akwizycji danych. Do poprawnego działania systemu potrzebny jest komputer osobisty z zainstalowanym dedykowanym oprogramowaniem.

Powysze urządzenia nie posiadają interfejsu sieciowego. Zapewniają jedynie możliwość akwizycji danych na podłączonym przez port USB komputerze.

## 2.2. Cechy systemu

System powinien udostępniać możliwość obsługi sygnałów zarówno cyfrowych jak i analogowych. Ze względu na brak przetwornika analogowo-cyfrowego na płytce Raspberry Pi, konieczne jest użycie zewnętrznego układu. Raspberry Pi może działać pod systemem operacyjnym Linux, który nie jest systemem czasu rzeczywistego, co powoduje, że w domyślnej konfiguracji systemu nie da się zagwarantować stałego odstępu czasowego pomiędzy odbieraniem próbek kolejnych z przetwornika. W celu wydajnego przetwarzania sygnałów większej częstotliwości dodano moduł do jądra systemu, zawierający implementację licznika wysokiej rozdzielczości, co powoduje zmniejszenie wahania częstotliwości próbkowania w trakcie pomiarów oraz pozwala na zachowanie momentu otrzymania próbki z dużą dokładnością.

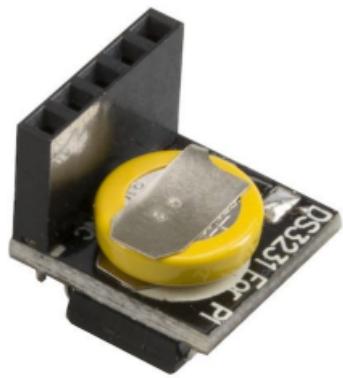
### 2.2.1. Informacja o czasie pomiaru

Raspberry Pi jest komputerem zoptymalizowanym pod względem kosztów i rozmiaru, co sprawia, że jest pozbawiony podzespołów, które są zbyt drogie lub zajmują zbyt dużo miejsca na płytce. Jednym z takich układów jest układ RTC (z ang. Real Time Clock - zegar czasu rzeczywistego). Układ ten jest zasilany z baterii niezależnie od komputera i zapewnia przechowywanie czasu rzeczywistego po odłączeniu zasilania.

System akwizycji danych powinien zapewniać w miarę możliwości technicznych jak najdokładniejszą informację o czasie zebrania próbki danych. Raspberry Pi za pomocą protokołu NTP (z ang. Network Time Protocol - Sieciowy Protokół Czasu) jest w stanie synchronizować czas w systemie do wybranego w konfiguracji protokołu serwera. Korzystając z odpowiednich serwerów, NTP jest w stanie dostosować czas systemu klienta, uwzględniając opóźnienia w transmisji danych. Protokół NTP umożliwia synchronizację z rozdzielczością rzędu nanosekund, jednak posiada ograniczoną dokładność rzędu kilkudziesięciu mikrosekund zależną od różnych czynników, m.in. od precyzji źródła czasu.

Wykorzystanie własnego sterownika do pomiaru napięcia przetwornika analogowo-cyfrowego pozwala na zebranie znacznika czasu z rozdzielczością pojedynczych nanosekund.

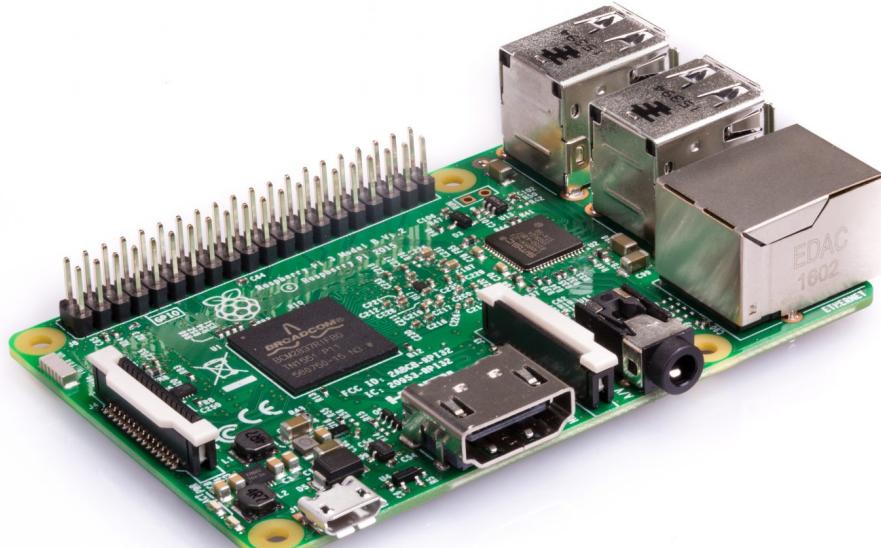
W przypadku, wykorzystania Raspberry Pi do pracy z ograniczonym dostępem do sieci warto dołączyć moduł zegara czasu rzeczywistego. Istnieje możliwość rozszerzenia systemu o moduł RTC. Na rynku są dostępne moduły obsługiwane za pomocą interfejsów szeregowych SPI lub I<sup>2</sup>C. Jednym z najbardziej popularnych modułów zegara czasu rzeczywistego kompatybilnych z Raspberry Pi jest moduł z układem DS3231[8]. Układ zawiera niezależne zasilanie baterijne i działa po odłączeniu Raspberry Pi od źródła napięcia. Moduł RTC z zasilaniem baterijnym przedstawiono na Rys. 2.3.



Rysunek 2.3. Moduł zegara czasu rzeczywistego z układem DS2331

### 2.3. Struktura systemu

System składa się z platformy Raspberry Pi używanej jako komputer główny oraz dołączanej płytki pomiarowej umożliwiającej podłączenie czujników z możliwością łatwego doprowadzenia zasilania do wszystkich urządzeń. Platforma Raspberry Pi zapewnia w systemie kontrolę pomiarów i pozwala na wizualizację oraz udostępnianie otrzymanych wyników.



Rysunek 2.4. Raspberry Pi 3 Model B

### 2.3.1. Platforma Raspberry Pi

Raspberry Pi to jednopłytkowa platforma stworzona przez Raspberry Pi Foundation. Pierwsza wersja minikomputera powstała w roku 2012 i była oparta o układ firmy Broadcom BCM2835 SoC (z ang. System on Chip - System wbudowany) oraz procesor graficzny Broadcom VideoCore IV. Głównymi zaletami platformy były niska cena w stosunku do możliwości.

Rozważano zastosowanie w pracy jednej z trzech wersji płytka:

- Raspberry Pi ZeroW
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B

Wersja płytki	ZeroW	2 Model B	3 Model B
Częstotliwość taktowania procesora	1GHz	1,2GHz	900MHz
Pamięć RAM	512MB	1GB	1GB
moduł WiFi	2,4GHz	brak	2,4GHz
złącze Ethernet	brak	10/100	10/100

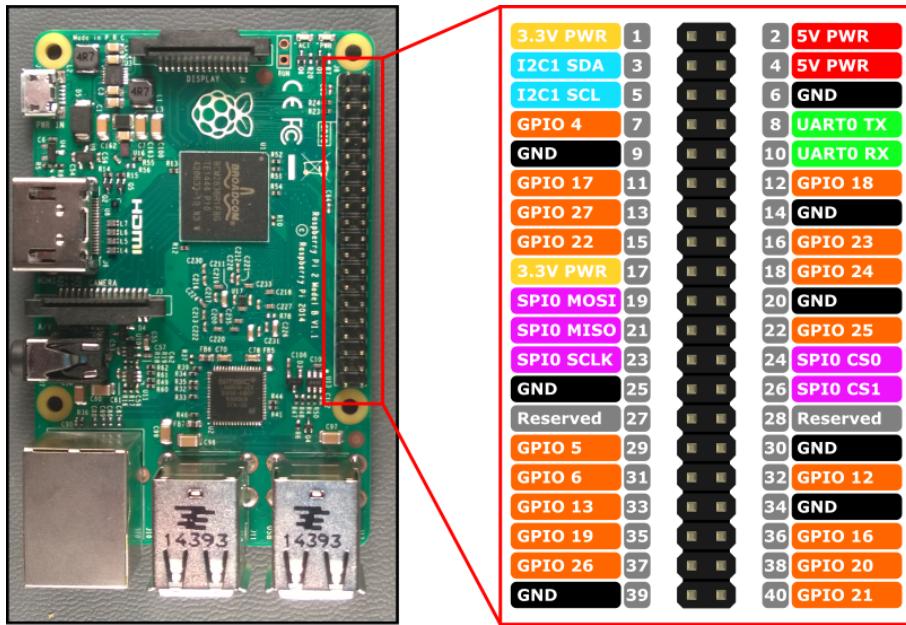
Tablica 2.1. Parametry techniczne różnych wersji płytka Raspberry Pi [5]

Ze względu na małe różnice w cenie i założenie zastosowania bezprzewodowego dostępu do sieci zrezygnowano z wersji Model 2 B. Rozważano zastosowanie wersji ZeroW, która jest cenowo konkurencyjna do wersji Model 3 B i pozwala na bezprzewodowy dostęp do sieci za pomocą modułu sieci Wi-Fi. Z powodu założenia przetwarzania dużej ilości danych przez system, wspólnie wykonywania obsługi czujników w równocześnie z obsługą protokołu sieciowego oraz konieczność instalacji serwera HTTP wybrano wersję Model 3B, której parametry techniczne powinny pozwolić na spełnienie postawionych założeń.

Platforma Raspberry Pi w wersji 3 posiada liczne złącza m.in.: HDMI, Ethernet, 4 porty USB 2.0, wyjście 3,5mm jack, złącze kart microSD oraz 40-pinowe złącze GPIO. W tabeli poniżej przedstawiono specyfikacje wersji płytka użytej w projekcie - Raspberry Pi 3 Model B:

Nazwa układu SoC	Broadcom BCM2837
Procesor CPU	1,2 GHz quad-core ARM Cortex-A53 (64-bit)
Procesor graficzny GPU	Broadcom VideoCore IV
Pamięć RAM	1 GB LPDDR2 (900 MHz)
Napięcie zasilania	5 V, złącze MicroUSB lub GPIO
Połączenia sieciowe	10/100 Ethernet, Wi-Fi (2,4 GHz 802.11n)
Nośnik danych	złącze kart MicroSD
Interfejsy szeregowe	SPI, I <sup>2</sup> C, UART

Tablica 2.2. Szczegółowe parametry techniczne platformy Raspberry Pi 3 Model B [5]



Rysunek 2.5. Rozkład pinów GPIO na płytce Raspberry Pi

### 2.3.2. Płytkę pomiarowa

Płytkę pomiarową, stanowiącą rozszerzenie do Raspberry Pi, miała umożliwiać podłączenie czujników do minikomputera oraz doprowadzenie do nich zasilania i zostało wykonana na potrzeby pracy dyplomowej. Zastosowanie przetwornika analogowo-cyfrowego pozwala na zbieranie danych z czujników analogowych. Czujniki cyfrowe podłączone przez magistralę SPI lub I<sup>2</sup>C powinny być wydajnie obsługiwane przez system.

### 2.3.3. Założenia oprogramowania

Oprogramowanie powinno umożliwiać przeprowadzenie wydajnej akwizycji, bez utraty danych. Niezbędne jest skorzystanie z odpowiednich bibliotek i funkcji udostępnianych przez jądro systemu Linux. Aplikacje użytkownika powinny zapisywać dane do pliku w formacie umożliwiającym późniejsze przetworzenie przez aplikację obsługującą protokół sieciowy MQTT (z ang. MQ Telemetry Transport) i aplikację webową umożliwiającą wyświetlanie danych i ustawienie konfiguracji parametrów pomiaru uruchamianą przez użytkownika w przeglądarce internetowej. Aplikacja udostępnia interfejs pozwalający na zdalne uruchomienie programów obsługujących zbieranie danych z czujników na Raspberry Pi. Oprogramowanie powinno posiadać cechy przenośności tak, aby mogło działać na innych systemach wbudowanych z systemem operacyjnym Linux.

### 2.3.4. Zapewnienie niezawodności systemu

Niezawodność działania systemu jest bardzo istotna w przypadku sieciowego systemu akwizycji danych. Urządzenie jest przystosowane do pracy autonomicznej, więc aby zdalny dostęp mógł być zapewniony, Raspberry Pi musi być w stanie wykrywać stan zawieszenia systemu,

gdyż wpływa to bezpośrednio na niezawodność i ciągłość działania. W komputerze Raspberry Pi istnieje możliwość uaktywnienia układu *watchdog*, zapewniającego automatyczny restart systemu w przypadku jego zawieszenia.

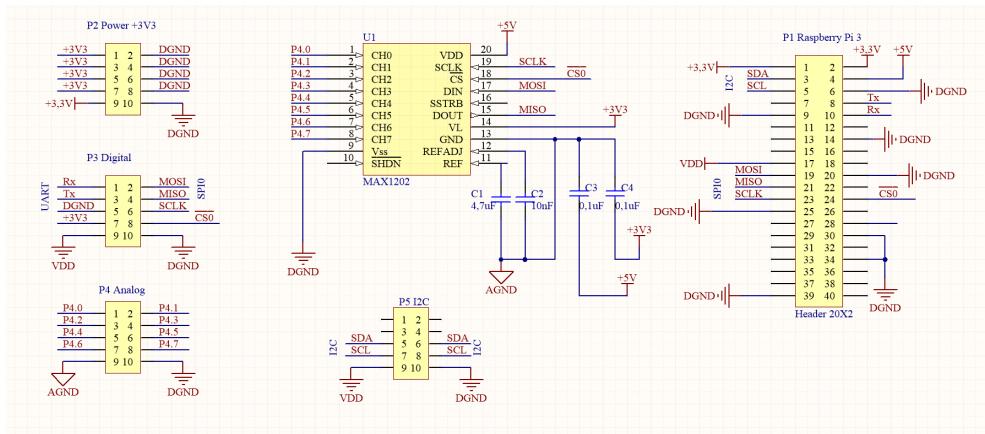
### **2.3.5. Protokół sieciowy MQTT**

Protokół MQTT jest opartym na publikacji i subskrypcji lekkim protokołem sieciowym warstwy TCP/IP[2]. Jest przeznaczony do transmisji wiadomości pomiędzy urządzeniami niewymagającymi dużej przepustowości. Do działania protokołu potrzebny jest broker (z ang. pośrednik), który pełni rolę serwera publikującego wiadomości. Urządzenie, zawierające dane do wysłania publikuje wiadomość podając nazwę tematu. Klient, który chce odebrać te dane zapisuje się na subskrypcję danego tematu również podając jego nazwę. Protokół dobrze sprawdza się w komunikacji urządzeń IoT (z ang. Internet of Things - Internetu Rzeczy).

# 3. Projekt

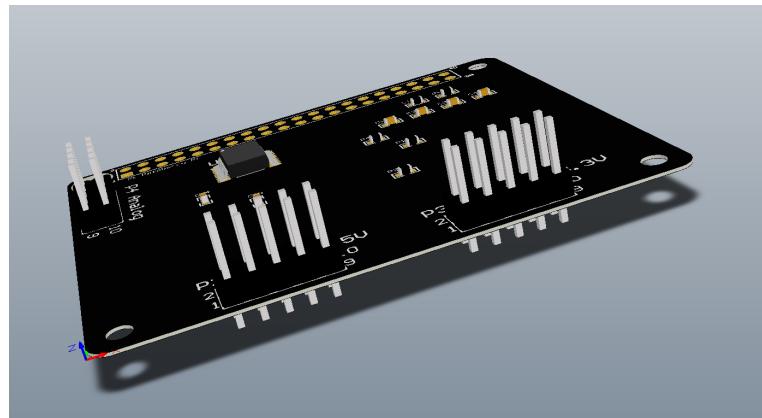
## 3.1. Płytki pomiarowa

Płytki pomiarowa składa się z dwóch przetworników ADC podłączonych przez odpowiednie interfejsy do wejść komputera Raspberry Pi. Zasilanie układów jest doprowadzone z wyjść zasilających Raspberry Pi. W celu wytłumienia zakłóceń dodano elementy pasywne o odpowiednich wartościach. Sygnały do przetwornika z oscyloskopu doprowadzono przy pomocy kabli BNC i odpowiednich końcówek. Na Rys. 3.1 przedstawiono schemat płytka zawierającej przetwornik analogowo-cyfrowy MAX1202 oraz wyjścia cyfrowe i analogowe do podłączenia zewnętrznych czujników. Wyprowadzono także wejścia interfejsu UART w celu komunikacji z komputerem.



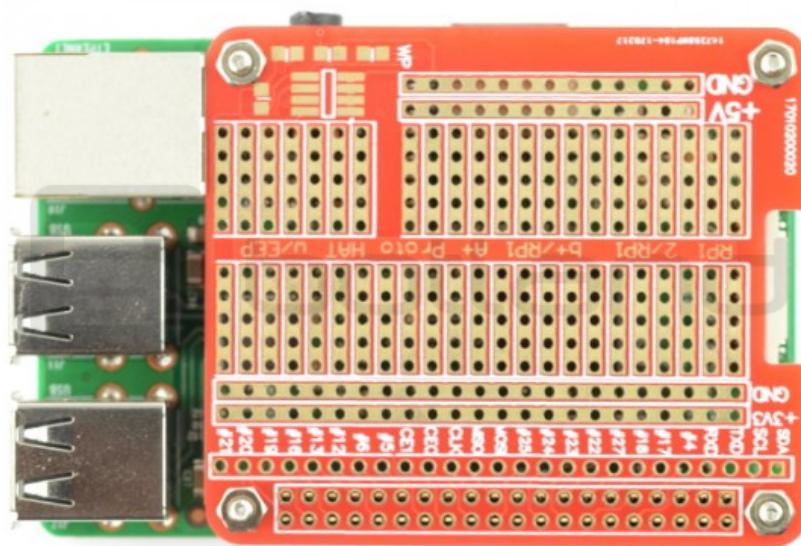
Rysunek 3.1. Schemat płytki dołączanej do Raspberry Pi

Na Rys.3.2 trójwymiarowy wygląd płytki wygenerowany w programie Altium PCB Designer.



Rysunek 3.2. Trójwymiarowy wygląd projektu płytki

Ze względu na ograniczenia czasowe i optymalizację projektu pod względem kosztu podjęto decyzję zmontowania układu na płytce prototypowej. Wygląd płytki bez zamontowanych elementów przedstawiono na Rys 3.3



Rysunek 3.3. Płytnka prototypowa dołączana do Raspberry Pi 2/3

## 3.2. Przetwornik analogowo-cyfrowy

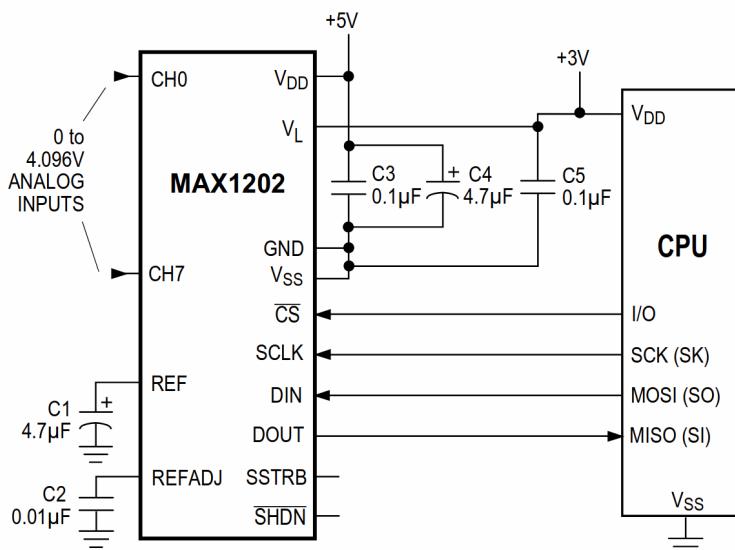
### 3.2.1. Analiza dostępnych układów

Przy wyborze przetwornika zwrócono uwagę na następujące parametry:

- cena układu,
- rozdzielczość,
- częstotliwość próbkowania,
- interfejs komunikacji,
- zasilanie i poziomy logiczne

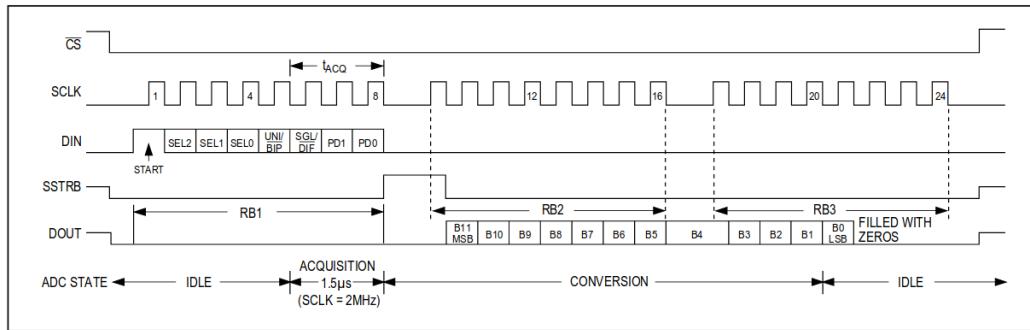
### 3.2.2. Przetwornik MAX 1202

Jednym z wybranych układów był 12-bitowy, 8-kanałowy przetwornik MAX 1202 o próbkowaniu 133 kS/s. Komunikacja z układem odbywa się przez interfejs szeregowy SPI, a poziomy logiczne mogą być ustalone na wartość w zakresie 2,7V - 5,5V w zależności od napięcia podanego na pin VL. Do komunikacji z Raspberry Pi potrzebne są poziomy 3,3V, więc należy podać napięcie 3,3V na odpowiedni pin przetwornika ADC. Układ może być zasilany napięciem 5V. Podłączenie przetwornika analogowo cyfrowego przez interfejs SPI do platformy Raspberry Pi przebiegało jak na Rys. 3.4. Przetwornik zasilono korzystając z jednego z wyjść +5V dostępnych na płytce. Korzystając z zaleceń producenta zawartych w nocy katalogowej [6] podłączono kondensatory tłumiące niekorzystne dla dokładności pomiaru zakłócenia.



Rysunek 3.4. Schemat podłączenia przetwornika MAX1202 do systemu

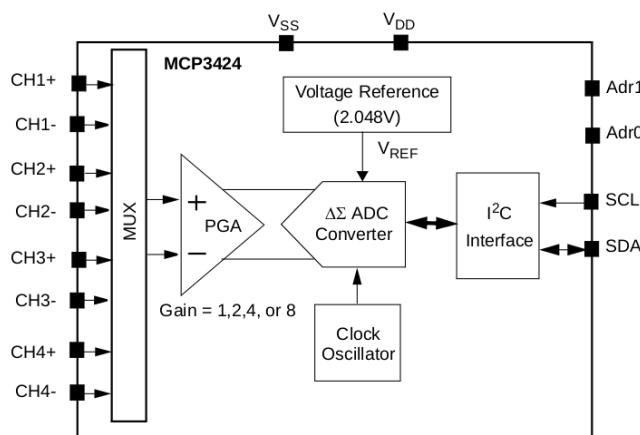
Na Rys. 3.5 przedstawiono przebiegi czasowe napięć na poszczególnych liniach interfejsu w trakcie jednej konwersji napięcia. Jak widać transmisja jednej próbki napięcia odbywa się po przesłaniu jednego bajtu konfiguracyjnego wysłanego z układu typu Master. Przetwornik dokonuje konwersji napięcia analogowego z wybranego kanału i wysyła dwa bajty zawierające dane. Z wykresu odczytać można parametr *ACQUISITION* wynoszący  $1,5 \mu\text{s}$ .



Rysunek 3.5. Wykres czasowy konwersji przetwornika ADC MAX1202

### 3.2.3. Przetwornik MCP 3424

Drugi testowany układ to 18-bitowy 4-kanałowy przetwornik MCP 3424. Przetwornik komunikuje się z systemem za pomocą interfejsu I<sup>2</sup>C i posiada wbudowany wzmacniacz programowalny (PGA - z ang Programmable Gain Amplifier). W układzie dostępne jest również źródło napięcia referencyjnego 2.048V oraz oscylator.



Rysunek 3.6. Schemat funkcjonalny układu przetwornika MCP3424

Nazwa przetwornika	MAX1202	MCP3424
Typ przetwornika	SAR	Sigma Delta
Ilość wejść	8 (4 różnicowe)	4 różnicowe
Interfejs komunikacji	SPI	I2C
Maksymalna częstotliwość próbkowania	133 kS/s	240 SPS (12bit)
Cena(w serwisie farnell.com [PLN])	49,63	16,42
Wzmacniacz programowalny	brak	x1, x2, x4 lub x8

Tablica 3.1. Parametry techniczne wybranych przetworników

### 3.3. Interfejs SPI

Interfejs szeregowy SPI (z ang. Serial Peripheral Interface) jest powszechnie wykorzystywany synchronicznym interfejsem do komunikacji pomiędzy systemem mikroprocesorowym, a układami peryferyjnymi. Interfejs pozwala na podłączenie tylko jednego układu typu Master komunikującego się z układami typu Slave w trybie full-duplex. Urządzenia podłączone do jednej magistrali muszą mieć jednakowe poziomy logiczne.

Transmisja odbywa się za pomocą czterech linii:

- MISO (Master In Slave Out),
- MOSI (Master Out Slave In),
- SCLK (Serial Clock),
- CS (Chip Select).

Przetwornik, którego użyto jest w stanie komunikować się przy częstotliwości taktowania zegara do 2MHz. Urządzenie zostało podłączone do Raspberry Pi jako układ typu Slave. Sygnał zegara jest podawany z układu typu Master, więc w celu uniknięcia błędów transmisji należy ustawić parametr maksymalnej częstotliwości zegara w sterowniku.

### 3.4. Interfejs I<sup>2</sup>C

Interfejs I<sup>2</sup>C (również IIC - Inter Integrated Circuit - z ang. pomiędzy układami scalonymi) to synchroniczny interfejs szeregowy zaprojektowany przez firmę Philips. Stosowany jest do komunikacji z urządzeniami nie wymagającymi dużych szybkości transmisji. Interfejs pozwala na podłączenie do magistrali wiele urządzeń typu Master oraz typu Slave. Wszystkie nadajniki posiadają wyjścia typu otwarty kolektor co powoduje, że poziomy logiczne napięć stanu wysokiego układów podłączonych do jednej magistrali mogą się różnić. Urządzenia komunikujące się za pomocą tego interfejsu mają z góry przypisany adres i są podłączone z resztą urządzeń za pomocą dwóch linii:

- SDA (Serial Data Line),
- SCL (Serial Clock Line).

Interfejs I<sup>2</sup>C nie gwarantuje stałego czasu transmisji, więc korzyści wynikające z implementacji obsługi w kodzie sterownika są mniejsze. Biorąc to pod uwagę założono, że przetworniki podłączane do systemu za pomocą magistrali I<sup>2</sup>C będą służyły do pomiarów z niską częstotliwością próbkowania.

### 3.5. Porównanie dostępnych interfejsów

Porównanie zostało wykonane pod kątem zastosowania do połączenia układów w systemie akwizycji danych. W tabeli 3.1 przedstawiono zestawienie parametrów technicznych standardów transmisji interfejsów SPI oraz I<sup>2</sup>C.

Nazwa interfejsu	SPI	I <sup>2</sup> C
Ilość linii sygnałowych	$3 + n$ (ilość urządzeń)	2
Maksymalna częstotliwość zegara	125MHz	3,2MHz
Adresacja urządzeń	linia CS	10-bitowa
Ilość układów typu Master	1	może być wiele
Rodzaj transmisji	full-duplex	half-Duplex

Tablica 3.2. Parametry transmisji interfejsów SPI i I<sup>2</sup>C

### 3.6. Sterowanie pinami GPIO

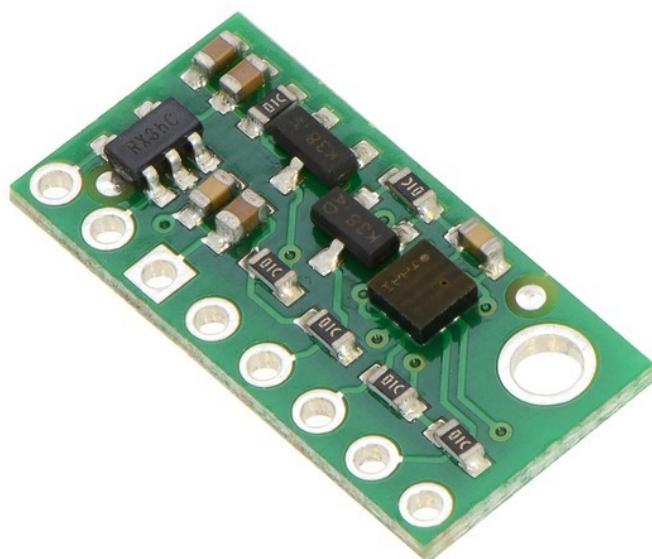
System akwizycji danych powinien dawać możliwość podłączenia czujników i wyzwolenia pomiaru. Aby zapewnić rozszerzalność systemu o kolejne moduły użytkownik powinien mieć możliwość sterowania pinami GPIO, w celu ustawienia konfiguracji dołączonego urządzenia, bądź wyzwolenia pomiaru. Pakiet pythona *RPi.GPIO* udostępnia przystępne API pozwalające niewielkim wysiłkiem sterować pinami GPIO.??

### 3.7. Pozostałe czujniki podłączane do systemu przez magistralę I<sup>2</sup>C

W celu wykonania akwizycji danych pomiarowych wykorzystano dwa moduły z czujnikami ciśnienia, wilgotności powietrza oraz temperatury. Układy są podłączane przez magistralę I<sup>2</sup>C do płytka Raspberry Pi. Moduły zawierają sensory mierzące wartości warunków atmosferycznych w postaci układów scalonych typu MEMS (z ang. Micro Electro-Mechanical System - Mikrosystemy elektromechaniczne).

### **3.7.1. Moduł z czujnikiem ciśnienia i temperatury LPS25H**

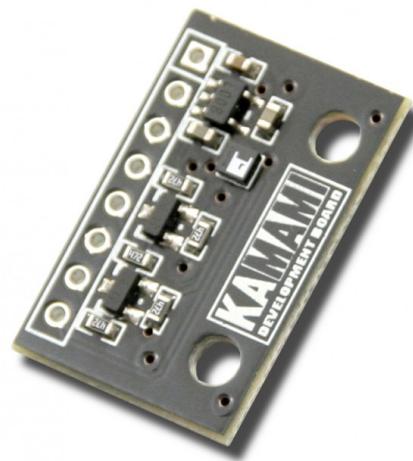
Moduł LPS25H od firmy Pololu zawiera barometr mierzący ciśnienie atmosferyczne w zakresie: 260 mbar - 1260 mbar (26 kPa - 126 kPa). W trybie wysokiej rozdzielczości czujnik mierzy z dokładnością 0,2 mbar (0,02kPa).



Rysunek 3.7. Moduł z czujnikiem ciśnienia i temperatury Pololu LPS25

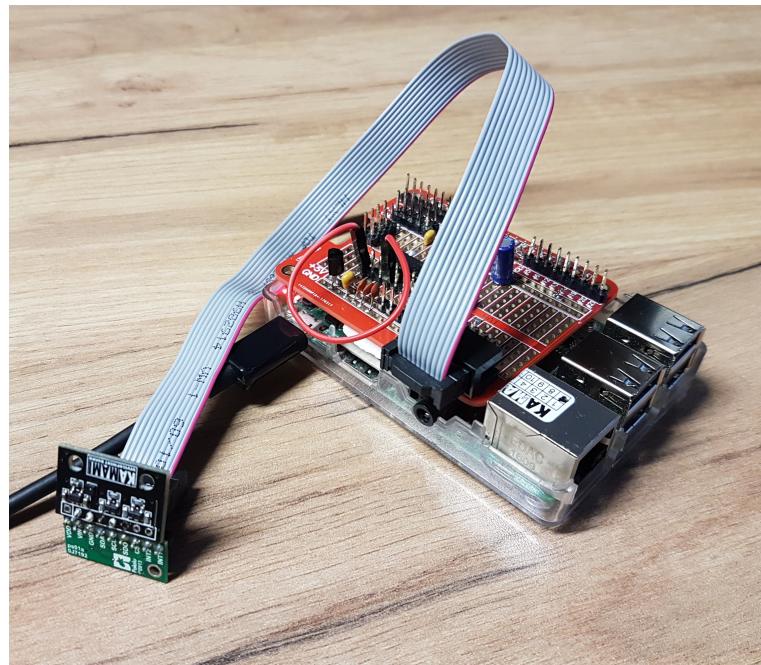
### **3.7.2. Moduł z czujnikiem wilgotności i temperatury z układem HTS221**

Moduł HTS221 zawiera sensor wilgotności i temperatury firmy STMicroelectronics. Zakres pomiaru temperatury to 0 - 60°C, a wilgotności powietrza: 0 - 100%. Układ obsługuje interfejsy I<sup>2</sup>C lub SPI. Na płytce znajdują się konwerter poziomów napięć co umożliwia współpracę układu z urządzeniami o poziomach 3,3V oraz 5V i stabilizator napięcia.



Rysunek 3.8. Moduł z czujnikiem wilgotności i temperatury HTS221

Moduły miały podobny rozstaw pinów co umożliwiło podłączenie ich do wspólnej szyny.



Rysunek 3.9. Podłączenie zewnętrznych czujników do płytki

## 4. Oprogramowanie

### 4.1. System operacyjny

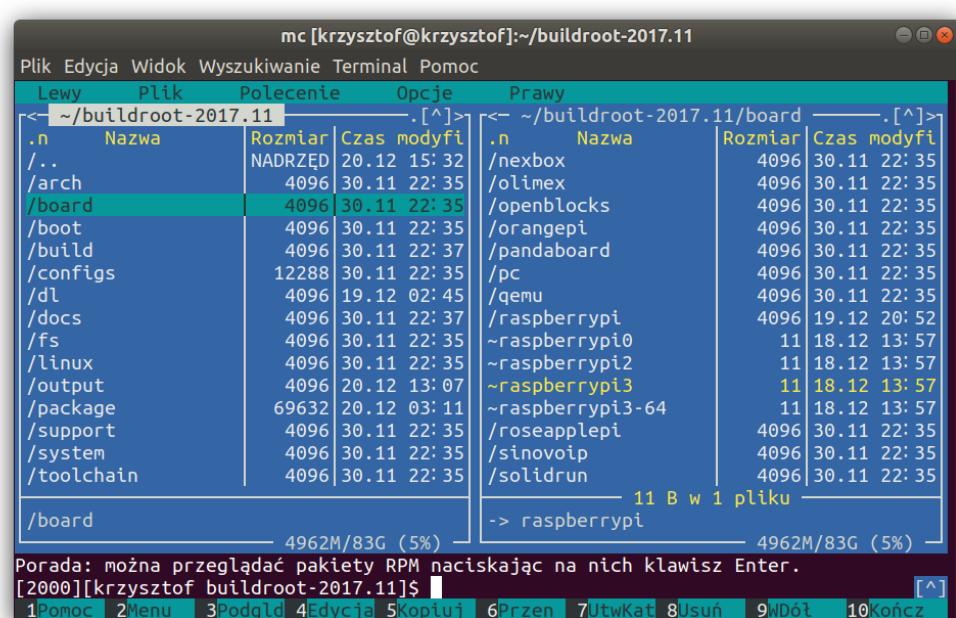
Najczęściej używanym systemem operacyjnym na platformie Raspberry Pi jest Raspbian. Jest to darmowa dystrybucja Linuxa oparta na Debianie, odpowiednio dostosowana do pracy z Raspberry Pi. System jest udostępniony przez twórców na stronie organizacji [4] Raspberry Pi Foundation zarówno w wersji 'Raspbian with Desktop' ze środowiskiem graficznym PIXEL (z ang. Pi Improved Xwindows Environment, Lightweight), jak i w wersji bez środowiska graficznego Raspbian Lite.

Raspbian jest wielozadaniowym systemem operacyjnym, więc przydziela wszystkim programom czas procesora według algorytmu szeregowania. Z punktu widzenia aplikacji użytkownika nie wiadomo kiedy nastąpi wywłaszczenie i na jak długo. Jest to problematyczne w przypadku obsługi krytycznych czasowo zadań. Rozwiążaniem może być zastosowanie modułu do jądra systemu operacyjnego i wykonanie krytycznych czasowo zadań w funkcji obsługi przerwania lub uruchomienie aplikacji na systemie czasu rzeczywistego (z ang. RTOS - Real Time Operating System) takim jak FreeRTOS lub ChibiOS. Systemy operacyjne czasu rzeczywistego zawierają mechanizmy ułatwiające implementacje systemu akwizycji danych. Podczas komplikacji jądra Linuxa istnieje możliwość dodania rozszerzeń czasu rzeczywistego, które udostępniają funkcje w przestrzeni jądra i użytkownika, ułatwiające implementację krytycznych czasowo aplikacji w systemie.

### 4.2. Środowisko Buildroot

Buildroot jest narzędziem umożliwiającym użytkownikowi stworzenie okrojonej na potrzeby danego zastosowania wersji systemu wbudowanego Linux. Środowisko to składa się z zestawu plików Makefile i kconfig, które upraszczają przygotowanie cross-kompilacji za pomocą zestawu narzędzi *toolchain* dla danej architektury. Producent dostarcza również domyślne konfiguracje do kilku popularnych platform m.in. Cubieboard i Raspberry Pi, co skraca czas rozpoczęcia projektu systemu. W pobranym ze strony producenta pliku w katalogu *board* (Rys. 4.1) znajduje się katalog raspberrypi zawierający domyślną konfigurację dla platformy Raspberry Pi 3.

Buildroot pozwala ustawić opcję dwóch rozszerzeń czasu rzeczywistego XENOMAI i RTAI. Są to rozszerzenia wykorzystywane w systemach wbudowanych do wspierania aplikacji czasu rzeczywistego. RTAI (z ang. real time application interface - interfejs dla aplikacji czasu rzeczywistego)



Rysunek 4.1. Katalog board z plikami konfiguracyjnymi

jest dostępny w środowisku Buildroot dla 64-bitowej architektury procesora Raspberry Pi 3. Moduły RTAI umożliwiają ustawienie priorytetu wywłaszczenia każdej aplikacji w systemie, co pozwala na zapewnienie poprawności działania krytycznych czasowo zadań.

#### 4.2.1. Interfejs środowiska

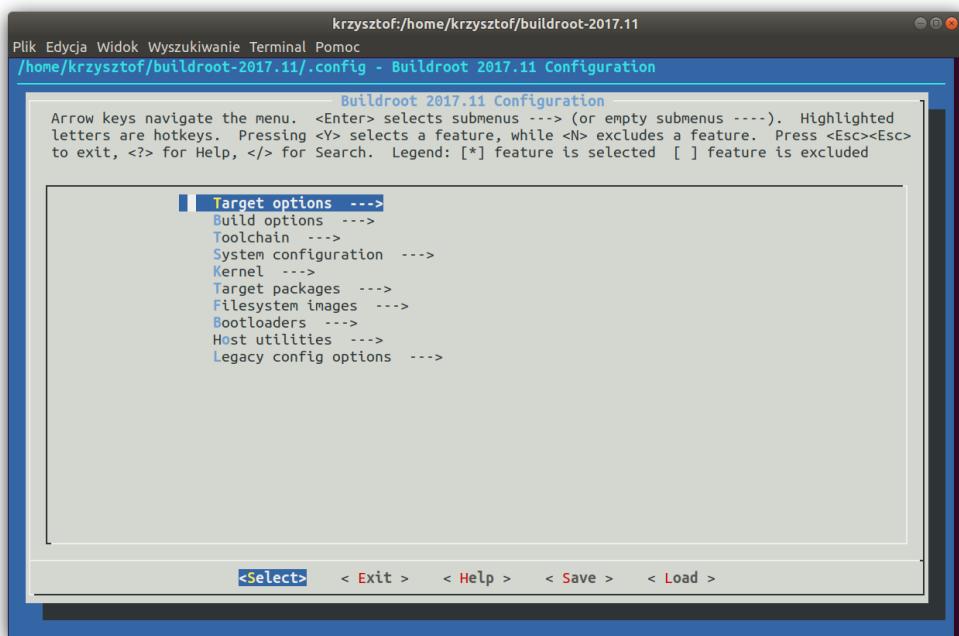
Buildroot tworzy system plików, kompliuje jądro systemu Linux i generuje *boot loader*. Konfiguracja komplikacji systemu jest ustawiana w menu głównym środowiska Buildroot za pomocą interfejsu wywoływanego komendą *make menuconfig*. Pojawia się graficzny interfejs przedstawiony na Rys. 4.2.

#### 4.2.2. Toolchain

Toolchain jest zestawem narzędzi programistycznych, pozwalających na wykonanie cross-kompilacji programu na maszynie o innej architekturze. Składa się z kompilatora, linkera oraz bibliotek danego języka programowania. W środowisku Buildroot użytkownik ma możliwość wybrania, czy Toolchain ma być stworzony przez środowisko zgodnie z opcjami dotyczącymi architektury sprzętu docelowego. Alternatywą jest podanie zewnętrznego zestawu narzędzi *Toolchain* i dołączenie do projektu.

#### 4.2.3. Dodawanie modułów jądra

Dodawanie skompilowanych modułów jądra odbywa się za pomocą komendy *modprobe*. Kod modułu wraz z plikami konfiguracyjnymi należy dodać podczas konfiguracji środowiska Buildroot. Na etapie dokonywania zmian w kodzie sterownika, Buildroot umożliwia cross-kompilację wybranego pakietu. Skompilowany moduł wysyłano za pomocą połączenia ssh.



Rysunek 4.2. Widok głównego menu środowiska Buildroot

### 4.3. Komunikacja z Raspberry Pi

Na etapie rozwoju oprogramowania, debugowania oraz testów potrzebna była komunikacja pomiędzy komputerem PC, a płytą Raspberry Pi. Platforma posiada kilka różnych możliwości komunikacji, jednak najbardziej niezawodnym i dającym pełną informację o stanie systemu był port UART. Aby móc obsługiwać konwerter USB-UART i komunikować się z Raspberry Pi na komputerze PC posłużono się programem minicom.

Komunikacja przez port Ethernet, bądź WiFi za pomocą klienta SSH (ang. Secure Shell) pozwalała na komunikację z większą przepustowością, co było istotne podczas testów systemu pod kątem akwizycji danych.

### 4.4. Sterowniki urządzeń

Do obsługi urządzeń dołączanych do systemu przy pomocy magistral SPI i I<sup>2</sup>C potrzebne są odpowiednie sterowniki urządzeń. Można skorzystać z gotowych modułów udostępniających aplikacjom użytkownika dostęp do pamięci urządzeń takich jak np. *spidev*[9] i *i2c-dev*[10]. W celu zapewnienia wydajnej pracy programom o ścisłym reżimie czasowym w projekcie zastosowano moduł do jądra systemu operacyjnego umożliwiający obsługę przetwornika analogowo-cyfrowego.

#### 4.4.1. Sterownik przetwornika analogowo-cyfrowego

Projekt zakładał napisanie kodu sterownika urządzenia umożliwiającego dostęp aplikacji do magistrali SPI w celu odebrania pomiaru napięcia analogowego z jednego z ośmiu kanałów przetwornika analogowo-cyfrowego MAX1202. Użytkownik powinien mieć możliwość ustawienia okresu

próbkowania przetwornika oraz wyzwolenia i przerwania pomiaru z poziomu aplikacji.

Komunikacja sterownika z aplikacją jest zapewniona poprzez urządzenie znakowe `kw_adc`. Do wysyłania komend do urządzenia zastosowano funkcję `ioctl()`. Przed pomiarem użytkownik ustawia za pomocą `ioctl` z komendą `ADC_SET` czas pomiędzy przerwaniami (pole `adc_sampling_period` w strukturze `dev`) co powoduje ustawienie okresu próbkowania przetwornika. W jednym transferze wysyłany jest 1 bajt komendy (pomiar z CH1 to 0x9f) i jednocześnie odbierane są 2 bajty danych pomiarowych.

Aby zaimplementować funkcjonalność zmiany częstotliwości próbkowania z poziomu aplikacji, użyto struktury `hr_timer`. Po skończeniu odliczania przez licznik uruchamiana jest funkcja obsługi przerwania, następuje przygotowanie odpowiednich struktur do transferu bajtów poprzez magistralę SPI. Ustawiane są parametry transmisji. Pomiar jest zlecanym przez funkcję `spi_async` asynchronicznie, więc proces nie jest blokowany. Gdy transfer jest zakończony uruchamiana jest funkcja `complete()`, gdzie dane pomiarowe są wpisywane do bufora cyklicznego. Z poziomu użytkownika dane są dostępne dzięki funkcji `poll()`, w której proces odczytu jest usypanym jeśli w buforze nie ma wystarczającej ilości danych do czytania. Po wpisaniu danych do bufora proces oczekujący na dane jest budzony w funkcji `complete()`.

Istotnym podczas pisania sterownika był fakt, że dane są odbierane w funkcji obsługi przerwania. Determinowało to użycie funkcji asynchronicznej, bo funkcja `spi_sync_transfer` może być usypana, więc nie możemy użyć jej w kontekście obsługi przerwania. Dokonując niewielkich zmian w konfiguracji zawartej w kodzie sterownika możliwe jest obsłużenie innych układów posiadających interfejs SPI. Zweryfikowaniu i ewentualnej zmianie powinny być poddane pola struktury `spi_transfer` opisującej parametry transferu spi, które są opisane w dokumentacji jądra systemu operacyjnego Linux.[12].

## 4.5. Protokół synchronizacji czasu

Aby otrzymać dokładną informację o czasie pobrania próbki z przetwornika lub czujnika potrzebna jest synchronizacja czasu pomiędzy urządzeniami, którą można zapewnić dzięki protokołowi synchronizacji czasu NTP (z ang. Network Time Protocol). Domyślnie w systemie Raspbian Jessie działa usługa `systemd-timesyncd`, która działa na podstawie protokołu SNTP (z ang. Simple NTP) i jest dużo mniej rozbudowana. W przeciwieństwie do NTP to tylko klient, który pobiera i aktualizuje czas w systemie. W przypadku gdy chcemy skorzystać z serwera NTP, najlepiej jest tę usługę wyłączyć[7]. Jak wspomniano w podrozdziale 2.2.1, dokładność synchronizacji protokołu utrzymuje się na poziomie kilkudziesięciu mikrosekund w zależności od wybranego serwera czasu.

### 4.5.1. Znacznik czasu

System akwizycji danych wraz z zapisaną wartością próbki zapisuje czas zebrania próbki. Wykorzystano czas systemowy, który został wcześniej zsynchronizowany z serwerem NTP. Początkowo w pierwszym podejściu do

problemu zastosowano rozwiązanie w przestrzeni użytkownika, przy użyciu funkcji *getMicrotime()*. Funkcja ta pozwala na zapisanie znacznika czasu w [ $\mu$ s] z racji na to, że jest wywoływana z programu w przestrzeni użytkownika, nie daje dużej dokładności. Nie da się jednoznacznie stwierdzić ile czasu minie pomiędzy zapisaniem wartości znacznika czasu i zebraniem próbki, ponieważ proces może być wywłaszczany.

Większą dokładność osiągnięto stosując licznik *hrtimer* w kodzie modułu jądra, który pozwala na odmierzanie czasu w nanosekundach. Na zwiększoną dokładność tego rozwiązania ma wpływ fakt, iż znacznik czasu jest zapisywany w przestrzeni jądra i dopiero później przekazywany do aplikacji użytkownika.

Dokładna wartość znacznika czasu jest przechowywana w strukturze *timespec*. Pobrano tą wartość dzięki funkcji *getnstimeofday()* zwracającej czas w nanosekundach i zapisano w zmiennej 64-bitowej. Znacznik czasu przekazywano do sterownika wraz z wartością próbki za pomocą bufora kołowego.

Tak zebrane dane mogą być wykorzystane w aplikacji z przestrzeni użytkownika na obliczenie znacznika czasu każdej próbki. To rozwiązanie pozwala osiągnąć większą dokładność i stałość częstotliwości próbkowania.

## 4.6. Drzewo urządzeń

W celu uruchomienia swojego modułu do jądra napisano nakładkę na drzewo urządzeń i skompilowaną dodano do odpowiedniego katalogu /boot/overlays, a następnie dopisano w pliku /boot/config.txt nazwę nakładki. Podane w nakładce parametry transmisji danych między urządzeniem podłączonym do magistrali, a płytą Raspberry Pi są ustalane przy załadowaniu modułu.

## 4.7. Konfiguracja modułu watchdog

Raspbian zawiera moduł do jądra obsługujący licznik układu *watchdog*. Aby uaktywnić układ należy dodać parametr do drzewa urządzeń za pomocą wpisania *dtparam=watchdog=on* w pliku zawierającym konfigurację ładowania systemu [15]. Dodatkowo w pliku /etc/modprobe.d/bcm2835-wdt.conf można zdefiniować czas braku aktywności systemu po jakim układ *watchdog* wywoła ponowne uruchomienie systemu, wpisując *options bcm2835\_wdt heartbeat=14 nowayout=0*. Należy stworzyć jeszcze plik /etc/modprobe.d/bcm2835-wdt.conf zawierający konfigurację modułu obsługującego układ *watchdog*: *options bcm2835\_wdt heartbeat=14 nowayout=0*.

## 4.8. Aplikacje w przestrzeni użytkownika

Programy użytkownika umożliwiają wyświetlenie danych zebranych przez niższą warstwę systemu. Wyniki pomiarów są zapisywane w plikach tekstowych i udostępniane sieciowo.

### 4.8.1. Aplikacja wykorzystująca moduł *spidev*

W celu porównania wydajności i parametrów akwizycji danych różnych rozwiązań programowych napisano program wykorzystujący moduł *spidev*. Moduł ten pozwala na odczyt danych doprowadzonych do magistrali SPI z poziomu aplikacji użytkownika. W rozdziale 5 przedstawiono wyniki testów i porównanie zmienności częstotliwości próbkowania. Aplikacja daje możliwość odczytu jednorazowego lub ciągłego z zadanym opóźnieniem. Opóźnienie jest ustalane za pomocą parametru *delay\_usecs*, będącego polem struktury *spi\_ioc\_transfer*. Komunikacja ze sterownikiem *spidev* jest zapewniona za pomocą funkcji *ioctl()* z komendą *SPI\_IOC\_MESSAGE*. Komenda jest wywoływana w pętli głównej programu, a odebrane dane z przetwornika są zapisywane do bufora. Następnie pomiar jest zapisywany do pliku wraz ze znacznikiem czasu.

### 4.8.2. Aplikacje wykorzystująca moduł *i2cdev*

W celu obsłużenia czujników podłączanych przez magistralę I<sup>2</sup>C wykorzystano moduł *i2cdev*. Moduł ten pozwala na odczyt danych doprowadzonych do magistrali SPI z poziomu aplikacji użytkownika.

Przydatnym narzędziem przy korzystaniu z magistrali I<sup>2</sup>C w Raspberry Pi jest pakiet *i2c-tools*. Po zainstalowaniu pakietu, można skorzystać z komendy *i2cdetect*, która wysyła zapytanie o adres urządzeń podłączonych do magistrali. W wyniku wyświetlana jest tablica urządzeń wraz z ich adresami jak przedstawiono na Rys. 4.3. Narzędzie umożliwia szybkie sprawdzenie, czy nie powstaje konflikt adresów pomiędzy urządzeniami dołączonymi do jednej magistrali oraz szybsze debugowanie błędów.

### 4.8.3. Programy publikujące i subskrybujące dane przy pomocy protokołu MQTT

Programy implementujące klienta i brokera MQTT napisano w języku Python w wersji 2 wykorzystując pakiet *paho.mqtt.client*, który udostępnia funkcje umożliwiające w łatwy sposób implementację aplikacji. Aplikacja publikująca *publish.py* wykonywana jest na Raspberry Pi i zawiera kod, który powoduje wysyłanie w pętli kolejno zebranych próbek. Publikacja zostaje nazwana tematem *client.publish("test\_mqtt", file\_content[l]);*. W aplikacji klienta *subscribe.py* zawarty temat subskrybcji, który musi być zgodny z tematem zawartym w aplikacji publikującej dane[14] *client.subscribe("test\_mqtt")*

Program odpowiadający za odebranie danych zawiera informację na temat połączenia sieciowego: adres IP oraz numer portu.

```

Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[689][pi /home/pi]$ i2cdetect 1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1.
I will probe address range 0x03-0x77.
Continue? [Y/n] Y
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  - - - - - - - - - - - - - - - - - - - - - -
10:  - - - - - - - - - - - - - - - - - - - - - -
20:  - - - - - - - - - - - - - - - - - - - - - -
30:  - - - - - - - - - - - - - - - - - - - - - -
40:  - - - - - - - - - - - - - - - - - - - - - -
50:  - - - - - - - - - - - - - - - - - - 5d - 5f
60:  - - - - - - - - - - - - - - - - 68 - - - -
70:  - - - - - - - - - - - - - - - - - - - - - -
[690][pi /home/pi]$ █

```

Rysunek 4.3. Tablica adresów urządzeń podłączonych do magistrali I<sup>2</sup>C

#### 4.8.4. Aplikacja webowa

Aplikacja webowa została napisana w języku php i html i jest programem uruchamianym na serwerze i zapewnia graficzny interfejs. Użytkownik jest w stanie uruchomić proces oraz go wyłączyć, korzystając z identyfikatora procesu zwracanego przez funkcję *shell\_exec*. Identyfikator uruchomionego procesu zapisywany jest w pliku pid.txt. Na Rys. 4.4 zamieszczono zrzut ekranu z działającej aplikacji w przeglądarce na komputerze użytkownika systemu.

### Pomiar napięcia analogowego

Okres próbkiowania [ns]:

Kanal przetwornika[1-8]:

### Twoja konfiguracja:

Okres próbkiowania: 1000000000  
 Kanal przetwornika: 1

Rysunek 4.4. Zrzut ekranu działającej aplikacji webowej

Dane zebrane przez czujniki zapisywane są w pliku tekstowym w katalogu, z którego była wywołana aplikacja. W przypadku wywołania z aplikacji internetowej jest to katalog `/var/www/html`. Parametry akwizycji danych mogą być ustawiane z poziomu aplikacji internetowej i zapisane do pliku konfiguracyjnego. Informacja o parametrach pomiarów jest wczytywana z pliku konfiguracyjnego przez program obsługujący dany czujnik. Po odebraniu próbki użytkownik jest w stanie podejrzeć aktualną wartość odczytaną z sensora uruchamiając aplikację obsługującą protokół MQTT.

## 5. Testy

Ostatnim etapem projektu było przeprowadzenie testów, pozwalających na sprawdzenie działania wszystkich elementów systemu akwizycji danych. Przeprowadzono testy sprzętu i oprogramowania, sprawdzana była poprawność i niezawodność działania całego systemu z dołączanymi zewnętrznymi czujnikami.

### 5.1. Przebieg testów

Przeprowadzono szereg testów funkcjonalnych w celu sprawdzenia czy system spełnia założenia projektowe zawarte we wstępie pracy.

- analiza częstotliwościowa z wykorzystaniem algorytmu FFT (z ang. Fast Fourier Transform - szybkich transformaty Fouriera) sygnału sinusoidalnego z zastosowaniem odpowiedniego okna czasowego
- sprawdzenie dokładności pomiarów przy użyciu laboratoryjnych przyrządów pomiarowych
- test niezawodności działania przy dużym obciążeniu aplikacji
- sprawdzenie poprawności danych z uwzględnieniem korelacji czasowej
- test funkcjonalny komunikacji sieciowej

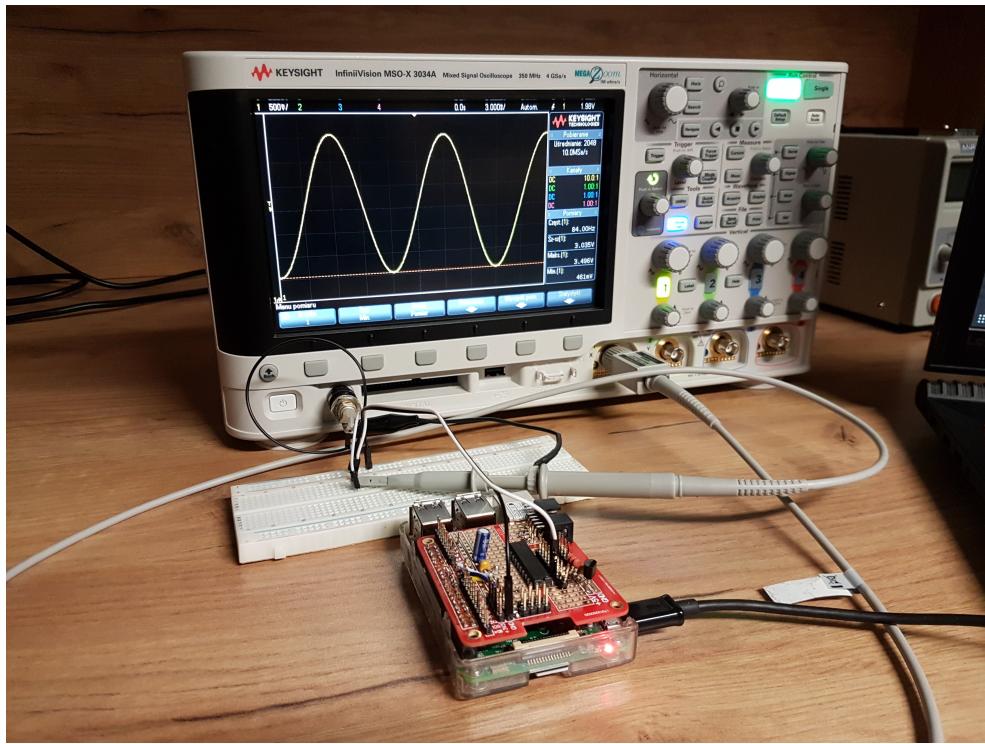
### 5.2. Użyte narzędzia

Sprawdzenie parametrów pomiaru napięcia przetwornika analogowo-cyfrowego przeprowadzono z użyciem oscyloskopu cyfrowego z funkcją generowania przebiegów sygnałów. Fakt, iż oscyloskop był cyfrowy ułatwił analizę różnic w sygnale wyjściowym z generatora i sygnałem spróbkowanym. Pomiary na oscyloskopie przeprowadzane były z użyciem sondy oscyloskopowej. Na Rys. 5.1 przedstawiono stanowisko pomiarowe systemu akwizycji danych.

### 5.3. Testy funkcjonalne

#### 5.3.1. Test interfejsu sieciowego

Wykonano test działania interfejsów sieciowych systemu użytkownika pod kątem spójności danych wysyłanych z systemu do użytkownika podłączonego przez sieć TCP/IP. Połączenie użytkownika korzystającego z przeglądarki w komputerze osobistym z systemem akwizycji danych było zestawione przy wykorzystaniu routera Wi-Fi.



Rysunek 5.1. Stanowisko pomiarowe do pomiaru napięcia sinusoidalnego przy użyciu przetwornikiem analogowo-cyfrowym

### 5.3.2. Testy niezawodności systemu

Po ustaleniu odpowiedniej konfiguracji przeprowadzono testy niezawodności działania układu *watchdog*, wywołując aplikacje powodujące zawieszenie systemu z załadowanym modułem. Oprócz konsoli *ssh* podłączono komputer przez konwerter USB-UART w celu upewnienia się, że system jest uruchamiany ponownie. Testy zakończono sukcesem, moduł sterujący układem *watchdog* działał poprawnie i wywoływał ponowne uruchomienie systemu przy każdym zawieszeniu.

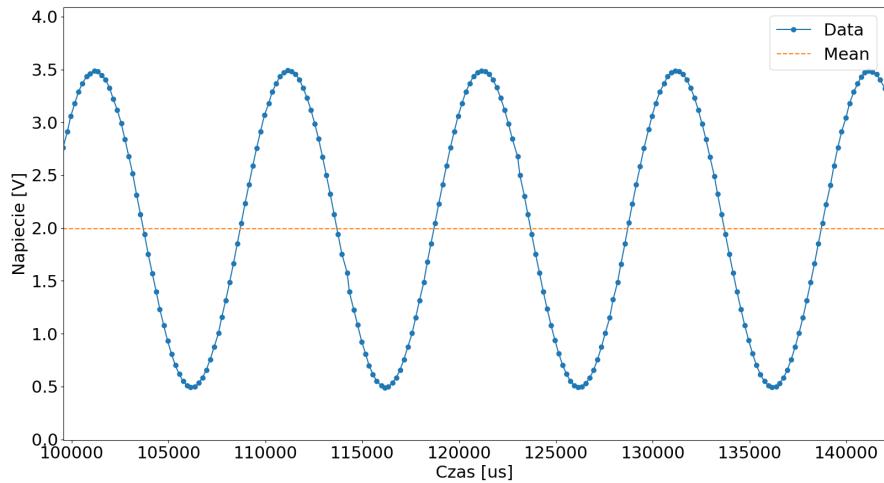
## 5.4. Wyniki pomiarów

### 5.4.1. Charakterystyki napięcia spróbkowanego przetwornikiem ADC MAX1202

#### Pomiar napięcia sinusoidalnego

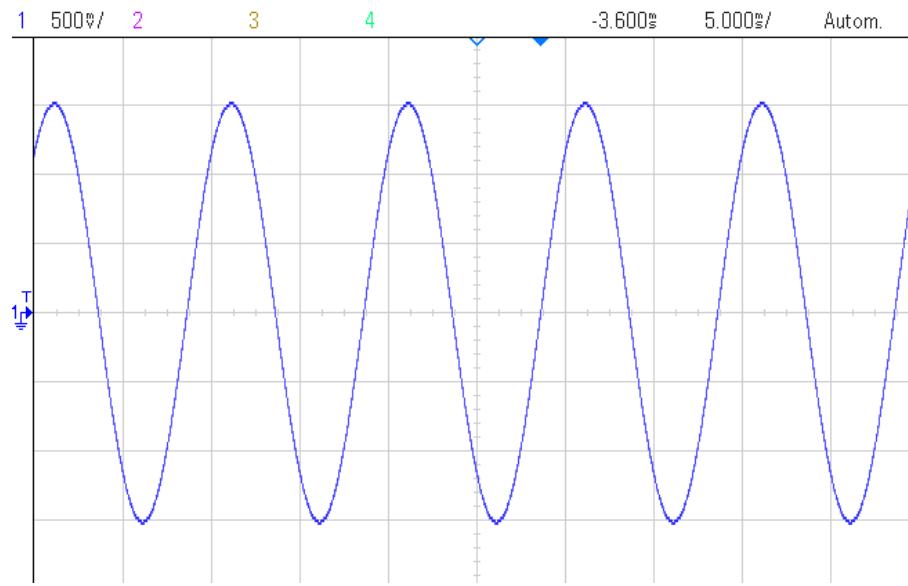
Pomiar napięcia sinusoidalnego i jego analiza częstotliwościowa, umożliwia sprawdzenie poziomu nierównomierności okresu próbkowania. Z oscyloskopu z funkcją generatora sygnałów wygenerowano przebieg sinusoidalny o częstotliwości 100Hz, amplitudzie 3Vp-p i składowej stałej 2V. Okres próbkowania przetwornika ustalono w aplikacji testowej na 200us co przekłada się na częstotliwość próbkowania 5kHz. Spróbkowany sygnał został poddany analizie za pomocą algorytmu FFT przy użyciu skryptu pythona z pakietami *numpy* i *matplotlib*. Poniżej na Rys. 5.2 przedstawiono fragment

wykreslonego przebiegu czasowego sygnału sinusoidalnego. Linią przerywaną zaznaczono wartość średnią sygnału.



Rysunek 5.2. Przebieg napięcia spróbkowanego przetwornika ADC MAX1202

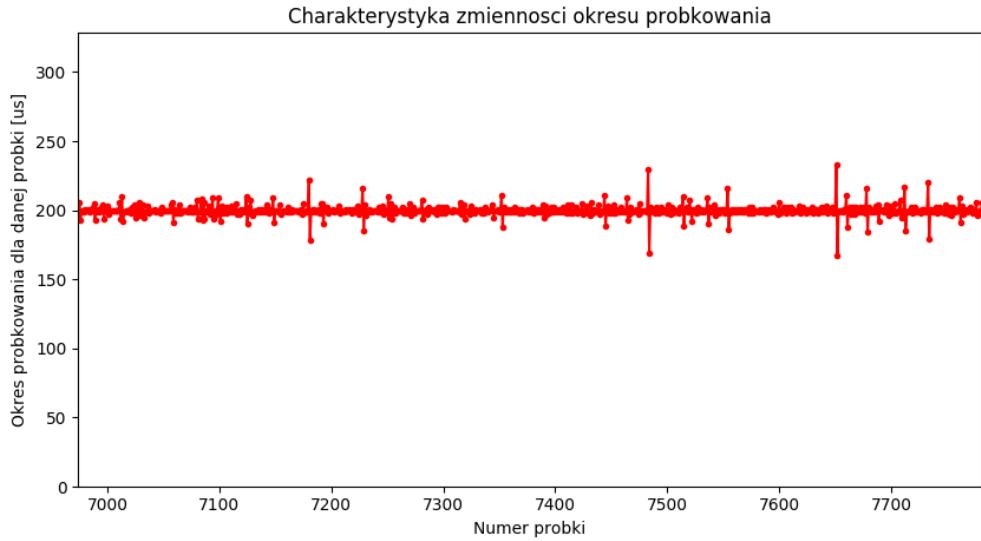
Dla porównania sygnał z generatora zmierzono oscyloskopem przy użyciu sondy oscyloskopowej. Zrzut z ekranu oscyloskopu poniżej na Rys. 5.3



Rysunek 5.3. Przebieg napięcia sinusoidalnego z oscyloskopu

Poniżej na Rys. 5.4 przedstawiono charakterystykę zmienności okresu próbkowania. Widać na niej jak zmienia się okres próbkowania w trakcie działania programu.

Zmienność okresu próbkowania opisano przy użyciu parametru odchylenia standardowego. Wyniki z pomiaru napięcia sinusoidalnego w Tablicy 5.1.:



Rysunek 5.4. Charakterystyka zmienności okresu próbkowania przy użyciu sterownika do ADC

Ilość zebranych próbek	19958
Średni okres próbkowania [ $\mu\text{s}$ ]	200
Średnia częstotliwość próbkowania [Hz]	5000
Wariancja okresu próbkowania [ $\mu\text{s}$ ]	454
Odchylenie standardowe okresu próbkowania [ $\mu\text{s}$ ]	21,3

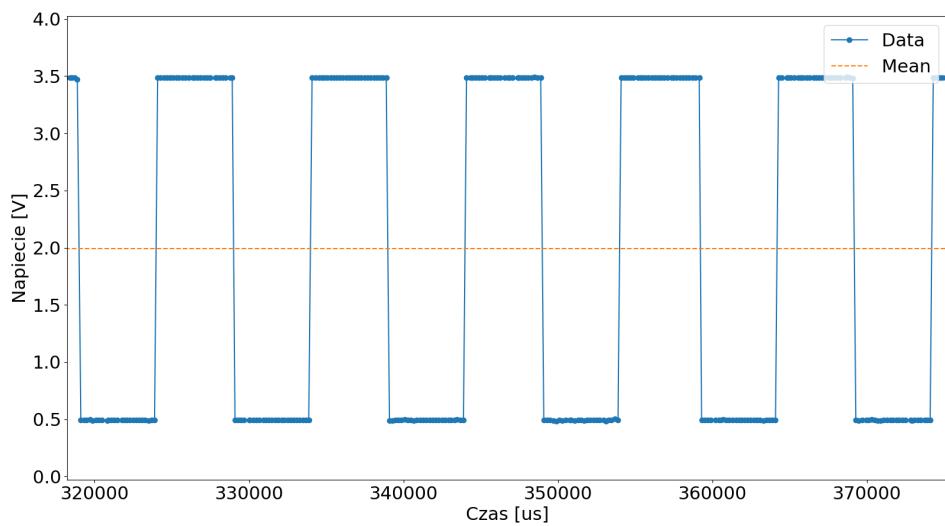
Tablica 5.1. Parametry opisujące zmienność okresu próbkowania

### Pomiar napięcia prostokątnego

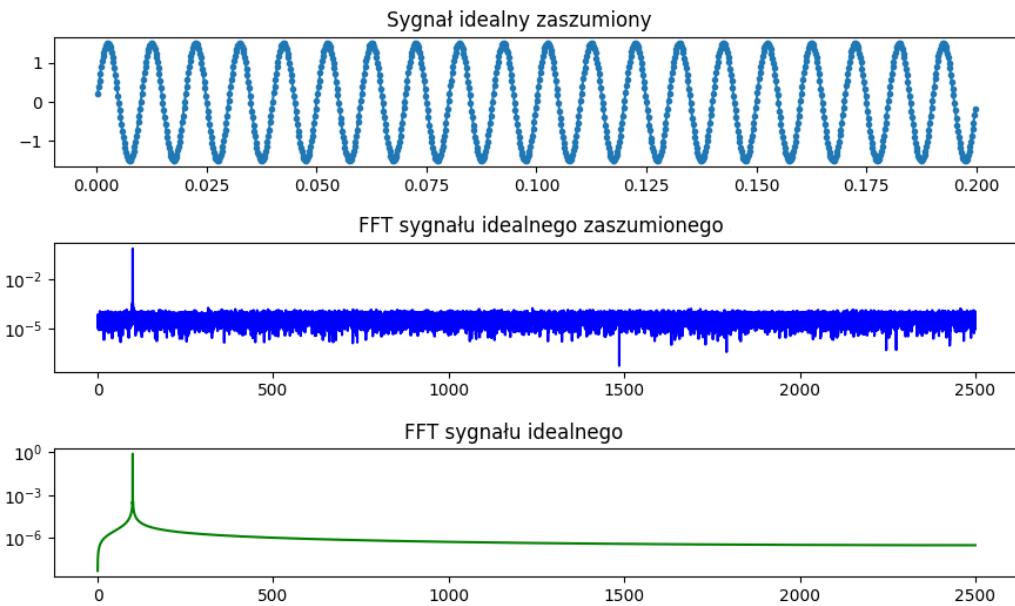
Wykonano również pomiar napięcia prostokątnego. Sygnał spróbkowano z częstotliwością 5kHz, charakterystyka jest przedstawiona na Rys. 5.5, a oscylogram na Rys. 5.6

#### 5.4.2. Analiza FFT sygnału sinusoidalnego

Przed wykonaniem pomiaru dokonano symulacji komputerowej w celu dostrzeżenia różnic charakterystyk częstotliwościowych między sygnałem sinusoidalnym bez zniekształceń, a sygnałem z nierównomiernymi odstępami czasowymi pomiędzy kolejnymi próbками. Sygnał został wygenerowany numerycznie, a następnie zniekształcony poprzez dodanie do wektora czasu pseudolosowych wartości z zakresu od 0 do 50us. Rysunek 5.7 przedstawia przebieg czasowy sygnału zniekształconego, jego charakterystykę częstotliwościową oraz widmo sygnału sinusoidalnego bez zniekształceń. Charakterystyki wyznaczono przy użyciu algorytmu FFT i przemnożeniu przez okno Hanninga.

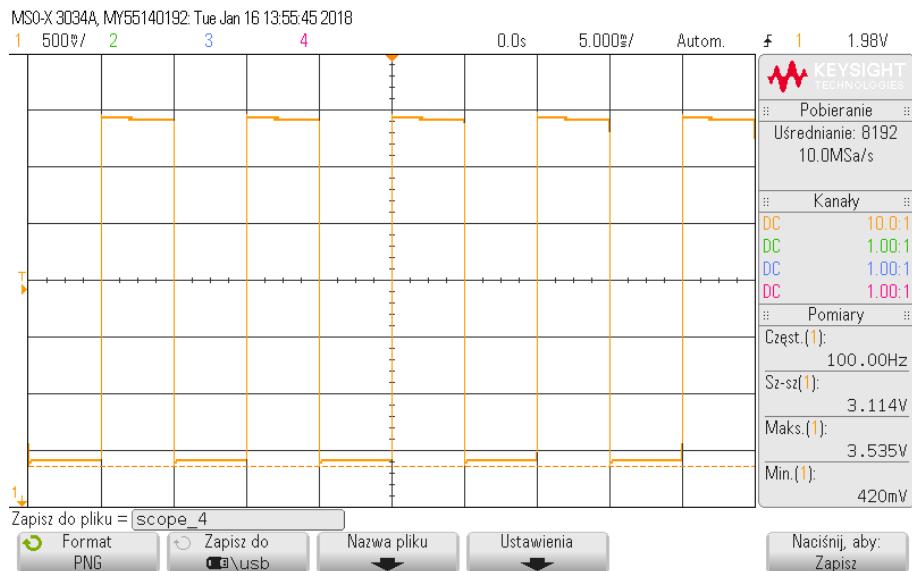


Rysunek 5.5. Przebieg napięcia spróbkowanego przetwornika ADC MAX1202

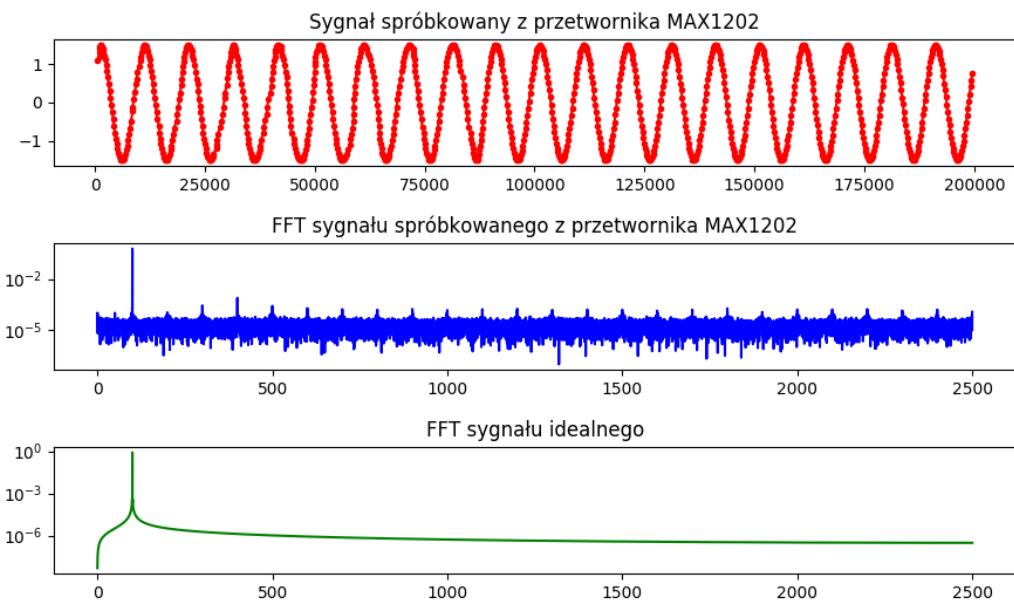


Rysunek 5.7. Analiza FFT dla sygnału sinus wygenerowanego numerycznie

W celu zbadania stałości częstotliwości próbkowania przetwornika dokonano analizy FFT spróbkowanego sygnału. Rysunek 5.8 przedstawia przebieg czasowy otrzymany z 50000 próbek zebranych przetwornikiem ADC MAX1202, widmo sygnału spróbkowanego przemnożonego przez okno Hanning'a oraz charakterystyka częstotliwościowa sygnału sinusoidalnego wygenerowanego numerycznie.



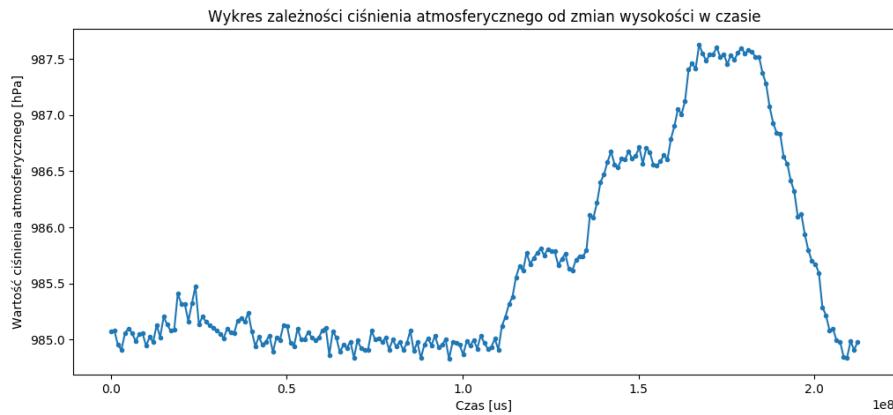
Rysunek 5.6. Przebieg napięcia prostokątnego na oscyloskopie



Rysunek 5.8. Analiza FFT dla sygnału spróbkowanego przetwornikiem ADC MAX1202

#### 5.4.3. Pomiary ciśnienia czujnikiem LPS25H

Zmierzono ciśnienie atmosferyczne w windzie gmachu Elektroniki i Technik Informacyjnych poruszającej się pomiędzy piętrami -2. i 4. zatrzymując się po drodze na 2. piętrze i na parterze. Na Rys. 5.9 przedstawiono wyniki pomiarów. Jak widać na rysunku ciśnienie maleje zgodnie z wzorem barometrycznym wraz ze wzrostem wysokości. Po odpowiedniej kalibracji czujnik można wykorzystać do pomiaru wysokości.



Rysunek 5.9. Wykres zmian ciśnienia panującego w poruszającej się windzie

## 5.5. Analiza wyników

W celu dokonania porównania pomiaru maksymalnej częstotliwości próbkowania przeprowadzono analizę częstotliwościową spróbkowanego sygnału przy użyciu algorytmu FFT. Zmienność częstotliwości próbkowania zobrazowano przy pomocy parametru odchylenia standardowego z poszczególnych odstępów czasowych pomiędzy momentami zebrania kolejnych próbek.

$$\sigma = \sqrt{\frac{1}{N} \sum_0^N Var(x)} \quad (5.1)$$

x - odstępy pomiędzy momentami zebraniem próbki

N - ilość zebranych próbek

### 5.5.1. Prezentacja i wizualizacja danych pomiarowych

Optymalnym do tego zadania i znacznie ułatwiającym analizę wyników narzędziem był skrypt napisany w języku Python przy wykorzystaniu pakietów numpy i matplotlib. Są to pakiety zapewniające interfejs pozwalający na prezentację wyników w formie ułatwiającej analizę danych. Pliki z danymi ładowane były do aplikacji wyświetlającej wykresy w formacie pliku rozdzielanego przecinkami (csv).

## 6. Podsumowanie

Cel pracy został osiągnięty, stworzono projekt i oprogramowanie sieciowego systemu akwizycji danych. Założenie optymalizacji kosztu zostało spełnione, układ jest dużo tańszy od rozwiązań komercyjnych, które najczęściej wymagają wykupienia licencji na oprogramowanie pozwalające sterować pomiarami. Główną część systemu - Raspberry Pi w wersji Zero W z modułem Wifi i Bluetooth 4.1 można nabyć za około 50zł, jednak zastąpienie przetestowanej wersji Raspberry Pi 3 wersją ZeroW wymagałoby wykonania dodatkowych testów, gdyż różnice parametrów sprzętowych tych dwóch płyt są znaczące i mogą wprowadzać ograniczenia możliwości systemu.

Opracowano sterownik przetwornika analogowo-cyfrowego podłączonego do komputera za pomocą interfejsu SPI i przetestowano pomyślnie jego działanie.

Sterowniki pozostałych urządzeń obsługiwanych po magistrali I<sup>2</sup>C nie zostały opracowane. W trakcie wykonywania pracy zdecydowano, że urządzenia podłączane przez magistralę I<sup>2</sup>C zostaną obsłużone za pomocą modułu *i2cdev*, gdyż nie wymagają próbkowania z dużą częstotliwością.

Podczas tworzenia oprogramowania kierowano się założeniem akwizycji danych z zachowaniem jak najpełniejszej (w granicach możliwości technicznych) informacji o czasie pomiaru. Podjęto szereg działań, które pomogły w spełnieniu powyższego założenia oraz przetestowano działanie sterownika za pomocą aplikacji testowej.

Zrealizowano funkcjonalność sterowania pomiarem przez użytkownika z poziomu aplikacji webowej oraz wysyłanie danych za pomocą protokołu MQTT. Obsłużono czujniki wilgotności i ciśnienia przy użyciu sterownika *i2cdev* oraz przetwornika analogowo-cyfrowego za pomocą sterownika *spidev*. Przetwornik przetestowano również za pomocą własnego sterownika i aplikacji testowej. Pomyślnie przetestowano działanie układów i akwizycję danych.

Użytkownik komunikuje się z systemem dzięki interfejsowi graficznemu aplikacji w przeglądarce internetowej. Zaletą systemu jest jego łatwa rozszerzalność. Dokonanie niewielkich zmian w oprogramowaniu i możliwość podłączenia urządzeń do magistrali SPI i I<sup>2</sup>C oraz UART i pozostałych pinów GPIO sprawia, że użytkownik jest w stanie rozszerzać system o nowe funkcjonalności. Ponadto fakt, iż Raspberry Pi jest popularnym minikomputerem dostępnym w niskiej cenie, zapewnia użytkownikowi wsparcie w dokonywaniu ewentualnych zmian w konfiguracji i strukturze systemu akwizycji danych.

Projekt może być użyty do zastosowań akwizycji danych, na przykład jako stacja meteorologiczna z możliwością zdalnego dostępu dzięki komunikacji sieciowej. Rozwiązania zawarte w projekcie umożliwiają podgląd danych w trakcie wykonywania pomiaru oraz pobranie danych przez użytkownika w celu wykonania dokładniejszej analizy. Część pracy zostanie wykorzystana podczas misji balonowej Studenckiego Koła Inżynierii Kosmicznej Politechniki Warszawskiej do zbierania danych z czujników udostępnianych przez łącze radiowe w paśmie radioamatorskim.

# Bibliografia

- [1] National Instruments website,  
<http://www.ni.com/en-us/support/model.usb-6008.html>
- [2] MQTT - documentation  
<http://mqtt.org/>
- [3] Analog Discovery 2 Reference Manual,  
<https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual?redirect=1>
- [4] Raspberry Pi Learning Resources, Raspberry Pi Software Guide,  
Raspberry Pi Foundation, 2017,  
<https://www.raspberrypi.org/learning/software-guide/quickstart/>
- [5] Raspberry Pi 3 on sale  
<https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/>
- [6] ADC MAX1202 datasheet,  
<https://datasheets.maximintegrated.com/en/ds/MAX1202-MAX1203.pdf>
- [7] Building a Raspberry-Pi NTP Server,  
<http://raspberrypi.tomasgreno.cz/ntp-client-and-server.html>
- [8] RTC DS3231 datasheet,  
<https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>
- [9] SPI driver - spidev documentantion  
<https://www.kernel.org/doc/Documentation/spi/spidev>
- [10] I2C driver - i2c-dev documentantion  
<https://www.kernel.org/doc/Documentation/i2c/dev-interface>
- [11] RPi.GPIO 0.6.3 : Python Package Index  
<https://pypi.python.org/pypi/RPi.GPIO>
- [12] Serial Peripheral Interface (SPI) — The Linux Kernel documentation,  
<https://www.kernel.org/doc/html/v4.14/driver-api/spi.html>
- [13] The Buildroot user manual, 2017,  
<https://buildroot.org/downloads/manual/manual.html>
- [14] paho-mqtt 1.1 : Python Package Index  
<https://pypi.python.org/pypi/paho-mqtt/1.1>
- [15] Watchdog install on Raspbian Jessie - Raspberry Pi Forums, 2017,  
<https://www.raspberrypi.org/forums/viewtopic.php?t=175432>