AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

Praca dyplomowa inżynierska

Patryk Synowiec

Imię i nazwisko

Automatyka i Robotyka

Kierunek studiów

Koncepcja urządzenia do analizy sygnałów audio na platformie wbudowanej

Temat pracy dyplomowej

Dr inż. Adam Jabłoński	…….
Promotor pracy	*Ocena*

Kraków, dnia ……

Imię i nazwisko: Patryk Synowiec

Numer albumu: 308012

Kierunek studiów: Automatyka i Robotyka

OŚWIADCZENIE

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tj. Dz. U. z 2006 r. Nr 90, poz. 631 z późn.zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (tj. Dz.U. z 2012 r. poz. 572, z późn.zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim", oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy".

…….

podpis dyplomanta

Kraków, dnia ……

Imię i nazwisko: Patryk Synowiec

Numer albumu: 308012

Kierunek studiów: Automatyka i Robotyka

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą inżynierską pracę dyplomową wykonałem osobiście i samodzielnie oraz nie korzystałem ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja oraz praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzyskałem w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza inżynierska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

…….

podpis dyplomanta

Kraków, dnia ……

Imię i nazwisko: Patryk Synowiec

Adres korespondencyjny: ul. Polskiej Organizacji Wojskowej 13/7, 42-200 Częstochowa

Temat pracy dyplomowej inżynierskiej: Koncepcja urządzenia do analizy sygnałów audio na platformie wbudowanej

Rok ukończenia studiów: 2022

Numer albumu: 308012

Kierunek studiów: Automatyka i Robotyka

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie, udzielam Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji inżynierskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej[1].

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej[2].

Kraków, …… ……

data podpis dyplomanta

[1] Na podstawie Ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (Dz.U. 2005 nr 164 poz. 1365) Art. 239. oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a. "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

[2] Ustawa z dnia 30 czerwca 2000r. – Prawo własności przemysłowej (Dz.U. z 2003r. Nr 119, poz. 1117 z późniejszymi zmianami) a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. nr 102 poz. 1119 oraz z 2005r. Nr 109, poz. 910).

Kraków, dnia ……

AKADEMIA GÓRNICZO-HUTNICZA
WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ INŻNYNIERSKIEJ

dla studenta IV roku studiów stacjonarnych

Patryk Synowiec

TEMAT PRACY DYPLOMOWEJ INŻNYNIERSKIEJ:

Koncepcja urządzenia do analizy sygnałów audio na platformie wbudowanej

Promotor pracy: dr inż. Adam Jabłoński

Recenzent pracy: prof. dr hab. inż. Tomasz Barszcz ……

podpis dziekana

PLAN PRACY DYPLOMOWEJ:

1. Omówienie tematu pracy i sposobu realizacji z promotorem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Zebranie i opracowanie wyników badań.
4. Analiza wyników badań, ich omówienie i zatwierdzenie przez promotora.
5. Opracowanie redakcyjne.

Kraków, …… ……

data podpis dyplomanta

TERMIN ODDANIA DO DZIEKANATU: **....**…….. **20**… r.

…….

podpis promotora

Kraków, dnia ……

Akademia Górniczo-Hutnicza im. Stanisława Staszica

Wydział Inżynierii Mechanicznej i Robotyki

Kierunek: Automatyka i Robotyka

Patryk Synowiec

Praca dyplomowa inżynierska

Koncepcja urządzenia do analizy sygnałów audio na platformie wbudowanej

Opiekun: dr inż. Adam Jabłoński

STRESZCZENIE

Celem niniejszej pracy jest zaprojektowanie urządzenia umożliwiającego rejestrowanie oraz dokonywanie analizy sygnałów audio. W ramach pracy stworzono dwa urządzenia: jedno oparte na Arduino UNO oraz drugie oparte na Raspberry Pi 2. Urządzenia komunikują się poprzez magistralę SPI z przetwornikiem analogowo-cyfrowym MCP3008 i rejestrują sygnały przekonwertowane z postaci analogowej do cyfrowej. Analizowane sygnały pochodzą z układu złożonego z czterech mikrofonów elektretowych połączonych z zaprojektowanymi wzmacniaczami. Wartości opóźnienia pomiędzy sygnałami zostały wyznaczone w oparciu o dwie metody, z czego jedna oparta jest o funkcję korelacji wzajemnej sygnałów. Na podstawie wartości opóźnień oraz położenia mikrofonów wyznaczone zostało położenie źródła dźwięku.

Cracow, ……

AGH University of Science and Technology

Faulty of Mechanical Engineering and Robotics

Field of Study: Automatics and Robotics

Patryk Synowiec

Engineer Diploma Thesis

The concept of a device for analysis of audio signals on the embedded platform

Supervisor: dr inż. Adam Jabłoński

SUMMARY

The purpose of this thesis is to design a device for recording and analysing audio signals. As part of the work, two devices were created: first one based on Arduino UNO and second one based on Raspberry Pi 2. The devices communicate via SPI bus with the MCP3008 analog-to-digital converter and record signals converted from analog to digital form. The analysed signals come from a circuit consisting of four electret microphones which were connected to designed amplifiers. The values of the delay between the signals were determined based on two methods, one of which is based on cross-correlation. Based on the delay values and the position of the microphones, the position of the sound source was determined.

PODZIĘKOWANIA

Dziękuję mojemu promotorowi dr inż. Adamowi Jabłońskiemu za udzielone wsparcie i uwagi niezbędne do napisania niniejszej pracy.

Spis treści

Wstęp

Przetwarzanie sygnałów jest dyscypliną łączącą wiedzę z zakresu telekomunikacji, informatyki, elektroniki i matematyki, dotyczącą analizowania informacji z otaczających człowieka procesów fizycznych. Z uwagi na rewolucję w zakresie technologii układów scalonych, cyfrowe przetwarzanie sygnałów stało się ważnym i nowoczesnym narzędziem stosowanym w różnorodnych działach nauki i techniki, takich jak akustyka, biomedycyna, hydrolokacja, sejsmologia, przetwarzanie i rozpoznawanie mowy, transmisja danych, grafika komputerowa oraz radiolokacja. Rozpowszechnienie dostępu do komputerów oraz stworzenie platform takich jak Arduino oraz Raspberry Pi sprawiło, że cyfrowe przetwarzanie sygnałów stało się bardziej powszechne i łatwe w realizacji.[1][2]

Analiza akustyczna jest dyscypliną opartą na przetwarzaniu sygnałów dźwiękowych pochodzących z otoczenia. Jednym z jej wielu zastosowań jest lokalizacja dźwięku, w ramach której określone zostaje położenie źródła dźwięku oraz jego odległość od słuchacza. Techniczne metody lokalizacji dźwięku naśladują mechanizmy obecne w przyrodzie i inspirowane się działaniem układów słuchowych człowieka oraz innych ssaków.

Cel i założenia pracy

Celem niniejszej pracy jest stworzenie urządzenia pozwalającego na rejestrowanie oraz dokonywanie analizy sygnałów akustycznych. Na podstawie wartości opóźnień pomiędzy sygnałami wyznaczone zostaną współrzędne źródła dźwięku, realizując algorytm lokalizacji dźwięku.

Zaprojektowany układ zostanie oparty na platformach Arduino oraz Raspberry Pi, które będą sterowały procesem rejestrowania sygnałów akustycznych pochodzących z macierzy mikrofonów elektretowych.

W ramach pracy przedstawiona zostanie koncepcja oraz opis działania urządzenia, schemat elektryczny pokazujący sposób połączenia jego elementów składowych i algorytmy, na podstawie których przeprowadzane są procesy rejestrowania i analizy sygnałów dźwiękowych, a także wyznaczania współrzędnych położenia źródła dźwięku.

1. Część teoretyczna

1.1. Raspberry Pi

1.1.1. Historia

Raspberry Pi to seria minikomputerów ogólnego przeznaczenia rozwijana przez edukacyjną organizację charytatywną Raspberry Pi Foundation założoną przez Ebena Uptona, we współpracy z amerykańskim przedsiębiorstwem Broadcom, będącym producentem układów półprzewodnikowych. Na początku działalności fundacji w 2009 roku, założycielowi przyświecała idea rozpowszechniania nauczania w zakresie informatyki na wczesnym etapie edukacji. W ramach tego w 2012 roku wypuszczony został pierwszy model minikomputera nazwany Raspberry Pi, który ze względu na niską cenę miał być wykorzystany do spopularyzowania nauki programowania w szkołach. Ze względu na niski koszt, wszechstronność zapewnioną poprzez zastosowanie wielu interfejsów komunikacyjnych, a także wolne i otwarte oprogramowanie, produkt znalazł uznanie również wśród bardziej doświadczonych programistów. Popularność produktu przekroczyła oczekiwania projektantów, dlatego seria sukcesywnie powiększała się o kolejne modele, a najnowszym modelem z serii Raspberry Pi jest pochodzący z 2019 roku Raspberry Pi 4 Model B.[3]

W 2021 roku wypuszczony został pierwszy model z serii Raspberry Pi Pico, pierwszy układ w historii organizacji oparty na pojedynczym, autorskim mikrokontrolerze RP2040 o taktowaniu 133 MHz. W odróżnieniu od wszystkich pozostałych minikomputerów ogólnego przeznaczenia stworzonych przez Raspberry, układ ten może być z powodzeniem zastosowany jako system wbudowany.[4]

Rys. 1. Raspberry Pi Pico

1.1.2. Specyfikacja Raspberry Pi 2 Model B v1.1

Do rejestrowania sygnałów audio wykorzystany został minikomputer Raspberry Pi 2 Model B v1.1. Do jego budowy użyto układu Broadcom BCM2836, opartego na czterordzeniowym procesorze ARM Cortex-A7 o 32-bitowej architekturze i częstotliwości taktowania wynoszącej 900 MHz. Dostępna pamięć RAM ma wielkość 1 GB.

Moc obliczeniowa Raspberry Pi 2 Model B v1.1 wynosi $1.47 \cdot 10^9$ FLOPS (floating point operations - operacje na liczbach zmiennoprzecinkowych na sekundę). Dla porównania moc obliczeniowa najnowszego modelu Raspberry Pi 4 (opartego na czterordzeniowym procesorze ARM Cortex-A72 o 64-bitowej architekturze i częstotliwości taktowania wynoszącej 1.5 GHz oraz dysponującego 4 GB pamięci RAM) wynosi $13.5 \cdot 10^9$ FLOPS.[5] [6]

Najważniejszymi portami, którymi dysponuje wykorzystany model Raspberry Pi są Ethernet i cztery porty USB. System operacyjny oraz dane użytkownika przechowywane są na karcie pamięci microSD. Układ zasilany jest poprzez 5 V napięcia stałego.

Rys. 2. Raspberry Pi 2 Model B v1.1, wykorzystany w ramach niniejszej pracy

1.1.3. Systemy operacyjne

Podstawowym systemem operacyjnym dla minikomputerów jest Raspberry Pi OS (dawniej nazywany Raspbianem) dostarczany przez Raspberry Pi Foundation. Jego bazą jest Debian – dystrybucja GNU/Linux pracująca w architekturze 32-bitowej. Wysoka popularność Rapsberry Pi spowodowała, że liczba dostępnych systemów operacyjnych obsługujących platformę jest duża, stąd oprogramowanie można dobrać pod potrzeby

programisty. W ramach niniejszej pracy inżynierskiej wykorzystano domyślny system operacyjny Raspberry Pi OS.[7][8]

1.1.4. Języki programowania

Kompilatory języków Python2, Python3, C/C++ oraz Scratch są domyślnie zainstalowane jako część Raspberry Pi OS. System operacyjny oparty jest na dystrybucji Linuxa, dlatego kompilatory każdego języka programowania są dostępne do użycia.[9] W ramach niniejszej pracy stworzono skrypty napisanie w Pythonie oraz C.

1.1.5. Architektura ARM

Procesory minikomputerów Raspberry Pi oparte są na architekturze ARM (Advanced RISC Machines) wprowadzonej w 1985 roku, spełniającej założenia modelu RISC (Reduced Instruction Set Computer). Jego podstawowe założenia to:

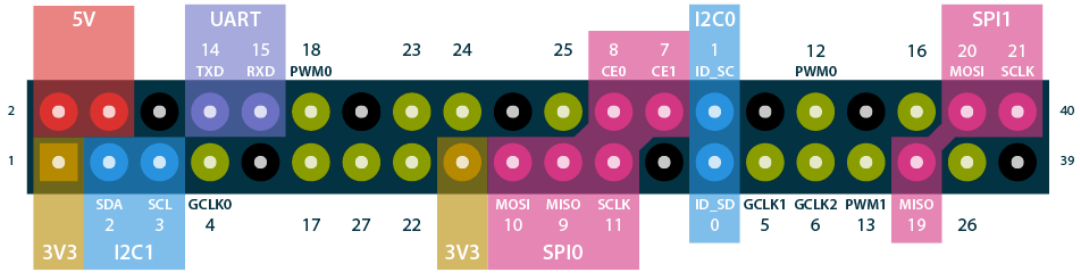
- Zredukowana do minimum liczba rozkazów.
- Jednakowy format – rozmiar rozkazów.
- Prosta treść operacyjna instrukcji.
- Redukcja trybów adresowania.
- Duża ilość rejestrów.
- Wykonywanie operacji przetwarzania danych tylko na rejestrach.
- Ograniczenie komunikacji procesora z pamięcią do dwóch instrukcji – load i store.
- Ujednolicony czas wykonywana rozkazów.[10]

Architektura RISC powstała w latach 70. i miała wyeliminować wady ówczesnego modelu CISC (Complex Instruction Set Computer), opartego na dużej ilości skomplikowanych rozkazów, z których duża część mogła operować na rejestrach i pamięci.[11]

Procesory pracujące w architekturze ARM charakteryzują się lepszym wykorzystaniem czasu procesora w porównaniu do mocniejszych układów pracujących według starego modelu CISC, co pozwala na zmniejszenie ich częstotliwości taktowania, rozmiarów jak i ceny. Czynniki te spowodowały, że procesory ARM znalazły powszechne zastosowanie w szerokiej gamie produktów, co potwierdza suma sprzedaży układów opartych na ARM wynosząca 200 miliardów.[12]

1.1.6. Port GPIO

Do komunikacji Raspberry Pi z zewnętrznymi układami elektronicznymi służą piny wejścia-wyjścia ogólnego przeznaczenia, które zgrupowane są w jeden port GPIO, złożony z 40 wyprowadzeń. Liczba programowalnych pinów wynosi 28, ponieważ 4 z nich są na stale podłączone do zasilania (3.3 V lub 5 V), a 8 jest uziemionych. Wszystkie pozostałe wyprowadzenia mogą być zaprogramowane według indywidualnych potrzeb, a drugorzędną funkcją części dostępnych wyprowadzeń jest obsługa komunikacji UART, I2C i SPI lub generowanie sygnałów PWM. Programista może kontrolować stan (wysoki/niski) oraz kierunek (wejście/wyjście) danego wyprowadzenia. Piny GPIO pracują z maksymalnym napięciem 3.3 V i maksymalnym obciążeniem prądowym 16 mA.[13]



Rys. 3. Port GPIO Raspberry Pi 2 Model B v1.1

Kolorem czarnym zaznaczono uziemione piny, a kolorem zielonym piny wejścia-wyjścia ogólnego przeznaczenia

1.2. Arduino

1.2.1. Historia

Pierwowzór Arduino – Wiring powstał w 2003 roku jako rezultat projektu zawartego w pracy magisterskiej Hernando Barragána, studiującego w Interaction Design Institute we włoskim mieście Ivrea. Głównym celem projektu było stworzenie platformy, która ułatwiałaby wykorzystanie układów elektronicznych przez ludzi nieposiadających wykształcenia w dziedzinie elektroniki. Najważniejszymi elementami Wiring było stworzenie: układu opartego o mikrokontroler ATmega128, przystępnego języka programowania mikrokontrolera oraz zintegrowanego środowiska programistycznego (IDE). Dodatkowo układ sprzętowy jak i oprogramowanie oparte było na licencji open-source. Wraz z upływem czasu kolejne grupy studentów z Interaction Design Institute były nauczane obsługi układu mikroprocesorowego i IDE Wiring, ze względu na niższą cenę w stosunku do ówcześnie istniejących rozwiązań. W 2005 roku Massimo Banzi (który nadzorował pracę Hernando Barragána w ramach tworzenia Wiring), David

Cuartielles oraz David Mellis (ówczesny student Interaction Design Institute) na podstawie kodu źródłowego Wiring dodały wsparcie platformy dla tańszego mikrokontrolera ATmega8. Projekt ten został nazwany Arduino, a do grupy jego projektantów dołączył Gianluca Martino, który miał zbudować nowy układ, oparty o ATmega8. Podobnie jak Wiring, sprzęt i oprogramowanie Arduino oparte jest na licencji open-source. Niski koszt oraz prostota programowania sprawiły, że platforma cieszy się dużą popularnością, a do tej pory wypuszczono 26 wersji Arduino, przeznaczonych do wykonywania różnych zadań.[14][15]

1.2.2. Arduino IDE i język programowania

Arduino IDE jest wieloplatformowym środowiskiem służącym do pisania i wgrywania programów na płytki Arduino. Powstało ono na podstawie Wiring IDE, które oparte było na Processing IDE. Środowisko wspiera język C i C++ ze składnią zmienioną w niewielkim stopniu. Działający skrypt Arduino wymaga wywołania funkcji setup() wykonywanej jednokrotnie po zasileniu płytki, która służy do ustalania parametrów początkowych oraz funkcji loop() wykonującej operacje w pętli. Napisany przez użytkownika skrypt kompilowany jest do postaci cyklicznie wykonywalnego kodu, a następnie konwertowany do tekstu w postaci szesnastkowej, który wgrywany jest na płytkę poprzez firmware (oprogramowanie sprzętowe na stale w niej zainstalowane).[16]

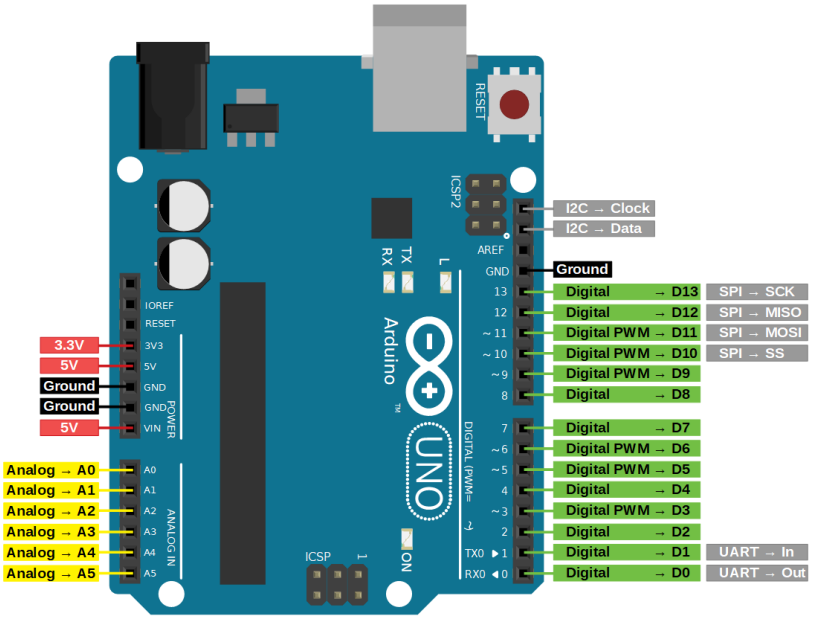
1.2.3. Budowa i wyprowadzenia

Najważniejszymi elementami składowymi Arduino UNO są:

- mikrokontroler AVR ATmega328,
- złącze USB (które jest źródłem zasilania dla płytki, umożliwia wgrywanie na nią napisanego programu poprzez Arduino IDE oraz umożliwia komunikację między płytką, a komputerem do niej podłączonym),
- wyprowadzenia GPIO.

Mikrokontroler AVR ATmega328 o częstotliwości taktowania 16 MHz posiada 31.5 kB nieulotnej pamięci programu (flash), 2kB pamięci danych (SRAM) oraz 1 kB nieulotnej pamięci EEPROM.

Schemat płytki Arduino UNO wykorzystanej w ramach niniejszej pracy inżynierskiej wraz z opisanymi wyprowadzeniami przedstawiony został na poniższej ilustracji:



Rys. 4. Arduino UNO wraz z opisanymi wyprowadzeniami

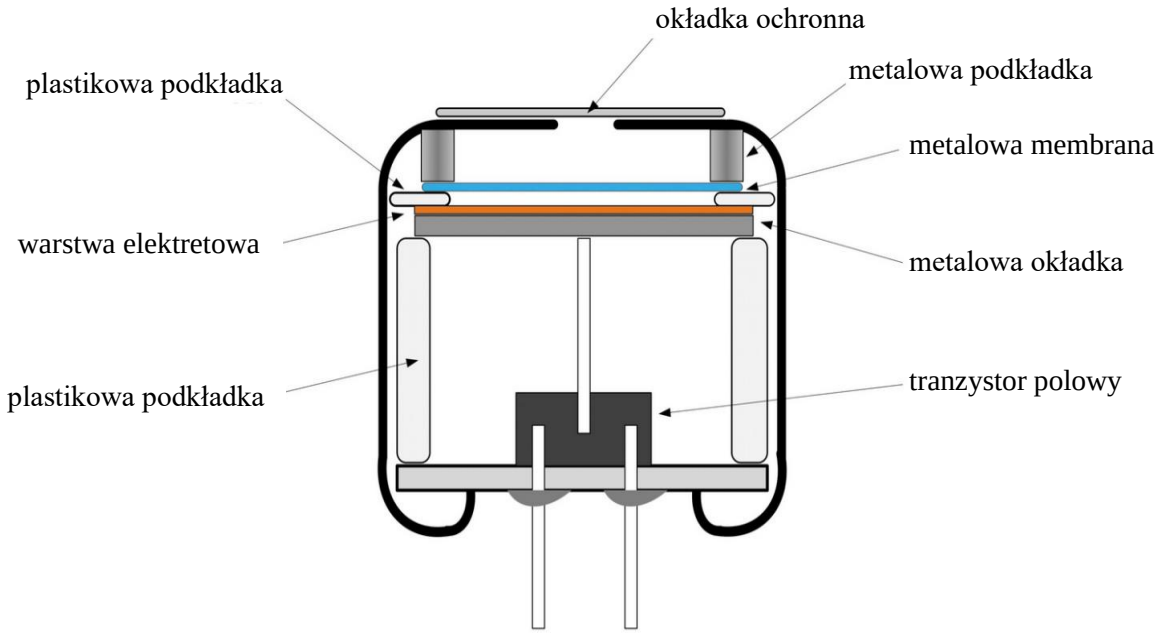
Układ posiada piny zasilające (3.3 V oraz 5 V) oraz piny uziemiające. Do płytki można podłączyć 6 różnych sygnałów analogowych, przetwarzanych za pomocą wbudowanego przetwornika analogowo-cyfrowego o 10-bitowej rozdzielczości. Piny cyfrowe umieszczone po prawej stronie płytki mogą być dowolnie zaprogramowane przez użytkownika (można określać ich stan oraz kierunek – wejście/wyjście), jednak część z nich wykorzystywana jest do generowania przerwań, obsługi komunikacji (UART, I2C, SPI) oraz generowania sygnałów PWM. Cyfrowe wyprowadzenia Arduino UNO pracują z maksymalnym napięciem 5 V i maksymalną obciążeniem prądowym 20 mA.[17]

1.2.4. Mikrokontrolery AVR

Rodzina mikrokontrolerów AVR, z których spora część znajduje zastosowanie w systemach wbudowanych i Arduino, jest rozwijana przez firmę Atmel od 1996 roku. Układy te oparte zostały na 8-bitowej, zmodyfikowanej architekturze harwardzkiej, w której dane i rozkazy adresuje się z wykorzystaniem tej samej przestrzeni adresowej, jednak przechowywane są one w oddzielnych obszarach pamięci cache. Pamięci flash, EEPROM oraz SRAM są zintegrowane z procesorem, eliminując konieczność korzystania z układów zewnętrznych.[18] Mikroprocesory AVR, podobnie jak układy ARM są maszynami RISC co sprawia, że taktowanie procesora oraz rozmiary urządzeń mogły być zmniejszone zapewniając wysoką wydajność i niższą cenę.

1.3. Mikrofon elektretowy

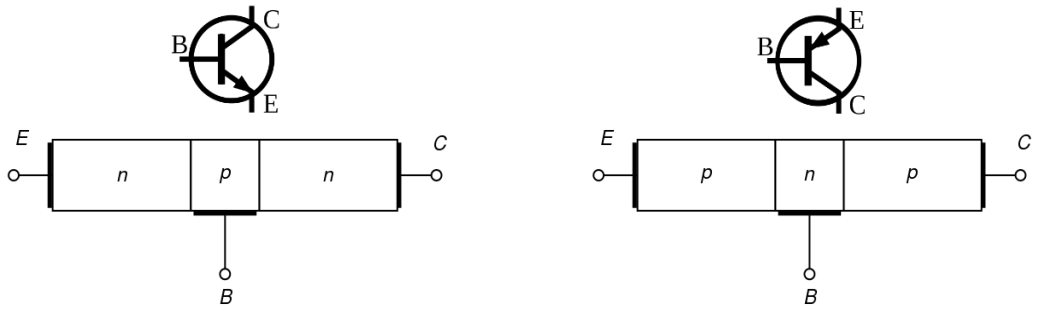
Mikrofon elektretowy jest typem mikrofonu pojemnościowego, w którym fala akustyczna padająca na urządzenie powoduje zmniejszenie odległości pomiędzy okładkami kondensatora, co przekłada się na zmniejszenie jego pojemności. Jedna z jego okładek wykonana jest z elektretu – materiału o stałej polaryzacji elektrycznej, co eliminuje potrzebę dostarczania zewnętrznego napięcia zasilania na okładki kondensatora. Pojemność takiego elementu jest mała, co przekłada się na wysoką impedancję, dlatego urządzenie nie może być łączone z innymi układami elektronicznymi o małej impedancji. W praktyce, w obudowie mikrofonu umieszcza się przedwzmacniacz oparty na tranzystorze polowym, który działa jako układ transformujący impedancję, umożliwiając połączenie mikrofonu z innymi urządzeniami. Sygnał wyjściowy z takiego układu ma niewielką amplitudę, co wymusza zastosowanie wzmacniaczy.[19]



Rys. 5. Budowa mikrofonu elektretowego

1.4. Tranzystor bipolarny

Tranzystor bipolarny jest elementem elektronicznym zbudowanym z trzech warstw półprzewodnika (nazwanych emiter - E, kolektor – C, i baza - B) o różnym typie przewodnictwa: n, gdzie występuje nadmiar wolnych elektronów albo p, gdzie występuje niedostateczna ilość elektronów na powłokach walencyjnych atomów.[20] Ze względu na ułożenie warstw wyróżniamy tranzystory p-n-p oraz n-p-n.



Rys. 6. Symbol oraz uproszczona struktura tranzystora n-p-n (po lewej stronie) oraz p-n-p (po prawej stronie)

Tranzystory bipolarne to elementy sterujące natężeniem prądu przepływającego z zacisku emitera do wyprowadzenia kolektora. Odbywa się to proporcjonalnie do napięcia sterującego przyłożonego do zacisku bazy, działając w ten sposób jak przełącznik sterowany prądem. Za pomocą małego natężenia prądu wejściowego układ steruje większym prądem wyjściowym. Wzmocnienie prądowe tranzystora oznacza się jako β i jest ono stosunkiem prądu kolektora do prądu bazy, a jego wielkość jest zdeterminowana przez konstrukcję układu. Tranzystor może działać w jednym z czterech stanów:

- aktywnym jako wzmacniacz - prąd kolektora jest β razy większy od prądu bazy,
- inwersyjnym - polaryzacja kolektora i emitera odwraca się, wzmocnienie prądowe takiego układu jest kilkukrotnie mniejsze niż w trybie aktywnym,
- nasycenia jako przełącznik - prąd bazy jest na tyle duży, że prąd kolektora przyjmuje stałą, maksymalną wartość,
- zatkania lub odcięcia - prąd kolektora spada do 0.

Otwarcie tranzystora NPN wymaga, by napięcie bazy było wyższe o około 0.7 V od napięcia emitera (jest to napięcie przewodzenia dla diody krzemowej). Jeżeli napięcie na kolektorze także będzie wyższe od napięcia na emiterze, prąd będzie mógł wpływać do kolektora i wypływać przez emiter.

Istnieją trzy podstawowe sposoby podłączenia tranzystora bipolarnego:

- w układzie ze wspólnym emiterem, które realizuje wzmocnienie napięciowe oraz prądowe oraz przesuwa fazę sygnału o 180°,
- w układzie ze wspólnym kolektorem, które realizuje wzmocnienie prądowe,
- w układzie ze wspólną bazą, które realizuje wzmocnienie napięciowe.[21]

W ramach niniejszej pracy stworzono układ wzmacniacza sygnału z mikrofonu elektretowego, oparty na tranzystorze bipolarnym NPN BC547B w układzie ze wspólnym emiterem.

1.5. Przetwornik analogowo-cyfrowy

Przetwornik analogowo - cyfrowy A/C jest układem służącym do zamiany sygnału analogowego (którego dziedzina i zbiór wartości są ciągłe) na sygnał cyfrowy (którego dziedzina i zbiór wartości są dyskretne). Przetwarzanie sygnału wejściowego składa się z 3 etapów:

- próbkowania – wejściowy sygnał ciągły w dziedzinie czasu jest zamieniany na sygnał dyskretny,
- kwantyzacji – ciągły zbiór wartości sygnału dyskretnego ograniczany jest skończonej liczby przedziałów (poziomów kwantowania), zależnej od rozdzielczości przetwornika,
- kodowania – zamiany amplitudy sygnału skwantowanego na słowo (najczęściej binarne), które może być zinterpretowane przez układy cyfrowe.

Dokładność przetwarzania jest ograniczona przez błędy powstające w procesie przetwarzania, które wynikają z charakteru samego przetwornika, mającego cechy zarówno układu analogowego, jak i cyfrowego. Na jakość przetwarzania największy wpływ mają błędy:

- kwantyzacji, który określa różnicę amplitud między sygnałem wejściowym a wyjściowym, a jego wartość maksymalna jest uzależniona od rozdzielczości przetwornika,
- nieliniowości całkowej, który określa odchylenie rzeczywistej charakterystyki przetwarzania od charakterystyki idealnej,
- nieliniowości różniczkowej, który określa różnicę szerokości pomiędzy sąsiednimi poziomami kwantowania.[22]

W ramach niniejszej pracy wykorzystano przetwornik analogowo-cyfrowy MCP3008. Jego najważniejsze parametry to:

- 10-bitowa rozdzielczość,
- 8 kanałów wejściowych,
- maksymalny błąd nieliniowości całkowej: ± 1 LSB (LSB - najmniej znaczący bit),
- maksymalny błąd nieliniowości różniczkowej: ± 1 LSB,
- napięcie zasilanie w zakresie od 2.7 V do 5 V,
- możliwość komunikacji poprzez SPI,
- maksymalna częstotliwość sygnału taktującego wynosząca 3.6 MHz.[23]

Układ MCP3008 jest przetwornikiem z sukcesywną aproksymacją (próbkowaniem bitowym), która działa na zasadzie iteracyjnego porównywania wartości napięcia wejściowego z napięciem odniesienia, wytworzonym za pomocą wewnętrznego przetwornika cyfrowo-analogowego w procesie obsługiwanym przez układ sterujący. Algorytm działania układu sterującego polega na ustawianiu kolejnych bitów słowa danych dla przetwornika C/A poczynając od najważniejszego bitu słowa. W przypadku kiedy napięcie wejściowe będzie mniejsze od napięcia odniesienia z przetwornika C/A, to dany bit słowa danych jest zerowany, a w przeciwnym wypadku pozostaje ustawiony. Następnie realizowana jest kolejna iteracja algorytmu, aż do osiągnięcia ostatniego bitu słowa danych. Tak ustawione słowo danych jest reprezentacją cyfrową napięcia wejściowego. Ze względu na iteracyjny charakter pracy przetwornika, jego częstotliwość próbkowania jest znacząco mniejsza od uzyskiwanej w przetwornikach o przetwarzaniu bezpośrednim (nazywanych przetwornikami flash), których rozdzielczość jest ograniczona i w praktyce rzadko przekracza 10 bitów.[24]

1.6. SPI

SPI (ang. Serial Peripheral Interface) jest jednym z najczęściej stosowanych interfejsów komunikacyjnych w systemach opartych o mikrokontrolery. Pozwala on na jednoczesną komunikację w dwóch kierunkach pomiędzy dwoma urządzeniami (full-duplex). Magistrala SPI złożona jest z 4 linii przesyłających sygnały:

- SCLK – sygnał zegarowy (taktujący) generowany przez urządzenie nadrzędne (nazwane masterem),
- MOSI (Master Out Slave In) – dane wysyłane przez urządzenie nadrzędne do podrzędnego (nazwanego slavem),
- MISO (Master In Slave Out) – dane wysyłane przez urządzenie podrzędne do nadrzędnego,
- SS (Slave Select) – sygnał, którego stan decyduje o rozpoczęciu oraz podtrzymywaniu komunikacji pomiędzy masterem, a danym urządzeniem podrzędnym.

Rys. 7. Schemat magistrali SPI umożliwiającej komunikację pomiędzy dwoma urządzeniami

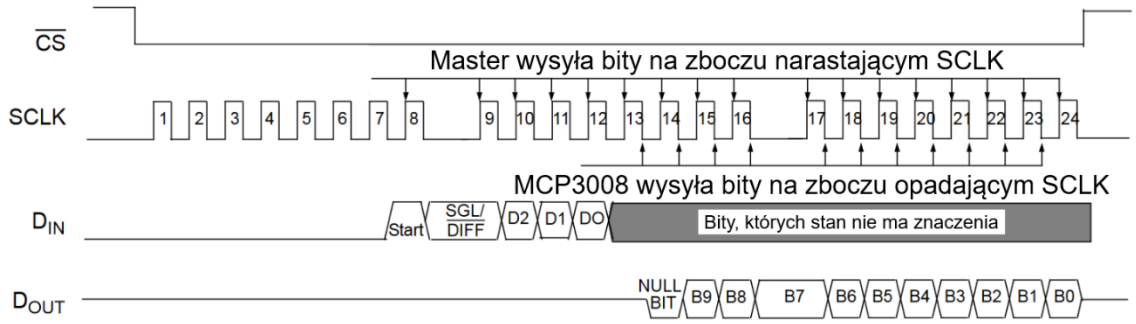
SPI daje możliwość komunikacji jednego urządzenia nadrzędnego z kilkoma urządzeniami podrzędnymi, w takim wypadku linie SCLK, MOSI i MISO wszystkich układów mogą być połączone, ale należy zadbać o to, aby master w danej chwili komunikował się tylko z jednym slavem, poprzez odpowiedni wybór aktywowanej linii SS.[25]

1.7. Komunikacja z przetwornikiem A/C MCP3008

W praktyce komunikacja pomiędzy dwoma urządzeniami z wykorzystaniem magistrali SPI opiera się na jednoczesnym wysyłaniu bajtów poprzez oba układy. Dla wykorzystanego przetwornika MCP3008 komunikacja polega na:

- wysyłaniu rozkazów przez mastera, wskazującego który kanał wejściowy ma zostać przetworzony do postaci cyfrowej,
- wysyłaniu słowa przez przetwornik MCP3008, w którym zawarta jest wartość napięcia wskazanego sygnału wejściowego, zakodowana binarnie.

Schemat tej komunikacji zawarty jest w dokumentacji technicznej dostarczonej przez producenta przetwornika.

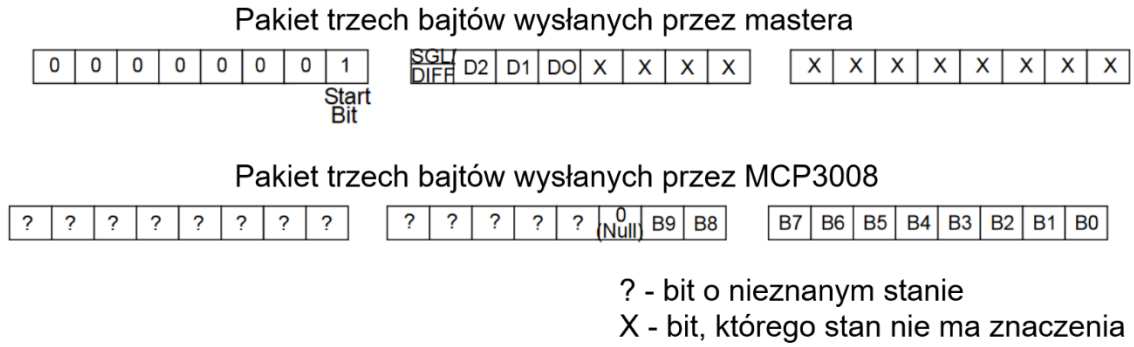


Rys. 8. Schemat komunikacji SPI z przetwornikiem A/C MCP 3008

Master wysyła bity na zboczu narastającym sygnału taktującego SCLK, korzystając z linii D_{IN} (MOSI). Przetwornik wysyła bity na zboczu opadającym SCLK, poprzez wykorzystanie linii D_{OUT} (MISO). Taka konfiguracja odpowiada pracy SPI w trybie 0. Zmiana stanu z wysokiego na niski na linii CS (SS) rozpoczyna komunikację. Następnie master wysyła bit startu w stanie wysokim, poprzedzony 7 bitami wiodącymi w stanie niskim. Następny bit SGL/$\overline{\text{DIFF}}$ wysłany przez mastera określa typ odczytu (stan wysoki - odczytanie napięcia z pojedynczego kanału, stan niski – odczytanie różnicy napięć pomiędzy dwoma kanałami). Kombinacja kolejnych 3 bitów wysłanych przez mastera określa wybrany kanał (np. kanał ósmy dla kombinacji bitów D2=1, D1=1, i D0=1). Następnie układ dokonuje konwersji analogowego sygnału wejściowego na podanym

kanale i po upływie 2 taktów sygnału taktującego SCLK przetwornik wysyła bit NULL (w stanie niskim). Następnie MCP3008 wysyła wartość przetworzonego napięcia wejściowego zakodowaną binarnie na 10 bitach, w kolejności od najbardziej do najmniej znaczącego bitu (MSB-LSB). Zakończenie komunikacji realizowane jest poprzez ustawienie linii CS ze stanu niskiego na stan wysoki.

Schemat praktycznej komunikacji pomiędzy masterem a MCP3008, polegającej na równoczesnym wysłaniu i odbieraniu 3 bajtów poprzez oba urządzenia, został zaprezentowany poniżej.



Rys. 9. Schemat wymiany bajtów pomiędzy MCP3008 a masterem

Od strony praktycznej algorytm pozwalający na odczytanie wartości napięcia pojedynczego sygnału analogowego podanego na kanał przetwornika polega na:

1. Ustawieniu przez mastera (Arduino UNO lub Raspberry Pi) stanu niskiego na linii SS.
2. Wysłaniu przez mastera bajtu startu (00000001).
3. Wysłaniu przez mastera bajtu określającego typ odczytu oraz numer wybranego kanału (SGL/$\overline{\text{DIFF}}$=1|D2|D1|D0|X|X|X|X). Jednocześnie master odbiera bajt przesłany przez MCP3008, którego 2 ostatnie bity (B9|B8) są dwoma najbardziej znaczącymi bitami słowa 10-bitowego, określającego wartość przetworzonego napięcia wejściowego.
4. Odebraniu przez mastera bajtu, zawierającego pozostałe 8 bitów słowa (B7|B6|B5|B4|B3|B2|B1|B0).
5. Ustawieniu przez mastera stanu wysokiego na linii SS.

Uzyskane słowo 10-bitowe opisuje napięcie wejściowe podane na kanał przetwornika zgodnie z równaniem:

$$S = \frac{1024 \cdot V_{IN}}{V_{REF}} \tag{0}$$

gdzie:

S – słowo 10-bitowe będące wynikiem przetwarzania A/C,

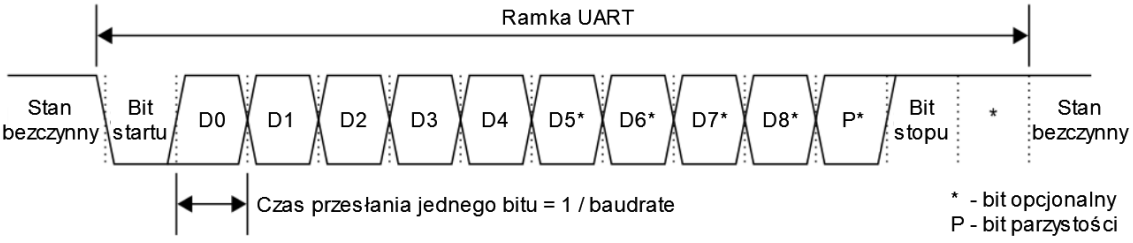
V_{IN} – wartość napięcia sygnału analogowego, podana na kanał przetwornika,

V_{REF} – napięcie referencyjne, określające zakres przetwarzania MCP3008.[26]

1.8. UART

Uniwersalny asynchroniczny nadajnik-odbiornik (ang. universal asynchronous receiver-transmitter) jest układem umożliwiający komunikację między dwoma urządzeniami poprzez ich porty szeregowe. Wymiana danych opiera się na przesyłaniu sygnałów dwoma liniami: Rx – odbioru danych oraz Tx – przesyłania danych. Format oraz prędkość transmisji są konfigurowalne. Dane przesyłane są w ramkach o konfigurowalnym formacie, składających się z:

- bitu startu w stanie niskim, który rozpoczyna transmisję (kiedy UART nie przesyła danych, linia Tx jest w stanie wysokim),
- bitów danych – od 5 do 8 bitów (lub do 9 bitów jeśli bit parzystości nie będzie używany) ułożonych w kolejności LSB-MSB lub MSB-LSB,
- bitu parzystości, na podstawie którego UART odbiornika stwierdza, czy otrzymane dane są poprawne – jeśli bit parzystości jest równy 0, to liczba bitów danych w stanie wysokim powinna być parzysta (jeśli jest inaczej, to transmisja była obarczona błędem), analogicznie jeśli bit parzystości jest równy 1, to liczba bitów danych w stanie wysokim dla poprawnie przeprowadzonej transmisji powinna być nieparzysta,
- bitu (lub 2 bitów) stopu w stanie wysokim, kończącym transmisję.

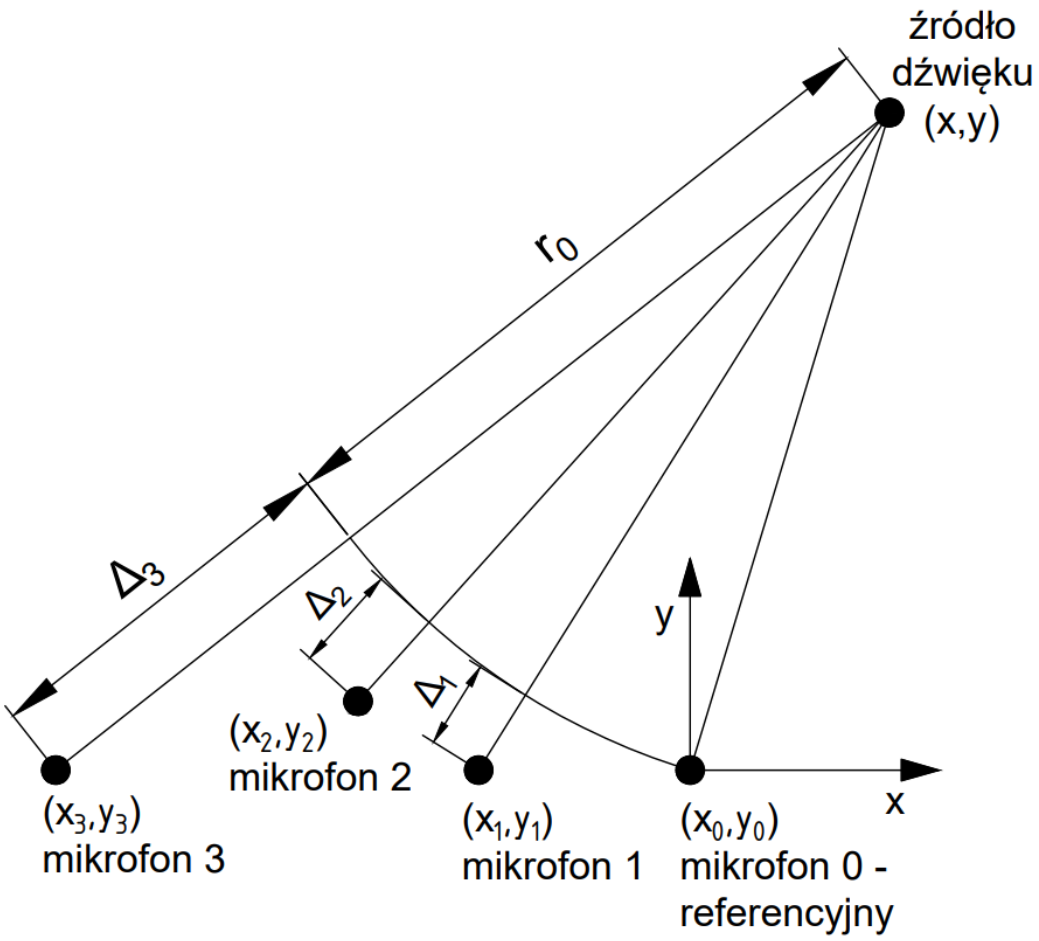


Rys. 10. Format ramki UART

Aby poprawnie przeprowadzić komunikację UART oba urządzenia muszą przesyłać i odbierać dane w tak samo zdefiniowanych ramkach. Z uwagi na asynchroniczność procesu oba urządzenia muszą zostać skonfigurowane do pracy z jednakową prędkością transmisji danych, nazwaną baud rate. Parametr ten jest miarą prędkości przesyłania zmian medium transmisyjnego, jednak dla urządzeń cyfrowych prędkość ta jest tożsama z prędkością transmisji danych.[27]

1.9. Algorytm wyznaczania położenia źródła dźwięku

Na podstawie sygnałów z mikrofonów wyznaczone zostaną współrzędne x oraz y odpowiadające położeniu źródła dźwięku w dwuwymiarowym, kartezjańskim układzie współrzędnych. Schemat rochodzenia się fali aktustycznej dla badanego układu wygląda następująco:



Rys. 11. Schemat rozchodzenia się fali dźwiękowej

Na podstawie twierdzenia Pitagorasa opisano model matematyczny zjawiska:

$$\begin{cases} r_0^2 = (x - x_0)^2 + (y - y_0)^2 & (1) \\ (r_0 + \Delta_1)^2 = r_0^2 + 2r_0\Delta_1 + \Delta_1^2 = (x - x_1)^2 + (y - y_1)^2 & (2) \\ (r_0 + \Delta_2)^2 = r_0^2 + 2r_0\Delta_2 + \Delta_2^2 = (x - x_2)^2 + (y - y_2)^2 & (3) \\ (r_0 + \Delta_3)^2 = r_0^2 + 2r_0\Delta_3 + \Delta_3^2 = (x - x_3)^2 + (y - y_3)^2 & (4) \end{cases}$$

gdzie:

x, y – nieznane współrzędne położenia źródła dźwięku,

r_0 – nieznana odległość źródła dźwięku od mikrofonu referencyjnego,

$x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3$ – znane współrzędne położenia mikrofonów.

Wiadomo, że:

$$\Delta_n = \tau_n \cdot v_{dź} \quad (5)$$

gdzie:

τ_n – przesunięcie czasowe (opóźnienie) sygnału z n-tego mikrofonu, względem sygnału z mikrofonu referencyjnego 0, dla $n = 1, 2, 3$,

$v_{dź}$ – prędkość rozchodzenia się fali akustycznej w powietrzu.

Opóźnienia τ_1, τ_2, τ_3 pozwalające obliczyć odpowiednio Δ_1, Δ_2 oraz Δ_3 zostały wyznaczone na podstawie dwóch metod, z czego jedna bazuje na funkcji korelacji wzajemnej sygnału z mikrofonu referencyjnego i sygnałów z pozostałych mikrofonów.

Rozmieszczenie mikrofonów nie ma wpływu na wyznaczone położenie źródła dźwięku, jedyne ograniczenie polega na tym, że aby układ równań był rozwiązywalny wszystkie wykorzystane mikrofony nie mogą leżeć na jednej prostej. Wybór mikrofonu referencyjnego również nie ma wpływu na wynik. Liczba równań w układzie równań (1), (2), (3), (4) jest równa liczbie użytych mikrofonów. Liczba niewiadomych wynosi 3, dlatego układ byłby rozwiązywalny przy zastosowaniu 3 mikrofonów. Dodanie czwartego mikrofonu uprościło rozwiązywanie układu równań – zamiast 3 równań kwadratowych rozwiązano 3 równania liniowe.

Wstawiono r_0^2 z równania (1) do równań (2), (3) i (4) oraz wymnożono:

$$\begin{cases} \cancel{x^2} - 2xx_0 + x_0^2 + \cancel{y^2} - 2yy_0 + y_0^2 + 2r_0\Delta_1 + \Delta_1^2 = \cancel{x^2} - 2xx_1 + x_1^2 + \cancel{y^2} - 2yy_1 + y_1^2 & (6) \\ \cancel{x^2} - 2xx_0 + x_0^2 + \cancel{y^2} - 2yy_0 + y_0^2 + 2r_0\Delta_2 + \Delta_2^2 = \cancel{x^2} - 2xx_2 + x_2^2 + \cancel{y^2} - 2yy_2 + y_2^2 & (7) \\ \cancel{x^2} - 2xx_0 + x_0^2 + \cancel{y^2} - 2yy_0 + y_0^2 + 2r_0\Delta_3 + \Delta_3^2 = \cancel{x^2} - 2xx_3 + x_3^2 + \cancel{y^2} - 2yy_3 + y_3^2 & (8) \end{cases}$$

Oddzielono dane od niewiadomych:

$$\begin{cases} x(2x_1 - 2x_0) + y(2y_1 - 2y_0) + 2r_0\Delta_1 = x_1^2 + y_1^2 - x_0^2 - y_0^2 - \Delta_1^2 & (9) \\ x(2x_2 - 2x_0) + y(2y_2 - 2y_0) + 2r_0\Delta_2 = x_2^2 + y_2^2 - x_0^2 - y_0^2 - \Delta_2^2 & (10) \\ x(2x_3 - 2x_0) + y(2y_3 - 2y_0) + 2r_0\Delta_3 = x_3^2 + y_3^2 - x_0^2 - y_0^2 - \Delta_3^2 & (11) \end{cases}$$

Zastąpiono znane wartości parametrami:

$$\begin{cases} Ax + By + Cr_0 = D & (12) \\ Ex + Fy + Gr_0 = H & (13) \\ Ix + Jy + Kr_0 = L & (14) \end{cases}$$

Ostatecznie:

$$\begin{cases} y = \dfrac{1}{B}(D - Cr_0 - Ax) & (15) \\ r_0 = \dfrac{B}{GB - FC}\left[H - \dfrac{FD}{B} + x\left(\dfrac{FA}{B} - E\right)\right] & (16) \\ x = \dfrac{L - \dfrac{JD}{B} - \dfrac{B}{GB - FC}\left(K - \dfrac{JC}{B}\right)\left(H - \dfrac{FD}{B}\right)}{I - \dfrac{JA}{B} + \dfrac{B}{GB - FC}\left(K - \dfrac{JC}{B}\right)\left(\dfrac{FA}{B} - E\right)} & (17) \end{cases}$$

1.10. Korelacja wzajemna sygnałów

Funkcja korelacji wzajemnej pomiędzy sygnałami ciągłymi $x(t)$ i $y(t)$ definiowana jest jako:

$$R_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t) \cdot y(t - \tau)\, dt \qquad (18)$$

R_{xy} jest iloczynem skalarnym dwóch sygnałów w funkcji przesunięcia jednego z nich. W celu wyznaczenia funkcji korelacji drugi sygnał opóźnia się w stosunku do pierwszego o czas τ, a następnie oba sygnały wymnaża się przez siebie i całkuje ich iloczyn. W taki sposób dla każdego τ otrzymuje się wartość, mówiącą na ile opóźniony drugi sygnał jest podobny do sygnału pierwszego – skorelowany z nim. Wartość τ, dla którego korelacja sygnałów pochodzących z dwóch różnych mikrofonów osiąga maksimum, informuje o czasie opóźnienia pomiędzy sygnałami. Znając czas opóźnienia między sygnałami τ oraz prędkość rozchodzenia się fali akustycznej w powietrzu, można wyznaczyć

dystans, jaki fala dźwiękowa przebyła od chwili zarejestrowania przez mikrofon referencyjny, do momentu zarejestrowania na pozostałych mikrofonach. Dystans ten oznaczono jako Δ i opisano równaniem (5).

Funkcja korelacji wzajemnej pomiędzy sygnałami dyskretnymi $x[t]$ i $y[t]$ definiowana jest jako:

$$R_{xy}(k) = \sum_{n=-\infty}^{+\infty} x[n] \cdot y[n-k] \tag{19}$$

W praktyce nie dysponuje się nieskończoną ilością próbek sygnału, dlatego wartości funkcji $R_{xy}(k)$ estymuje (przybliża) się na podstawie dostępnej liczbie próbek N za pomocą zależności: $(-N+1 \leq k \leq N-1)$

$$R_{xy}(k) = \sum_{n=0}^{N-1-|k|} x[n] \cdot y[n-k] \tag{20}$$

$$R_{xy}(k) = \frac{1}{N} \sum_{n=0}^{N-1-|k|} x[n] \cdot y[n-k] \tag{21}$$

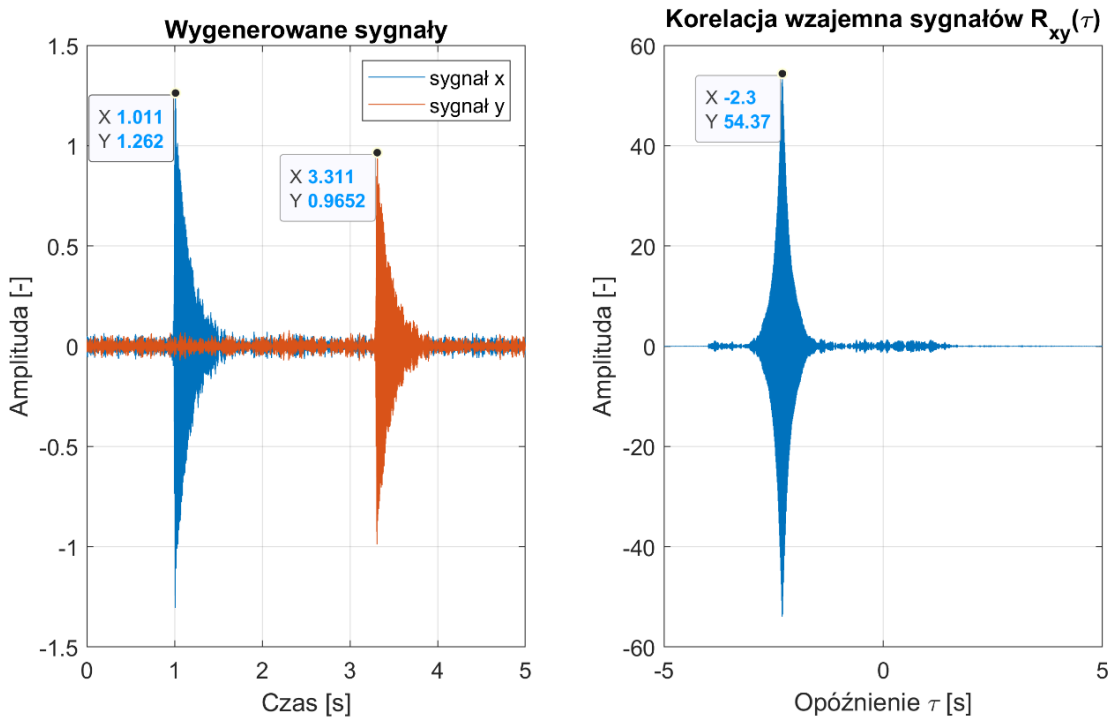
$$R_{xy}(k) = \frac{1}{N-|k|} \sum_{n=0}^{N-1-|k|} x[n] \cdot y[n-k] \tag{22}$$

W zależności od wartości przesunięcia k liczba sumowanych, niezerowych elementów jest różna, ponieważ liczba dostępnych próbek jest skończona. Wraz ze wzrostem $|k|$ ilość dodawanych składników maleje, maleje więc również wartość $R_{xy}(k)$, dana wzorem (20). Stąd wprowadza się normowanie tej wartości dzieląc funkcję (20) przez całkowitą liczbę próbek N i uzyskując równanie (21) albo poprzez uwzględnienie aktualnej liczby składników $|k|$, otrzymując zależność (22). Korelację liczoną na podstawie wzoru (20) nazywa się nieunormowaną, zaś te wyznaczone ze wzorów (21) i (22) – unormowanymi. Estymator funkcji korelacji opisany równaniem (21) jest obciążony, a estymator opisany zależnością (22) nieobciążony.[28]

2. Część praktyczna

2.1. Właściwości funkcji korelacji wzajemnej sygnałów

W celu zbadania właściwości korelacji wzajemnej wygenerowano dwa impulsowe sygnały o gasnących oscylacjach oraz dodano do nich szum, a następnie wyznaczono funkcję korelacji wzajemnej. Wynik działań został przedstawiony na poniższych wykresach.

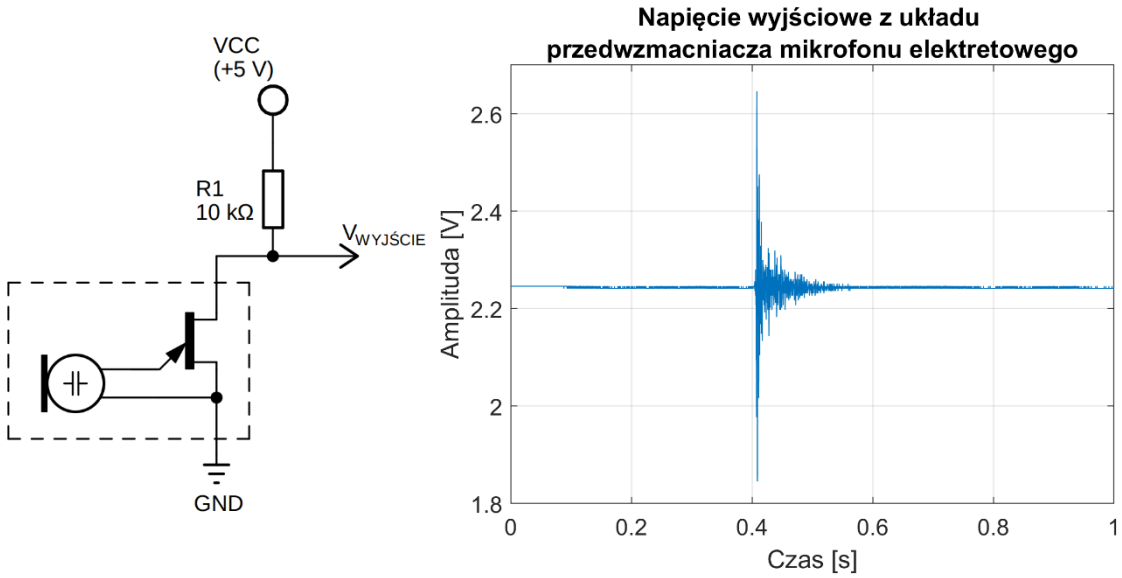


Rys. 12. Wygenerowane sygnały oraz ich korelacja wzajemna

Sygnał y jest opóźniony w stosunku do sygnału x o 2.3 sekundy. Funkcja korelacji wzajemnej sygnałów x i y przyjmuje maksimum dla opóźnienia τ równego -2.3 sekundy. Znak „-" oznacza, że sygnał y jest opóźniony w stosunku do sygnału x, analogicznie znak „+" oznaczałby, że to sygnał x jest opóźniony w stosunku do sygnału y. Aby zachować logiczny sens znaku otrzymanego opóźnienia (który powinien być dodatni, jeśli sygnał y jest opóźniony w stosunku do sygnału x), zostanie on zmieniony na przeciwny. Zastosowanie korelacji wzajemnej sygnałów daje informację nie tylko o wartości opóźnienia, ale także o tym, który z sygnałów jest opóźniony. Z tego powodu funkcja korelacji wzajemnej sygnałów zostanie wykorzystana do wyznaczenia opóźnienia pomiędzy sygnałem z mikrofonu referencyjnego, a sygnałami z pozostałych mikrofonów.

2.2. Układ wzmacniający sygnał z mikrofonu elektretowego

W celu zbadania właściwości sygnału pochodzącego z mikrofonu elektretowego podłączono mikrofon z wbudowanym przedwzmacniaczem do źródła zasilania oraz zarejestrowano dźwięk klaśnięcia.



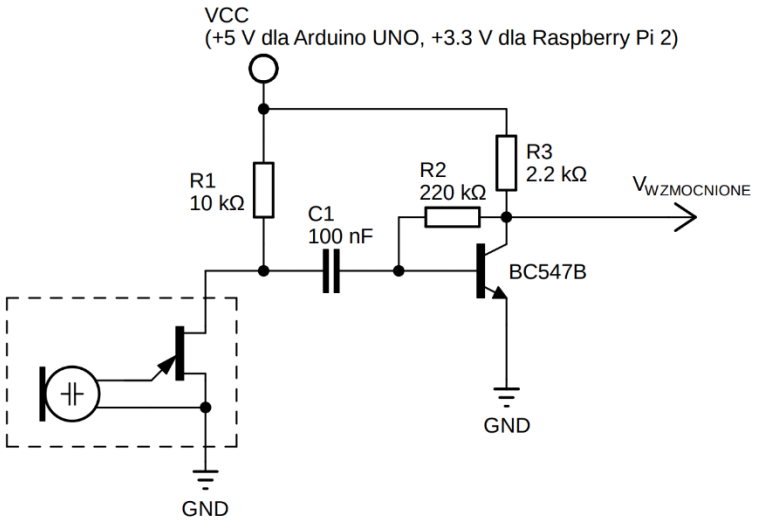
Rys. 13. Schemat układu przedwzmacniacza i jego odpowiedź na falę akustyczną

Rezystor R1 ma za zadanie zmniejszyć napięcie zasilania, które rozpływa się pomiędzy układem przedwzmacniacza mikrofonu oraz wyjściem.

Analizując przebieg uzyskanego sygnału można zauważyć przedział, w którym amplituda sygnału najpierw znacząco rośnie, następnie oscyluje, a na końcu gaśnie. Przebieg ten odpowiada chwili, w których fala akustyczna wprawiła membranę mikrofonu w drgania. Maksymalna wartość napięcia wynosi 2.65 V, a maksymalna wartość napięcia międzyszczytowego jest równa 0.8 V.

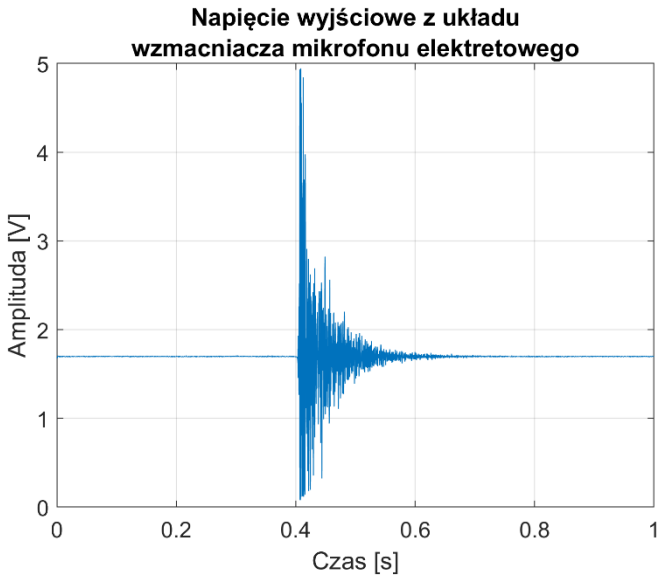
Porównując zarejestrowane napięcia z maksymalnym dopuszczalnym napięciem wynoszącym 3.3 V dla Raspberry Pi 2 oraz 5 V dla Arduino UNO stwierdzono, że sygnał należy wzmocnić tak, aby jak najlepiej wypełnił on przedział od 0 V do maksymalnego dopuszczalnego napięcia, wynoszącego 3.3 V dla Raspberry Pi 2 i 5 V dla Arduino UNO.

W ramach tego zaprojektowano układ wzmacniacza sygnału z mikrofonu elektretowego.



Rys. 14. Schemat zaprojektowanego wzmacniacza

Rezystor R1 ma za zadanie zmniejszyć napięcie zasilania, które rozpływa się pomiędzy układem przedwzmacniacza mikrofonu oraz kondensatorem C1. Zastosowanie kondensatora C1 sprawia, że sygnał pozbawiony jest składowej stałej – oscylacje napięcia są odwzorowaniem drgań membrany mikrofonu. Wzmocnienie sygnału realizowane jest przy pomocy tranzystora bipolarnego NPN BC574B w układzie ze wspólnym emiterem. Na jego bazę podane jest oscylujące napięcie z kondensatora C1, do którego dodano składową stałą poprzez dzielnik napięcia stworzony z rezystorów R2 i R3, zapewniając poprawne otwarcie tranzystora. Napięcie wzmocnionego sygnału wyjściowego z kolektora jest zmniejszone o spadek napięcia na rezystorze R3. Poprzez zmianę oporności rezystora R3 można zmieniać składową stałą wzmocnionego sygnału. Zarejestrowano wzmocnione napięcie wyjściowe towarzyszące dźwiękowi klaśnięcia.

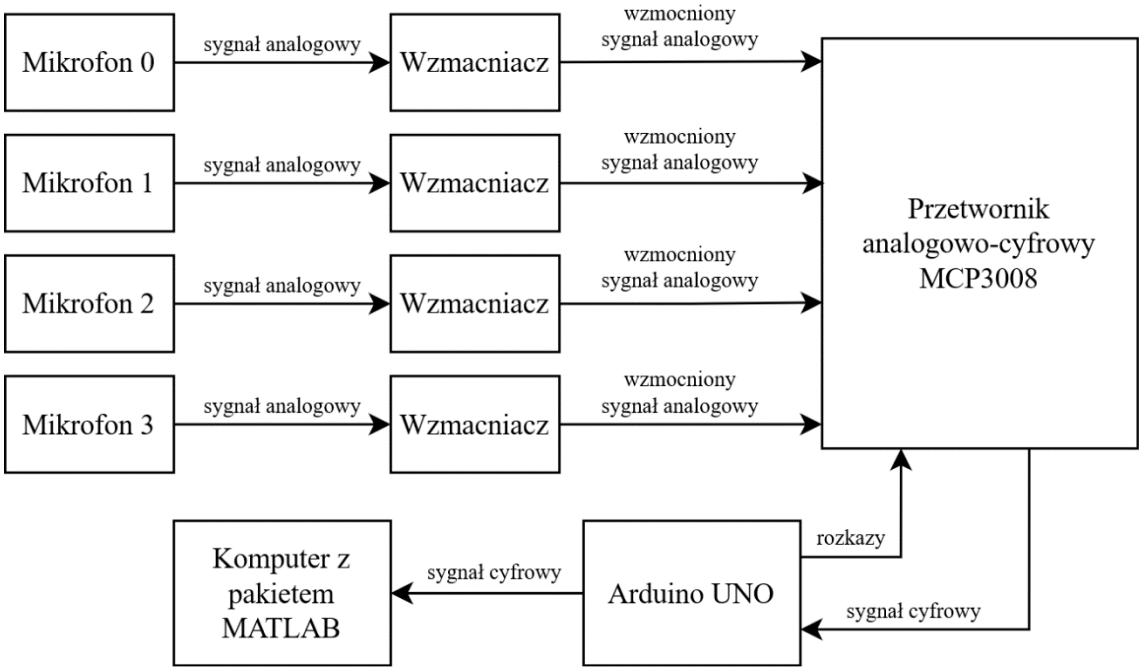


Rys. 15. Odpowiedź zaprojektowanego wzmacniacza na falę akustyczną

Maksymalna wartość napięcia wynosi 4.94 V, a maksymalna wartość napięcia międzyszczytowego jest równa 4.86 V. Co świadczy o tym, że wzmocniony sygnał poprawnie wypełnia cały zakres przetwarzania przetwornika.

2.3. Rejestrowanie sygnałów audio przy pomocy Arduino UNO

Zaprojektowano układ rejestrujący sygnały z mikrofonów i wyznaczający na ich podstawie położenie źródła dźwięku. Projekt opisany jest poniższym schematem blokowym.

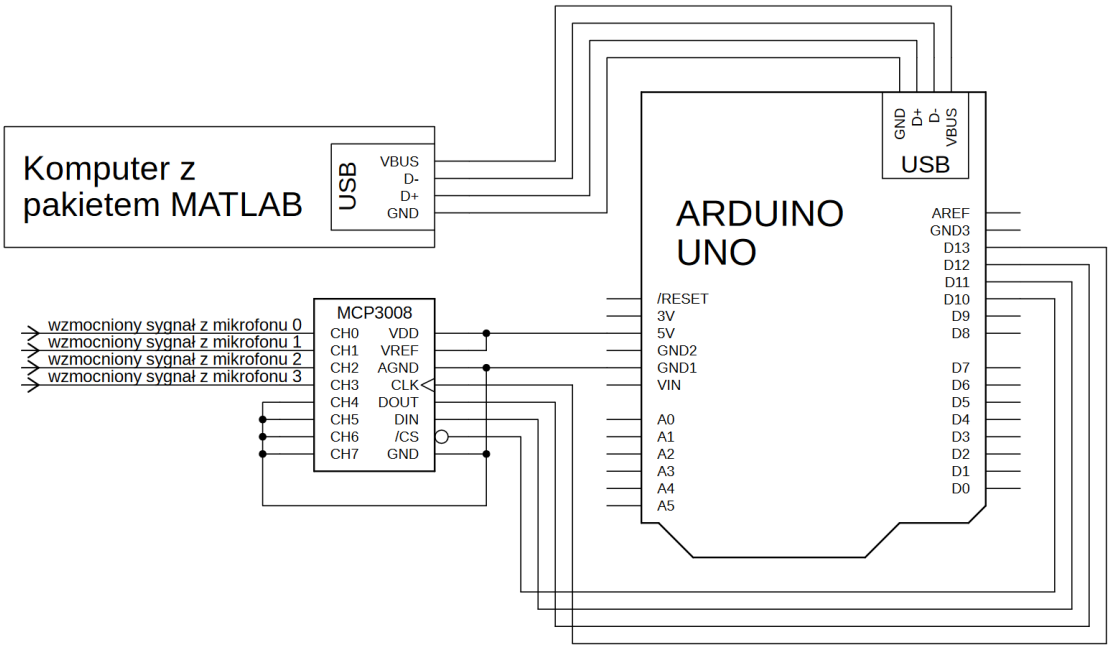


Rys. 16. Schemat blokowy układu do rejestrowania dźwięku na Arduino UNO

Sygnały analogowe z mikrofonów elektretowych zostają wzmocnione, a następnie podłączone do kanałów wejściowych przetwornika analogowo-cyfrowego. Arduino UNO komunikuje się poprzez magistralę SPI z przetwornikiem odbierając dane, w których zawarta jest informacja o amplitudzie 4 spróbkowanych wejść przetwornika. Następnie Arduino UNO przesyła dane poprzez port szeregowy do komputera z pakietem MATLAB®, który na ich podstawie wyznacza położenie źródła dźwięku.

Sygnały z mikrofonów muszą być próbkowane ze stałą częstotliwością, dlatego odbieranie danych przez Arduino UNO i ich wysyłanie odbywa się co stały interwał czasowy.

Połączono układ opisany poniższym schematem elektrycznym.



Rys. 17. Schemat elektryczny układu do rejestrowania dźwięku na Arduino UNO

Sygnały z mikrofonu referencyjnego 0, mikrofonu 1, mikrofonu 2 oraz mikrofonu 3 podłączono odpowiednio do kanału pierwszego (CH0), drugiego (CH1), trzeciego (CH2) oraz czwartego (CH3) przetwornika A/C. Arduino UNO komunikuje się z MCP3008 za pośrednictwem magistrali SPI, a wyprowadzenia D13, D12, D11 i D10 pełnią rolę linii odpowiednio: SCK, MISO, MOSI oraz SS. Wymiana danych pomiędzy urządzeniami taktowana jest sygnałem cyfrowym SCLK. Zgodnie z dokumentacją techniczną przetwornik MCP3008 może być taktowany sygnałem o maksymalnej częstotliwości 3.5 MHz,[26] jednak w trakcie testów uzyskano stabilne przetwarzanie dla częstotliwości nieprzekraczających 4 MHz.

W ramach komunikacji z MCP3008, Arduino UNO odbiera słowa 10-bitowe zawierające informację o amplitudzie sygnałów, podłączonych do kanałów wejściowych przetwornika. Napisano algorytm w środowisku IDE realizujący odbieranie danych, który został opisany poniżej.

```
#include <SPI.h>        //Dołącz bibliotekę SPI.h realizującą komunikację SPI.
/*Ustaw częstotliwość sygnału SCLK na 3.5 MHz, przesyłanie bajtów w kolejności od najbardziej znaczącego bitu i tryb SPI 0 (opisany w podrozdziale 1.7.). /*
SPISettings settings(3500000, MSBFIRST, SPI_MODE0);
//Rozpocznij komunikację poprzez magistralę SPI z zadanymi ustawieniami.
SPI.begin();
```

```
SPI.beginTransaction(settings);
PORTB = PORTB & B11111011;    //Ustaw pin 10 (linia SS) w stan niski.
SPI.transfer(0x01);      //Wyślij do MCP3008 bajt startu – 00000001.
/*Wyślij do MCP3008 bajt określający typ odczytu i wybrany numer kanału (SGL/DIFF|D2|D1|D0|0|0|0|0), który dla odczytu pojedynczego i kanału pierwszego (do którego podłączony jest mikrofon referencyjny 0) jest równy 10000000 (dla kanału drugiego bajt ten byłby równy 10010000, dla trzeciego 10100000, a dla czwartego 10110000). Jednocześnie zapisz w zmiennej bx0 2 najmniej znaczące bity z bajtu odesłanego przez MCP3008, bx0=(0|0|0|0|0|0|B9|B8). */
bx0 = SPI.transfer(0x80)&0x3;
/*Wyślij bajt 0000 0000 do MCP3008. Jednocześnie zapisz w zmiennej bx00 bajt odesłany przez MCP3008, bx00=(B7|B6|B5|B4|B3|B2|B1|B0). Amplituda sygnału z mikrofonu zakodowana jest w słowie 10-bitowym (B9|B8|B7|B6|B5|B4|B3|B2|B1|B0). */
bx00= SPI.transfer(0x0);
//Ustaw pin 10 (linia SS) w stan wysoki. Odczyt kanału został zakończony.
PORTB = PORTB | B00000100;
//Analogicznie odczytaj słowo z kanału drugiego i zapisz je w bajtach bx1 i bx11.
PORTB = PORTB & B11111011;
SPI.transfer (0x01);
bx1 = SPI.transfer (0x90)&0x3;
bx11= SPI.transfer (0x0);
PORTB = PORTB | B00000100;
//Analogicznie odczytaj słowo z kanału trzeciego i zapisz je w bajtach bx2 i bx22.
PORTB = PORTB & B11111011;
SPI.transfer (0x01);
bx2 = SPI.transfer (0xA0)&0x3;
bx22= SPI.transfer (0x0);
PORTB = PORTB | B00000100;
//Analogicznie odczytaj słowo z kanału czwartego i zapisz je w bajtach bx3 i bx33.
PORTB = PORTB & B11111011;
SPI.transfer (0x01);
bx3 = SPI.transfer (0xB0)&0x3;
bx33= SPI.transfer (0x0);
PORTB = PORTB | B00000100;
```

Funkcja **SPI**.transfer zawarta w bibliotece SPI.h wysyła do MCP3008 bajt podany jako jej argument oraz zwraca bajt, który przetwornik wysłał w odpowiedzi.[29] Napisany skrypt jest implementacją algorytmu z podrozdziału 1.7. W ramach działania programu wartości napięcia sygnałów z mikrofonów zapisano w zmiennych: bx0 i bx00 (sygnał

z mikrofonu referencyjnego 0), bx1 i bx11 (sygnał z mikrofonu 1), bx2 i bx22 (sygnał z mikrofonu 2) oraz bx3 i bx33 (sygnał z mikrofonu 3).

Aby zapewnić stałą częstotliwość próbkowania sygnałów Arduino UNO musi odbierać dane z przetwornika MCP3008 ze stałym interwałem czasowym. W ramach tego zdefiniowano 8-bitowy timer-licznik TC2 mikrokontrolera AVR ATmega328, zliczający impulsy rezonatora kwarcowego taktującego układ. Przepełnienie licznika odbywa się ze stałym okresem i generuje przerwanie, w ramach którego przeprowadzono odczyt amplitudy sygnałów z mikrofonów. Działanie licznika T2C skonfigurowano poprzez modyfikację jego rejestrów, z których najważniejsze rolę pełnią:

- TCCR2A, którego bity WGM21 i WGM20 określają tryb pracy,
- TCCR2A, którego bit WGM22 określa tryb pracy, a bity CS22, CS21 oraz CS22 określają wartość preskalera – dzielnika częstotliwości sygnału taktującego licznik,
- TCNT2 przechowujący aktualną wartość licznika,
- OCR2A przechowujący maksymalną wartość licznika, po której nastąpi jego przepełnienie i zresetowanie w trybie pracy CTC,
- TIMSK2, którego bity aktywują obsługę przerwań pochodzących od licznika,
- TIFR2 informujący o nieobsłużonym przerwaniu.

Tryb pracy w którym działa licznik T2C informuje o maksymalnej wartości, po której nastąpi przepełnienie i zresetowanie licznika. W trybie normalnym wartość ta jest równa 255, ponieważ licznik jest 8-bitowy. Poza trybem normalnym istnieje tryb CTC (Clear Timer on Compare), dla którego przepełnienie i zresetowanie licznika następuje kiedy aktualna wartość licznika jest równa wartości rejestru OCR2A (TCNT2=OCR2A). Minimalna częstotliwość przepełnienia 8-bitowego licznika T2C taktowanego sygnałem o częstotliwości 16 MHz wynosi 62.5 kHz (16 MHz / 256). Zastosowanie preskalera pozwala na zmniejszenie częstotliwości przepełniania licznika poprzez dzielenie częstotliwości sygnału taktującego, przykładowo użycie preskalera 8 spowoduje, że licznik taktowany jest sygnałem o ośmiokrotnie niższej częstotliwości 2 MHz.[30][31][32]

Konfiguracja T2C została przeprowadzona w sposób opisany poniżej.

```
TCCR2A = 0;     //Wyzeruj rejestr TCCR2A.
TCCR2B = 0;     //Wyzeruj rejestr TCCR2B.
TCNT2  = 0;     //Zresetuj aktualną wartość licznika.
/*Ustaw bit WGM21, co spowoduje pracę licznika w trybie CTC (WGM22=0, WGM21=1, WGM20=0). */
TCCR2A |= (1 << WGM21);
```

```
//Ustaw wartości bitów CS22, CS21 i CS20 odpowiadające wyborowi preskalera 8.
        TCCR2B = (0 << CS22) | (1 << CS21) | (0 << CS20);
/*Ustaw wartość rejestru OCR2A na 199, dla której licznik przepełnia się z częstotliwością 10 kHz (OCR2A = 16 MHz / (10 kHz * 8) - 1). */
        OCR2A =199;
/*Aktywuj przerwanie kiedy aktualna wartość licznika będzie równa wartości rejestru OCR2A (TCNT2 = OCR2A). */
        TIMSK2 |= (1 << OCIE2A);
```

Ilość dostępnej pamięci na platformie jest niewystarczająca, aby móc przechować całość zarejestrowanych danych. Dlatego Arduino UNO pobierając dane w ramach przerwania dodatkowo wysyła odebrane słowa w formie bajtów do komputera, za pośrednictwem portu szeregowego. Wysłanie danych zostało przeprowadzone w sposób opisany poniżej.

```
//Rozpocznij komunikację UART z prędkością 2 Mb/s.
        Serial.begin(2000000);
/*Wyślij poprzez port szeregowy bajt bx00 zawierający 8 najmniej znaczących bitów słowa określającego napięcie z mikrofonu referencyjnego 0. */
        Serial.write(bx00);
        Serial.write(bx11);     //Analogicznie wyślij bajt bx11 (mikrofon 1).
        Serial.write(bx22);     // Analogicznie wyślij bajt bx22 (mikrofon 2).
        Serial.write(bx33);     // Analogicznie wyślij bajt bx33 (mikrofon 3).
/*2 najbardziej znaczące bity słów dla każdego kanału zostają połączone w jeden bajt (B9|B8 z CH3, B9|B8 z CH2, B9|B8 z CH1, B9|B8 z CH0) oraz wysłane poprzez port szeregowy. */
        Serial.write((bx3<<6) | (bx2<<4) | (bx1<<2) | (bx0));
```

Procedura przerwania od licznika T2C, w ramach której Arduino UNO rejestruje napięcia z mikrofonów elektretowych i wysyła je poprzez port szeregowy do komputera, wywoływana z częstotliwością 10 kHz. Wykonanie procedury 10 000 razy oznacza zarejestrowanie 4 sygnałów trwających 1 sekundę, wystarczających do wyznaczenia położenia źródła dźwięku. W ramach komunikacji szeregowej wysłano w sumie 50 000 bajtów.

Skrypt napisany w MATLABie odbiera bajty przesyłane przez Arduino UNO i konwertuje je do pierwotnej postaci słów 10-bitowych. Zadanie to zostało zrealizowane w sposób opisany poniżej.

```
%Rozpocznij komunikację UART z Arduino UNO podłączonym do portu szeregowego COM3, z prędkością 2 Mb/s.
        s = serialport("COM3",2000000);
```

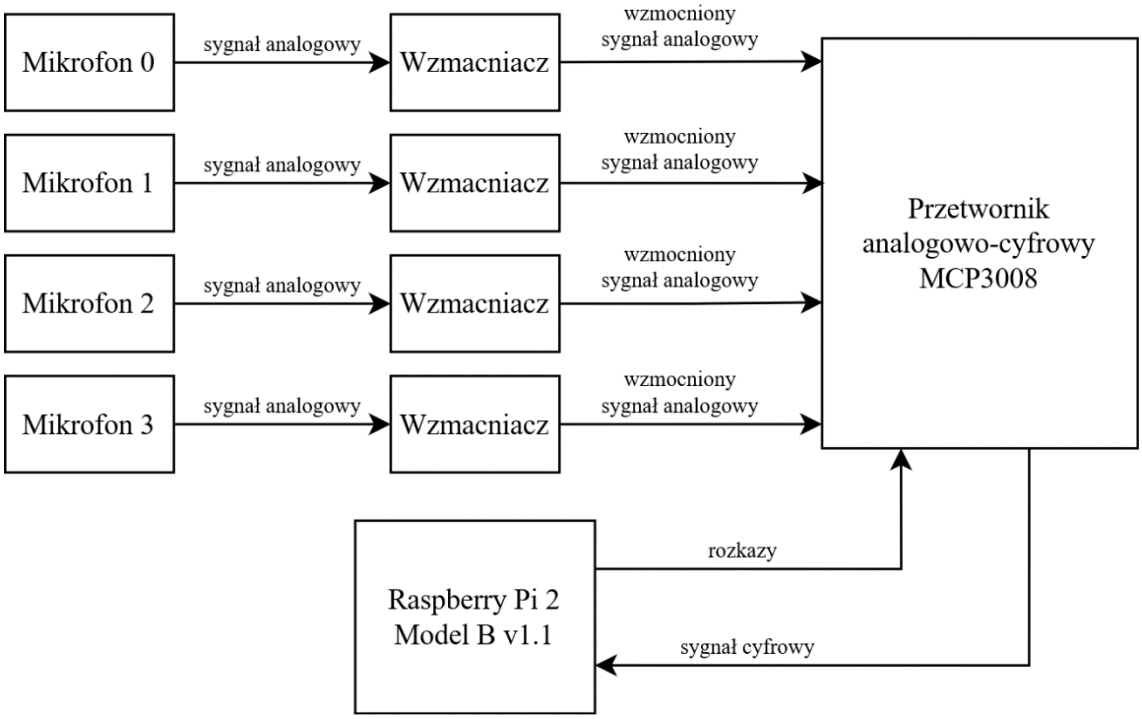
```
        bytesRead=0;                %Liczba przeczytanych do tej pory bajtów.
        dataRead=[];                %Wektor przechowujący kolejno odczytane bajty.
%Kontynuuj czytanie bajtów dopóki nie przeczytano wszystkich przesłanych bajtów.
        while bytesRead<50000
%Odbierz 50 bajtów wysłanych poprzez UART i zapisz je w wektorze.
        dataRead = [dataRead read(s,50,"uint8")];
%Po przeczytaniu bajtów zaktualizuj liczbę aktualnie przeczytanych bajtów.
        bytesRead=bytesRead+50;
        end
%Przekonwertuj przeczytane bajty do słów 16-bitowych (wymagane do wykonania dalszych operacji bitowych).
        data16bit=cast(dataRead,'uint16')
%Stwórz wektor ch0, który przechowuje słowa z zakodowaną wartością napięcia z mikrofonu referencyjnego 0. Ciąg bajtów data16bit(1:5:end) przechowuje wartości ośmiu mniej znaczących bitów słowa (B9|B8|B7|B6|B5|B4|B3|B2|B1|B0). Ciąg bajtów data16bit(5:5:end) przechowuje wartości 2 najbardziej znaczących bitów słów każdego kanału (B9|B8 z CH3, B9|B8 z CH2, B9|B8 z CH1, B9|B8 z CH0). Dokonując operacji bitowych na wyrazach ciągu otrzymane słowo ma postać (0|0|0|0|B10|B9|B8|B6|B5|B4|B3|B2|B1|B0) i jest równe pierwotnemu słowu zakodowanemu przez przetwornik MCP3008.
        ch0=bitor(data16bit(1:5:end),bitshift(bitand(data16bit(5:5:end),0x0003),8));
%Dokonaj analogicznych operacji i stwórz wektor ch1.
        ch1=bitor(data16bit(2:5:end),bitshift(bitand(data16bit(5:5:end),0x000C),6));
%Dokonaj analogicznych operacji i stwórz wektor ch2.
        ch2=bitor(data16bit(3:5:end),bitshift(bitand(data16bit(5:5:end),0x0030),4));
%Dokonaj analogicznych operacji i stwórz wektor ch3.
        ch3=bitor(data16bit(4:5:end),bitshift(bitand(data16bit(5:5:end),0x00C0),2));
%Przekonwertuj słowo na wartość napięcia sygnału z mikrofonu referencyjnego 0
        mic0=cast(ch0*5/1024,'double')
%Przekonwertuj słowo na wartość napięcia sygnału z mikrofonu 1
        mic1=cast(ch1*5/1024,'double')
%Przekonwertuj słowo na wartość napięcia sygnału z mikrofonu 2
        mic2=cast(ch2*5/1024,'double')
%Przekonwertuj słowo na wartość napięcia sygnału z mikrofonu 3
        mic3=cast(ch3*5/1024,'double')
```

W ramach działania algorytmu stworzono wektory: mic0, mic1, mic2 i mic3 przechowujące wartości napięć sygnałów z: mikrofonu 0, mikrofonu 1, mikrofonu 2 i mikrofonu 3.

Wyznaczanie opóźnienia pomiędzy sygnałami zostało opisane w podrozdziale 2.5.

2.4. Rejestrowanie sygnałów audio przy pomocy Raspberry Pi 2

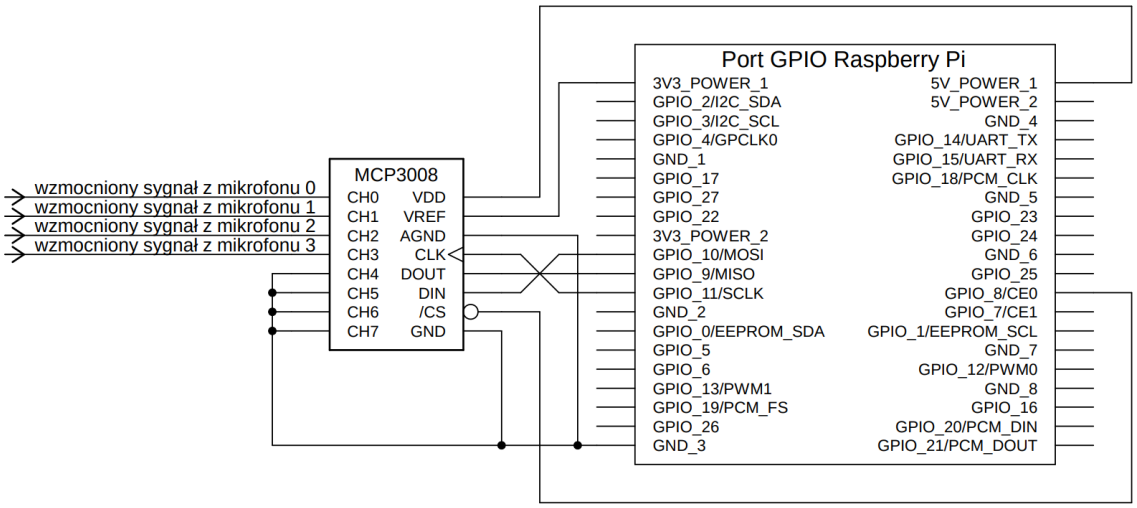
Zaprojektowano układ rejestrujący sygnały z mikrofonów i wyznaczający na ich podstawie położenie źródła dźwięku. Projekt opisany jest poniższym schematem blokowym.



Rys. 18. Schemat blokowy układu do rejestrowania dźwięku na Raspberry Pi 2

Sygnały analogowe z mikrofonów elektretowych zostają wzmocnione, a następnie podłączone do kanałów wejściowych przetwornika analogowo-cyfrowego. Raspberry Pi 2 komunikuje się poprzez magistralę SPI z przetwornikiem odbierając dane, w których zawarta jest informacja o amplitudzie 4 spróbkowanych wejść przetwornika. Minikomputer na podstawie danych wyznacza opóźnienia pomiędzy sygnałami z mikrofonów, a następnie wykorzystuje je do wyznaczenia położenia źródła dźwięku. Sygnały z mikrofonów muszą być próbkowane ze stałą częstotliwością, dlatego odbieranie danych przez Raspberry Pi 2 i ich wysyłanie odbywa się co stały interwał czasowy. W przeciwieństwie do Arduino UNO, Raspberry Pi 2 posiada wystarczającą ilość pamięci pozwalającą na przechowanie kompletu zarejestrowanych danych. Umożliwia to wyznaczenie położenia dźwięku bezpośrednio na platformie i eliminuje konieczność przesyłania danych do komputera.

Połączono układ opisany poniższym schematem elektrycznym.



Rys. 19. Schemat elektryczny układu do rejestrowania dźwięku na Raspberry Pi 2

Sygnały z mikrofonu referencyjnego 0, mikrofonu 1, mikrofonu 2 oraz mikrofonu 3 podłączono odpowiednio do kanału pierwszego (CH0), drugiego (CH1), trzeciego (CH2) oraz czwartego (CH3) przetwornika A/C. Raspberry Pi 2 komunikuje się z MCP3008 za pośrednictwem magistrali SPI, a GPIO 10, GPIO 9, GPIO 11 oraz GPIO 8 pełnią rolę linii odpowiednio: SCK, MISO, MOSI oraz SS. Wymiana danych pomiędzy urządzeniami taktowana jest sygnałem cyfrowym SCLK.

W ramach komunikacji Raspberry Pi 2 odbiera słowa 10-bitowe zawierające informację o amplitudzie sygnałów, podłączonych do kanałów wejściowych przetwornika. Odczyt słowa dla zadanego kanału wejściowego przetwornika odbywa się poprzez wywołanie funkcji, której działanie zostało omówione poniżej.

```
        uint8_t start = 0x01;            //Bajt startu 0000 0001.
//Bajt określający kanał, którego napięcie ma zostać odczytane.
        uint8_t chan = 0x00;
        uint8_t end = 0x00;     //Trzeci bajt 0000 0000
/*Funkcja readChannel wywoływana jest z argumentem channel i zwraca słowo,
w którym zakodowana jest wartość napięcia sygnału z danego kanału MCP3008.
Dla wartości argumentu channel z przedziału 0-7 odczytane zostaną kanały CH0-CH7
przetwornika. */
        int readChannel(uint8_t channel) {
/*Stwórz bufor writeBuf z bajtami, które mają zostać wysłane do MCP3008. Wynikiem
operacji bitowej (0x08|channel) dla argumentu channel przyjmującego wartości
z przedziału 0-3 są bajty: (10000000) umożliwiający odczyt pierwszego kanału
przetwornika, (10010000) umożliwiający odczyt drugiego kanału przetwornika,
```

```
(10100000) umożliwiający odczyt trzeciego kanału przetwornika, (10110000) umożliwiający odczyt czwartego kanału przetwornika. */
        char writeBuf[] = {start, (0x08|channel)<<4,end};
/*Stwórz pusty bufor trzyelementowy, w którym zapisane zostaną 3 bajty wysłane w odpowiedzi przez MCP3008. */
        char readBuf[3];
/*Wyślij do MCP3008 bajty z buforu writeBuf i jednocześnie odbierz od MCP3008 3 bajty i wpisz je do buforu readBuf. */
        bcm2835_spi_transfernb(writeBuf,readBuf,3);
/*Bajt readBuf[2] składa się z 8 mniej znaczących bitów słowa 10-bitowego (B7|B6|B5|B4|B3|B2|B1|B0). Dwa ostatnie bity bajtu readBuf[2] są dwoma najbardziej znaczącymi bitami słowa 10-bitowego (B9|B8). Dokonując operacji bitowych na obu bajtach i otrzymane zostanie słowo (B9|B8|B7|B6|B5|B4|B3|B2|B1| B0), w którym zakodowana jest wartość napięcia z danego kanału przetwornika. Słowo to jest zwrócone przez funkcję. */
        return ((int)readBuf[1] & 0x03) << 8 | (int) readBuf[2];
}
```

Funkcja bcm2835_spi_transfernb(tbuf, rbuf, len) zawarta w bibliotece bcm2835.h[33] wysyła do MCP3008 bufor bajtów tbuf o długości len oraz wpisuje do buforu rbuf bajty, które przetwornik wysłał w odpowiedzi. Napisana funkcja readChannel(channel) jest implementacją algorytmu z podrozdziału 1.7. Wywołanie jej z argumentami channel z przedziału 0-3, spowoduje zwrócenie słów, w których zakodowana jest wartość napięć z kanałów CH0-CH3, do których podłączone zostały sygnały z mikrofonów elektretowych.

Inicjalizację komunikacji SPI przeprowadzono w sposób opisany poniżej.

```
/*Zainicjalizuj działanie biblioteki bcm2835.h i zapewnij jej dostęp do pamięci fizycznej, w tym do pamięci wykorzystywanej przy komunikacji SPI i portów GPIO. */
        if (!bcm2835_init()) {
        printf("Niepowodzenie inicjalizacji bcm2835_init. Wywolaj skrypt jako root.\n");
        return 1;
        }
/*Zainicjalizuj komunikację SPI konfigurując odpowiednie piny GPIO do pracy jako linie MOSI, MISO, SCLK i SS. */
        if (!bcm2835_spi_begin()) {
        printf("Niepowodzenie inicjalizacji SPI.\n");
        return 1;
        }
```

```
//Ustaw przesyłanie bajtów w kolejności od najbardziej znaczącego bitu.
        bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_MSBFIRST);
//Ustaw tryb pracy SPI 0 (opisany w podrozdziale 1.7.).
        bcm2835_spi_setDataMode(BCM2835_SPI_MODE0);
/*Ustal sygnał SCLK jako sygnał taktujący procesora z preskalerem o wartości 64. Dla takiej konfiguracji sygnał SCLK ma częstotliwość 3.9 MHz dla Rasbperry Pi 2. */
        bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_64);
/*Skonfiguruj pin GPIO 8 do pracy jako SS. Komunikacja z MCP3008 będzie odbywać się dla SS w stanie niskim. */
        bcm2835_spi_chipSelect(BCM2835_SPI_CS0);
        bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0, LOW);
```

Aby zapewnić stałą częstotliwość próbkowania sygnałów Raspberry Pi 2 musi odbierać dane z przetwornika MCP3008 ze stałym interwałem czasowym. Dlatego stworzono przerwanie cyklicznie, realizujące odczyt napięć z wejść przetwornika. Przerwanie stworzono w sposób opisany poniżej.

```
        int data[50000];        //Tablica przechowująca zarejestrowane słowa.
        Int dataRead=0;         //Liczba dokonanych pomiarów napięcia sygnałów.
//Deklaracja funkcji wywoływanej w ramach przerwania.
        void readInterrupt (int sig_num);
/*Ustal funkcję readInterrupt jako funkcję obsługującą przerwanie pochodzące od odebrania sygnału SIGALRM. */
        signal(SIGALRM, readInterrupt);
/*Wyślij sygnał SIGALRM po upływie 80 mikrosekund oraz cyklicznie powtarzaj wysyłanie sygnału co 80 mikrosekund. */
        ualarm(80,80);
/*Funkcja readInterrupt wywoływana w ramach obsłużenia przerwania od sygnału SIGALRM rejestruje słowa odpowiadające napięciom sygnałów z wejść przetwornika - sygnałów z mikrofonów. */
        void readInterrupt (int sig_num) {
//Sprawdzenie czy funkcja została wywołana przez sygnał SIGALRM.
        if(sig_num == SIGALRM) {
//Wpisanie słów z odczytu sygnałów z mikrofonów jako kolejnych elementów tablicy data.
        data[dataRead ++]=readChannel(0);
        data[dataRead ++]=readChannel (1);
        data[dataRead ++]=readChannel (2);
        data[dataRead ++]=readChannel (3);
        }
        }
```

Cykliczne rejestrowanie danych zostało zrealizowane poprzez obsługę przerwania pochodzącego od sygnału SIGALRM, który wysyłany jest okresowo co 80 mikrosekund, uzyskując częstotliwość próbkowania sygnałów wynoszącą 12.5 kHz. W praktyce okres pomiędzy wysyłaniem sygnału SIGALRM i jego obsłużeniem może się różnić od zadanej wartości, co związane jest z aktywnością systemu operacyjnego, która nie pozwala na obsłużenie przerwania natychmiastowo w chwili jego otrzymania.[34][35][36]

2.5. Algorytmy wyznaczania opóźnień pomiędzy sygnałami

Do wyznaczenia położenia źródła dźwięku konieczne jest obliczenie opóźnień pomiędzy sygnałem z mikrofonu referencyjnego 0, a sygnałami z pozostałych mikrofonów. Opóźnienia pomiędzy sygnałami wyznaczono na dwa sposoby:

Metoda 1:

Wyznaczono chwile: t_0, t_1, t_2 i t_3, w których sygnały: z mikrofonu referencyjnego 0, mikrofonu 1, mikrofonu 2 i mikrofonu 3 przyjmują wartość maksymalną. Wartość opóźnienia między sygnałami obliczono jako różnice odpowiednich chwil:

$$\tau_1 = t_0 - t_1 \tag{23}$$

$$\tau_2 = t_0 - t_2 \tag{24}$$

$$\tau_3 = t_0 - t_3 \tag{25}$$

Metoda 2:

Opóźnienia pomiędzy odpowiednimi sygnałami wyznaczono poprzez wykorzystanie korelacji wzajemnej sygnałów. Zgodnie z informacjami zawartymi w podrozdziale 2.1. opóźnienie pomiędzy sygnałami jest argumentem, dla którego funkcja korelacji wzajemnej przyjmuje maksimum. W ramach tego wyznaczono opóźnienia jako:

$$\tau_1 = \arg\max R_{01}(\tau) \tag{26}$$

$$\tau_2 = \arg\max R_{02}(\tau) \tag{27}$$

$$\tau_3 = \arg\max R_{03}(\tau) \tag{28}$$

gdzie:

$R_{0i}(\tau)$ – funkcja korelacji wzajemnej sygnału z mikrofonu referencyjnego 0 i sygnału z mikrofonu i, dla $i = 1, 2, 3$.

W ramach obu metod wyznaczono wartości:

τ_1 – opóźnienie pomiędzy sygnałem z mikrofonu referencyjnego 0 i mikrofonu 1,

τ_2 – opóźnienie pomiędzy sygnałem z mikrofonu referencyjnego 0 i mikrofonu 2,

τ_3 – opóźnienie pomiędzy sygnałem z mikrofonu referencyjnego 0 i mikrofonu 3.

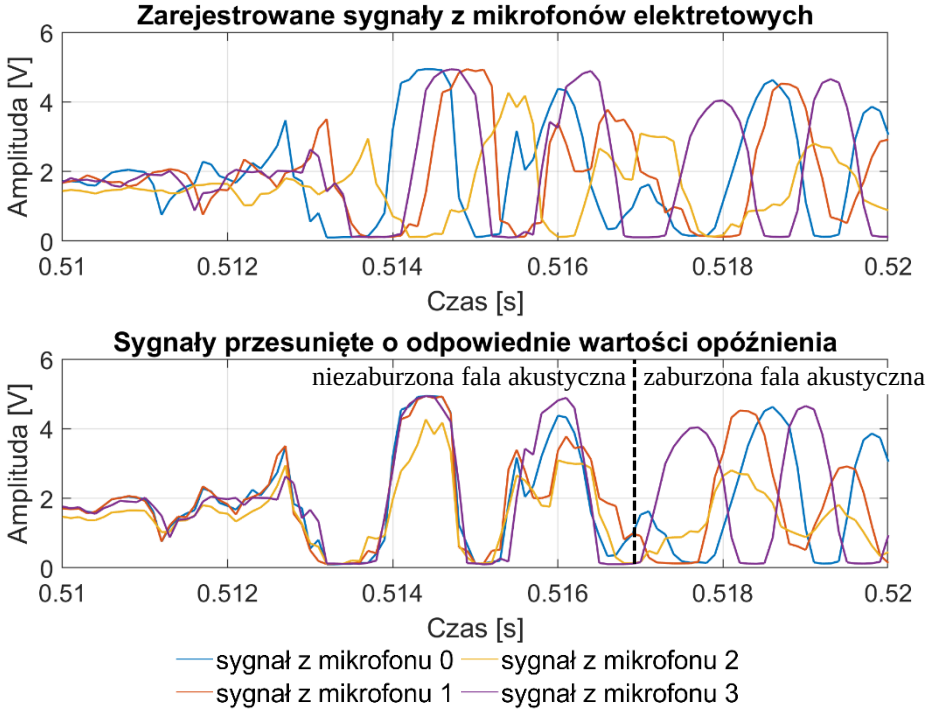
Aby wykluczyć możliwość uzyskania błędnych wyników, korelację wzajemną wyznaczono dla przedziału czasu, w którym mikrofony rejestrowały falę akustyczną pochodzącą od klaśnięcia. Korelowanie całości zarejestrowanych sygnałów wiązałoby się z korelowaniem szumów, co mogłoby przyczynić się do dużego błędu , jeżeli stosunek sygnału do szumu jest niewielki.

Obie metody zostały zaimplementowane w skrypcie napisanym w środowisku MATLAB® (dla sygnałów zarejestrowanych na Arduino UNO) oraz Pythonie (dla sygnałów zarejestrowanych na Raspberry Pi 2). Z uwagi na podobieństwo składni obu algorytmów przeanalizowano jedynie działanie skryptu w MATLABie.

```
%Wektor ch0 przechowuje wartości napięcia sygnału z mikrofonu referencyjnego 0.
%Wektor ch1 przechowuje wartości napięcia sygnału z mikrofonu 1.
%Wektor ch2 przechowuje wartości napięcia sygnału z mikrofonu 2.
%Wektor ch3 przechowuje wartości napięcia sygnału z mikrofonu 3.
        dt=100*10^-6;  %Czas próbkowania sygnałów z mikrofonów.
%Zapisz indeks elementu wektora ch0 przyjmującego wartość maksymalną,
pod zmienną Ich0.
        [M,Ich0] = max(ch0);
%Zapisz indeks elementu wektora ch1 przyjmującego wartość maksymalną,
pod zmienną Ich1.
        [M,Ich1] = max(ch1);
%Zapisz indeks elementu wektora ch2 przyjmującego wartość maksymalną,
pod zmienną Ich2.
        [M,Ich2] = max(ch2);
%Zapisz indeks elementu wektora ch3 przyjmującego wartość maksymalną,
pod zmienną Ich3.
        [M,Ich3] = max(ch3);
%Wyznacz wartość opóźnienia pomiędzy sygnałem z mikrofonu referencyjnego
0 i mikrofonu 1.
%Wartość tau > 0 oznacza, że sygnał z mikrofonu 1 jest opóźniony w stosunku
do sygnału z mikrofonu referencyjnego 0.
        tau1=(Ich1-Ich0)*dt;
%Wyznacz wartość opóźnienia pomiędzy sygnałem z mikrofonu referencyjnego
0 i mikrofonu 2.
        tau2=(Ich2-Ich0)*dt;
%Wyznacz wartość opóźnienia pomiędzy sygnałem z mikrofonu referencyjnego
0 i mikrofonu 3.
        tau3=(Ich3-Ich0)*dt;
```

Wektory ch0-ch3 zawierają kolejno zarejestrowane wartości napięć sygnałów z mikrofonów 0-3. W ramach działania skryptu wyznaczono indeksy elementów wektorów, dla których przyjmowane są maksymalne wartości. Następnie obliczono opóźnienia τ_1, τ_2 oraz τ_3 zgodnie z metodą pierwszą.

Zarejestrowano dźwięk klaśnięcia oraz wyznaczono opóźnienie pomiędzy sygnałami z mikrofonów elektretowych. Zestawiono przebiegi sygnałów na wykresie, a następnie przesunięto sygnały o odpowiednie wartości opóźnień i zestawiono na drugim wykresie.



Rys. 20. Zarejestrowane sygnały z mikrofonów elektretowych

Analizując pierwszy wykres, na którym zestawiono nieprzesunięte sygnały, zauważono różnicę w fazie pomiędzy nimi, spowodowaną opóźnieniem z jakim fala akustyczna trafia na membrany poszczególnych mikrofonów.

Przesunięcie sygnałów z mikrofonów 1-3 o wartości zgodne z wyznaczonymi opóźnieniami τ_1, τ_2 oraz τ_3 powoduje, że pokrywają się one z sygnałem z mikrofonu referencyjnego 0, co potwierdza poprawność wyznaczonych wartości. Zauważono, że wszystkie sygnały pokrywają się aż do czasu 0.517 sekundy. Po tym czasie przebiegi sygnałów przestają być jednakowe. Najprawdopodobniej związane jest to odbijaniem się fali akustycznej od otaczających przeszkód i jej nakładaniem co powoduje, że membrana każdego mikrofonu wprawiana jest w drgania o różnej częstotliwości. Przed upływem 0.517 sekund przebiegi sygnałów są jednakowe, ponieważ odbita fala akustyczna nie zdążyła dotrzeć do mikrofonów.

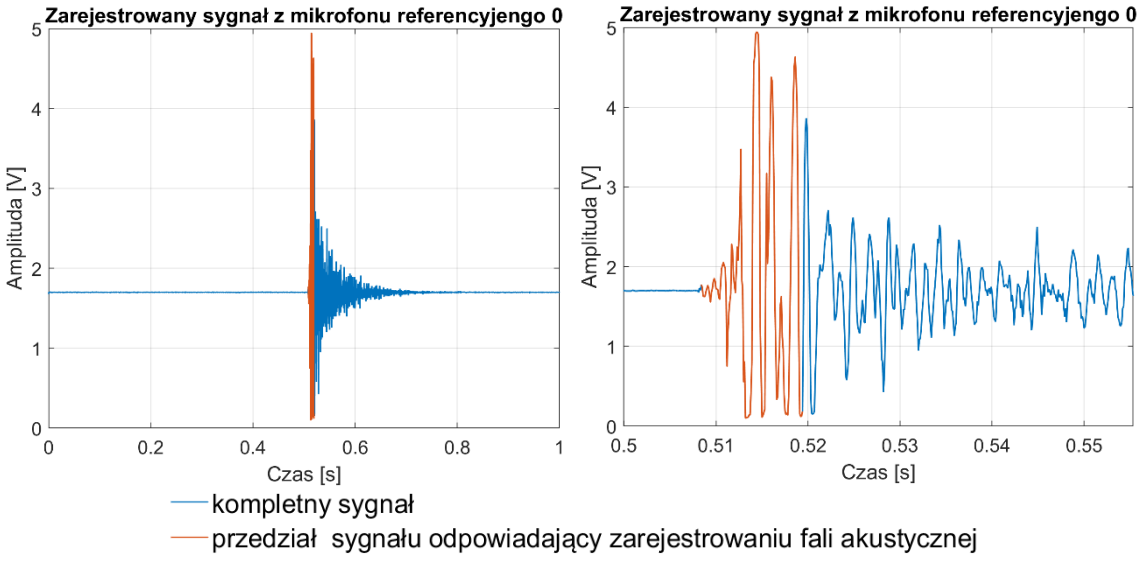
Drugą metodę wyznaczania opóźnień zaimplementowano w opisany poniżej sposób.

```
%Wektor ch0 przechowuje wartości napięcia sygnału z mikrofonu referencyjnego 0.
%Wektor ch1 przechowuje wartości napięcia sygnału z mikrofonu 1.
%Wektor ch2 przechowuje wartości napięcia sygnału z mikrofonu 2.
%Wektor ch3 przechowuje wartości napięcia sygnału z mikrofonu 3.
```

```
        dt=100*10^-6;  %Czas próbkowania sygnałów z mikrofonów.
%Zapisz indeks elementu wektora ch0 przyjmującego wartość maksymalną, pod zmienną
Ich0.
        [M,Ich0] = max(ch0);
%Zapisz indeks elementu wektora ch1 przyjmującego wartość maksymalną, pod zmienną
Ich1.
        [M,Ich1] = max(ch1);
%Zapisz indeks elementu wektora ch2 przyjmującego wartość maksymalną, pod zmienną
Ich2.
        [M,Ich2] = max(ch2);
%Zapisz indeks elementu wektora ch3 przyjmującego wartość maksymalną, pod zmienną
Ich3.
        [M,Ich3] = max(ch3);
%Posortuj rosnąco indeksy, dla których napięcia z poszczególnych mikrofonów przyjmują
%maksimum.
        sortUp=sort([Ich3 Ich2 Ich1 Ich0]);
%Określ przedział zawierający początkowy przebieg zarejestrowanej fali akustycznej.
%Zapobiegnie to niepotrzebnemu korelowaniu jej odbitej i zniekształconej części oraz
%szumów. Przedział oparty jest na indeksie sygnału, dla którego najwcześniej
%zarejestrowano falę akustyczną.
        range=sortUp(1)-60:sortUp(1)+50;
%Wyznacz wartości funkcji korelacji wzajemnej sygnałów z mikrofonu referencyjnego
%0 i mikrofonu 1 i zapisz je w wektorze c1.
        [c1,lags1]=xcorr(ch0(range),ch1(range));
%Zapisz indeks, dla którego funkcja korelacji wzajemnej sygnałów przyjmuje maksimum.
        [M,I1] = max(c1);
%Wyznacz wartość opóźnienia pomiędzy sygnałami z mikrofonu 0 i mikrofonu 1
        tau1=(length(ch0(range))-I1)*dt;
%Wykonaj analogiczne operacje dla sygnałów z mikrofonu 0 i mikrofonu 2 wyznaczając
%wartość opóźnienia tau2.
        [c2,lags2]=xcorr(ch0(range),ch2(range));
        [M,I2] = max(c2);
        tau2=(length(ch0(range))-I2)*dt;
%Wykonaj analogiczne operacje dla sygnałów z mikrofonu 0 i mikrofonu 3 wyznaczając
%wartość opóźnienia tau3.
        [c3,lags3]=xcorr(ch0(range),ch3(range));
        [M,I3] = max(c3);
        tau3=(length(ch0(range))-I3)*dt;
```

W ramach działania algorytmu odnaleziono przedział, w którym mikrofony zarejestrowały fale akustyczną pochodzącą od klaśnięcia. Dla tego zakresu wyznaczono korelację wzajemną sygnałów z poszczególnych mikrofonów. Na podstawie indeksów, dla których odpowiednie funkcje korelacji przyjęły maksimum, obliczono opóźnienia τ_1, τ_2 oraz τ_3 zgodnie z metodą drugą.

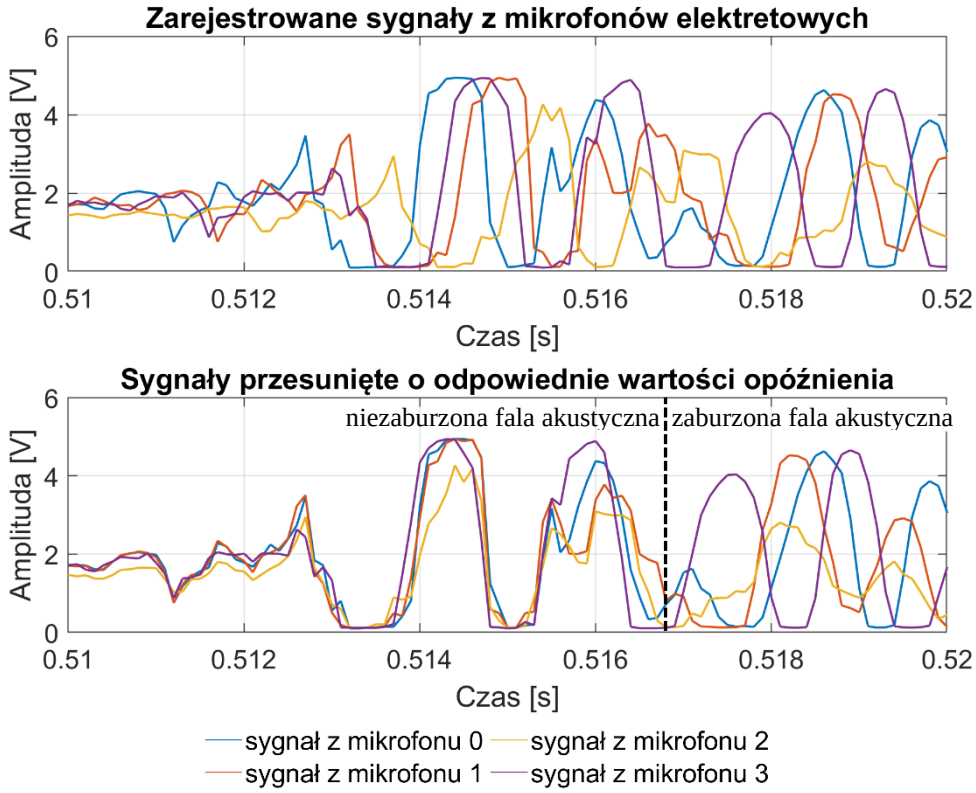
Zarejestrowano dźwięk klaśnięcia oraz wyznaczono przedział, w którym fala akustyczna dotarła na membranę mikrofonów. Przedstawiono na wykresie sygnał z mikrofonu referencyjnego 0 oraz zaznaczono przedział, dla którego obliczona zostanie korelacja wzajemna sygnałów.



Rys. 21. Zarejestrowany sygnał z mikrofonu 0 z zaznaczonym przedziałem

Zawężony przedział poprawnie odwzorowuje chwilę, w której fala akustyczna pochodzącą od klaśnięcia została zarejestrowana przez mikrofon elektretowy. Przedział zawiera początkowy przebieg fali akustycznej, która w miarę upływu czasu została zniekształcona poprzez odbijanie się od obiektów i nakładanie się.

Aby określić poprawność obliczonego opóźnienia, zestawiono przebiegi zarejestrowanych sygnałów na wykresie, a następnie przesunięto je o odpowiednie wartości wyznaczonych opóźnień i przedstawiono na drugim wykresie.



Rys. 22. Zarejestrowane sygnały z mikrofonów elektretowych

Przesunięcie sygnałów z mikrofonów 1-3 o wartości zgodne z wyznaczonymi opóźnieniami τ_1, τ_2 oraz τ_3 powoduje, że pokrywają się one z sygnałem z mikrofonu referencyjnego 0, co potwierdza poprawność wyznaczonych wartości. Ponownie zauważono, że wszystkie sygnały pokrywają się aż do czasu 0.517 sekundy, po którym następuje odbijanie i nakładanie się fali.

Dla znanych wartości opóźnień oraz położenia mikrofonów wyznaczano położenie źródła dźwięku, korzystając z algorytmu omówionego w podrozdziale 1.9. Poniżej przedstawiono jego implementację w środowisku MATLAB®.

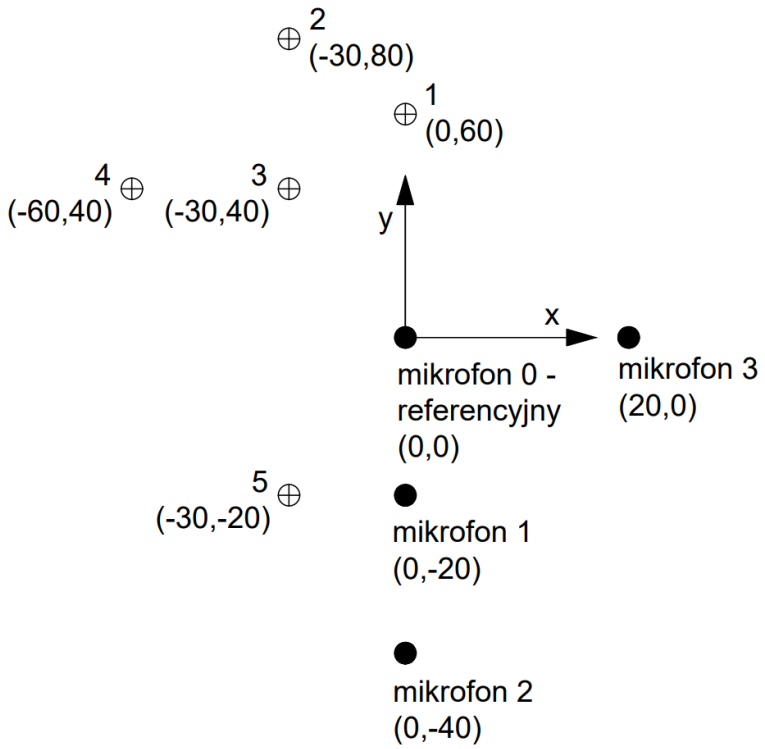
```
x0=0;  y0=0;  %Współrzędne mikrofonu referencyjnego 0.
x1=0;  y1=-20; %Współrzędne mikrofonu 1.
x2=0;  y2=-40; %Współrzędne mikrofonu 2.
x3=20; y3=0;  %Współrzędne mikrofonu 3.
vs=34300;     %Prędkość dźwięku równa 34300 cm/s.
d1=tau1*vs;   %Wyznaczono wartości delta_1, zgodnie z równaniem (5).
d2=tau2*vs;   %Wyznaczono wartości delta_2, zgodnie z równaniem (5).
d3=tau3*vs;   %Wyznaczono wartości delta_3, zgodnie z równaniem (5).
%Wyznaczono wartości parametrów A-L z równań (12), (13), (14).
A=2*x1-2*x0;  B=2*y1-2*y0;  C=2*d1;  D=x1^2+y1^2-x0^2-y0^2-d1^2;
E=2*x2-2*x0;  F=2*y2-2*y0;  G=2*d2;  H=x2^2+y2^2-x0^2-y0^2-d2^2;
```

```
I=2*x3-2*x0;  J=2*y3-2*y0;  K=2*d3;  L=x3^2+y3^2-x0^2-y0^2-d3^2;
%Wyznaczono współrzędne x i y źródła dźwięku oraz jego odległość od początku układu
%współrzędnych - r0. Implementacja równań (15), (16), (17).
x=(L-J*D/B-B/(G*B-F*C)*(K-J*C/B)*(H-F*D/B))/(I-J*A/B+B/(G*B-F*C)*(K-J*C/B)*(F*A/B-
E))
r0=B/(G*B-F*C)*(H-F*D/B+x*(F*A/B-E))
y=1/B*(D-C*r0-A*x)
```

W ramach działania algorytmu, na podstawie znanych współrzędnych położenia mikrofonów oraz obliczonego opóźnienia pomiędzy zarejestrowanymi sygnałami, wyznaczono poszukiwane współrzędne x i y źródła dźwięku.

2.6. Przeprowadzone badania i uzyskane wyniki

Dokładność wyznaczenia położenia źródła dźwięku przez zaprojektowane układy oparte na Arduino UNO oraz Raspberry Pi 2, zbadano dla pięciu prób, w których zmieniano położenie źródła fali akustycznej. Rozmieszczenie mikrofonów oraz położenie źródła dźwięku dla każdej próby przedstawiono na schemacie poniżej.



Rys. 23. Schemat pokazujący położenie mikrofonów oraz miejsca emitowania fali akustycznej

Położenie źródła fali akustycznej dla kolejnych prób oznaczone zostało liczbami od jednego do pięciu. Początek układu współrzędnych został przyjęty w punkcie umieszczenia mikrofonu referencyjnego 0.

W ramach testów układu opartego na Arduino UNO otrzymano następujące wartości opóźnień wyznaczonych na podstawie metod opisanych w poprzednim rozdziale.

Próba	Wyznaczone wartości opóźnień						Teoretyczne wartości opóźnień		
	Wykorzystując metodę 1			Wykorzystując metodę 2					
	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]
1	600	1200	0	500	1200	100	583	1166	95
2	600	1100	200	500	1000	100	553	1115	259
3	500	1000	300	400	1000	400	498	1033	409
4	400	800	400	300	800	400	371	813	505
5	-100	0	500	-100	0	400	-177	0	519

Tab. 1. Zestawienie wartości opóźnień pomiędzy sygnałami zarejestrowanymi przy użyciu Arduino UNO

Dla wszystkich otrzymanych wyników obliczono błąd bezwzględny $\Delta\tau = |\tau_t - \tau_w|$, zdefiniowany jako różnica pomiędzy teoretyczną oraz wyznaczoną wartością opóźnienia, gdzie τ_t jest wartością teoretyczną, a τ_w jest wartością wyznaczoną. Dla zebranych wyników otrzymano następujące wartości błędu bezwzględnego.

Próba	Wartość błędu bezwzględnego wyznaczonych opóźnień							
	Dla metody 1				Dla metody 2			
	$\Delta\tau_1$ [µs]	$\Delta\tau_2$ [µs]	$\Delta\tau_3$ [µs]	$\Delta\tau_{śr}$ [µs]	$\Delta\tau_1$ [µs]	$\Delta\tau_2$ [µs]	$\Delta\tau_3$ [µs]	$\Delta\tau_{śr}$ [µs]
1	17	34	95	44	83	34	5	65
2	47	15	59		53	115	159	
3	2	33	109		98	33	9	
4	29	13	105		71	13	105	
5	77	0	19		77	0	119	

Tab. 2. Zestawienie wartości błędu bezwzględnego wyznaczonych opóźnień

Na podstawie wartości opóźnień z tabeli 1. wyznaczono współrzędne położenia źródła dźwięku i zestawiono je z wartościami teoretycznymi.

Próba	Wyznaczone współrzędne położenia źródła dźwięku				Teoretyczne współrzędne położenia źródła dźwieku	
	Wykorzystując metodę 1		Wykorzystując metodę 2			
	x [cm]	y [cm]	x [cm]	y [cm]	x [cm]	y [cm]
1	-	-	8,0	5,9	0	60
2	-	-	8,0	5,9	-30	80
3	4,8	1,7	2,7	-2,7	-30	40
4	2,7	-2,7	1,8	-4,7	-60	40
5	-13,1	-12,9	-3,1	-11,8	-30	-20

Tab. 3. Zestawienie wyznaczonych współrzędnych położenia źródła dźwięku

W ramach testów układu opartego na Raspberry Pi 2 otrzymano następujące wartości opóźnień wyznaczonych na podstawie metod opisanych w poprzednim rozdziale.

Próba	Wyznaczone wartości opóźnień						Teoretyczne wartości opóźnień		
	Wykorzystując metodę 1			Wykorzystując metodę 2					
	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]	τ_1 [µs]	τ_2 [µs]	τ_3 [µs]
1	560	1120	80	560	1040	80	583	1166	95
2	480	1040	240	480	1040	240	553	1115	259
3	560	1120	480	480	960	400	498	1033	409
4	1520	800	480	320	800	400	371	813	505
5	-160	0	560	-160	0	480	-177	0	519

Tab. 4. Zestawienie wartości opóźnień pomiędzy sygnałami zarejestrowanymi przy użyciu Raspberry Pi 2

Dla zebranych wyników otrzymano następujące wartości błędu bezwzględnego.

Próba	Wartość błędu bezwzględnego wyznaczonych opóźnień							
	Dla metody 1				Dla metody 2			
	$\Delta\tau_1$ [µs]	$\Delta\tau_2$ [µs]	$\Delta\tau_3$ [µs]	$\Delta\tau_{śr}$ [µs]	$\Delta\tau_1$ [µs]	$\Delta\tau_2$ [µs]	$\Delta\tau_3$ [µs]	$\Delta\tau_{śr}$ [µs]
1	23	46	15	114	23	126	15	44
2	73	75	19		73	75	19	
3	62	87	71		18	73	9	
4	1149	13	25		51	13	105	
5	17	0	41		17	0	39	

Tab. 5. Zestawienie wartości błędu bezwzględnego wyznaczonych opóźnień

Na podstawie wartości opóźnień z tabeli 4. wyznaczono współrzędne położenia źródła dźwięku i zestawiono je z wartościami teoretycznymi.

Próba	Wyznaczone współrzędne położenia źródła dźwięku				Teoretyczne współrzędne	
	Wykorzystując metodę 1		Wykorzystując metodę 2			
	x [cm]	y [cm]	x [cm]	y [cm]	x [cm]	y [cm]
1	7,3	16,9	7,3	16,9	0	60
2	5,8	1,8	5,8	1,8	-30	80
3	1,6	1,1	-50,7	64,0	-30	40
4	20,2	4,2	2,0	-4,4	-60	40
5	23,5	-2,8	-12,6	-14,5	-30	-20

Tab. 6. Zestawienie wyznaczonych współrzędnych położenia źródła dźwięku

2.7. Analiza wyników

Różnica pomiędzy dokładnością wyznaczenia opóźnień przy wykorzystaniu Arduino UNO oraz Raspberry Pi jest niewielka. Porównując dane z tabeli 2. oraz 5. zauważono, że wartości średnie błędów bezwzględnych dla obu urządzeń są zbliżone oraz nie przekraczają wartości okresu próbkowania T, wynoszącego 100 μs dla Arduino UNO oraz 80 μs dla Raspberry Pi 2. Wartość błędu bezwzględnego sporadycznie przekracza okres próbkowania (20% wszystkich wyznaczonych opóźnień dla Arduino UNO i 10% dla Raspberry Pi 2), jednak nigdy nie przekracza jego dwukrotności. Oznacza to, że opóźnienia między sygnałami wyznaczano z dokładnością wynoszącą dwukrotność okresu próbkowania (± 2·T), co świadczy o poprawnym rejestrowaniu sygnałów przez obie platformy.

Różnica pomiędzy dokładnością z jaką wyznaczono wartości opóźnień przy wykorzystaniu metody pierwszej oraz metody drugiej jest nieznaczna. Analizując dane z tabeli 2. i 5. zauważono, że oba algorytmy posiadają zbliżoną skuteczność, co potwierdzają zbliżone wartości średnich błędów bezwzględnych. Na przestrzeni całego badania tylko jedna wartość wyznaczonego opóźnienia może zostać zakwalifikowana jako niewłaściwa i obarczona błędem, jest nią wartość opóźnienia τ_1 wyznaczona metodą pierwszą dla próby 4. przeprowadzonej na układzie Raspberry Pi 2, która przekroczyła dziesięciokrotność okresu próbkowania.

Porównując wyznaczone współrzędne położenia źródła dźwięku z wartościami teoretycznymi stwierdzono, że wszystkie wyznaczone wartości są błędne, bez względu na metodę wykorzystaną do obliczenia opóźnień oraz platformę, na której rejestrowane były sygnały.

Powodem tego jest specyfika przyjętego modelu matematycznego, dla którego wartości rozwiązań układu równań (1), (2), (3), (4) są wrażliwe na dokładność, z jaką wyznaczone zostają odległości Δ_1, Δ_2 oraz Δ_3 (5). Aby wyznaczyć położenie źródła dźwięku z dokładnością ±1 cm, wartości Δ_n muszą zostać wyznaczone z dokładnością ±1 mm. Zaprojektowane układy wyznaczają odległości Δ_n z dokładnością ±3.43 cm dla platformy Arduino UNO oraz ±2.74 cm dla Raspberry Pi. Dokładność ta zależy od częstotliwości próbkowania sygnałów oraz prędkości dźwięku wynoszącej w przybliżeniu 34300 cm/s. Aby osiągnąć wymaganą dokładność, sygnały z mikrofonów należałoby próbkować z częstotliwością 343 kHz .

Podsumowanie i wnioski

W ramach niniejszej pracy inżynierskiej udało się zaprojektować dwa urządzenia rejestrujące sygnały audio, oparte na platformach Arduino UNO oraz Raspberry Pi. Stworzono również układ wzmacniacza przeznaczonego dla sygnałów z mikrofonów elektretowych. Wzmocnione sygnały zarejestrowano poprzez zaprogramowanie komunikacji SPI pomiędzy platformami oraz przetwornikiem analogowo-cyfrowym, odbywającej się w ramach cyklicznego przerwania. Dla Arduino UNO stworzono dodatkowo algorytm szeregowej transmisji danych, opartej na komunikacji UART. Zarejestrowane sygnały z czteroelementowej macierzy mikrofonowej zostały wykorzystane do wyznaczenia źródła położenia dźwięku. W ramach tego stworzono model matematyczny rozchodzenia się fali akustycznej w powietrzu pozwalający na wyznaczenie poszukiwanych współrzędnych. Konieczne do tego było ustalenie opóźnienia pomiędzy sygnałami z poszczególnych mikrofonów. W tym celu stworzono dwie metody pozwalające na jego wyznaczenie, z których jedna została oparta o funkcję korelacji wzajemnej sygnałów. Dla obu urządzeń przeprowadzone zostało badanie złożone z pięciu prób, w ramach których zmieniano miejsce emitowania fali akustycznej. Podczas każdej próby zarejestrowano sygnały z mikrofonów elektretowych i wyznaczono opóźnienie pomiędzy nimi oraz położenie źródła dźwięku. Na końcu dokonano analizy otrzymanych wyników, porównując wyznaczone wartości z wartościami teoretycznymi oraz określono wielkość błędu bezwzględnego wyznaczonych opóźnień.

Pomimo tego, że opóźnienia pomiędzy sygnałami wyznaczone zostały z satysfakcjonującą dokładnością, to żadne z wyznaczonych współrzędnych nie są poprawne. Aby osiągnąć poprawne rezultaty częstotliwość rejestrowania sygnałów musiałaby zostać zwiększona niemal 35-krotnie. Stąd konieczne byłoby zastosowanie znacząco wydajniejszego układu mikrokontrolera sterującego procesem rejestrowania sygnałów. Należałoby również zastosować przetwornik analogowo-cyfrowy o przetwarzaniu bezpośrednim (flash ADC), mogący być taktowany sygnałem o znacząco większej częstotliwości, w porównaniu do zastosowanego przetwornika A/C MCP3008.

Bibliografia

[1] Richard G. Lyons; z jęz. ang. przeł. Jan Zarzycki *Wprowadzenie do cyfrowego przetwarzania sygnałów*, Wydawnictwa Komunikacji i Łączności, Warszawa 2010, str. 19

[2] Alan V. Oppenheim, Ronald W. Schafer; z jęz. ang. przeł. zespół pod kierunkiem Andrzeja Lizonia *Cyfrowe przetwarzanie sygnałów*, Wydawnictwa Komunikacji i Łączności, Warszawa 1979, str. 17

[3] https://raspberrytips.com/raspberry-pi-history/

[4] https://forbot.pl/blog/premiera-raspberry-pi-pico-za-4-z-nowym-ukladem-rp2040-id46800

[5] https://developer.arm.com/Processors/Cortex-A7

[6] https://web.eece.maine.edu/~vweaver/group/green_machines.html

[7] https://www.raspberrypi.com/documentation/computers/os.html

[8] https://www.debian.org/intro/about

[9] https://initialcommit.com/blog/what-programming-language-does-raspberry-pi-use

[10] https://edux.pjwstk.edu.pl/mat/198/lec/main110.html

[11] http://ics.p.lodz.pl/~dpuchala/Architektura%20RISC%20-%20wyklad.pdf

[12] https://www.arm.com/blogs/blueprint/200bn-arm-chips

[13] https://solectroshop.com/pl/content/60-5-pinowy-gpio-i-jego-programowanie

[14] https://arduinohistory.github.io/

[15] https://www.arduino.cc/en/Main/Products

[16] https://arduino.github.io/arduino-cli/0.20/sketch-build-process/

[17] https://forbot.pl/blog/leksykon/arduino-uno

[18] https://eduinf.waw.pl/inf/prg/009_kurs_avr_old/0016.php

[19] https://livesound.pl/tutoriale/4477-mikrofony-iii-mikrofony-pojemnosciowe-zasada-dzialania-wady-i-zalety

[20] https://www.electronics-tutorials.ws/pl/dioda/podstawy-polprzewodnikow.html

[21] https://www.electronics-tutorials.ws/pl/tranzystor/tranzystor-bipolarny.html

[22] http://www.zsp1slupsk.pl/elektronika/pliki/Przetworniki_AC_CA.pdf

[23] https://pdf1.alldatasheet.com/datasheet-pdf/view/304549/MICROCHIP/MCP3008.html

[24] http://nanophysics.pl/teaching/systemy_pomiarowe/systemy_pomiarowe_04.pdf

[25] http://extronic.pl/content/60-kurs-xmega-interfejs-spi
[26] https://pdf1.alldatasheet.com/datasheet-pdf/view/304549/MICROCHIP/MCP3008.html
[27] https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html
[28] Tomasz P. Zieliński *Cyfrowe przetwarzanie sygnałów: od teorii do zastosowań*, Wydawnictwa Komunikacji i Łączności, Warszawa 2005, str. 14-16
[29] https://www.arduino.cc/en/reference/SPI
[30] https://www.tutorialspoint.com/timer-registers-in-arduino
[31] https://maxembedded.com/2011/07/avr-timers-ctc-mode/
[32] https://www.robotshop.com/community/forum/t/arduino-101-timers-and-interrupts/13072
[33] http://www.airspayce.com/mikem/bcm2835/index.html
[34] https://man7.org/linux/man-pages/man3/ualarm.3.html
[35] http://linuxonflash.blogspot.com/2015/02/a-look-at-raspberry-pi-2-performance.html
[36] https://home.agh.edu.pl/~kozlak/PS2010/signals.html

Spis ilustracji