

Zespół Szkół Mechaniczno-Elektrycznych
w Żywcu
34-300 Żywiec, ul. KEN 3

Rok szkolny
2022/2023

Olimpiada Innowacji Technicznych

Temat:

FuturFlow - prototyp lewitatora akustycznego

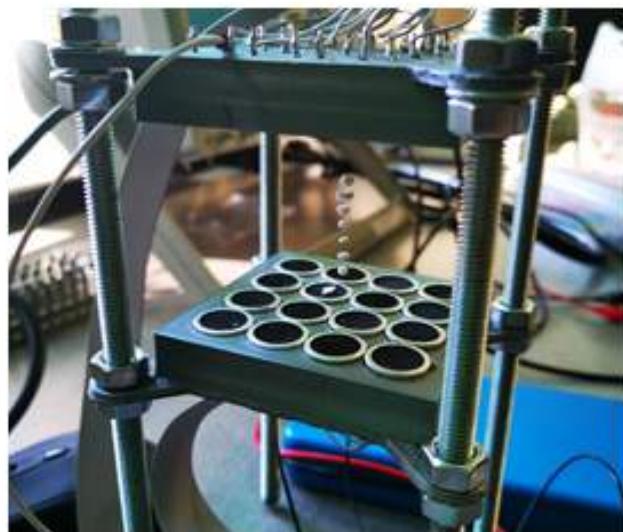
Konsultanci:
mgr. Elżbieta Burlaga
mgr. inż. Leszek Szpila

Autorzy:
Krzysztof Mrózek
Krzysztof Nowak

Lewitator Akustyczny

Krzysztof Nowak i Krzysztof Mrózek

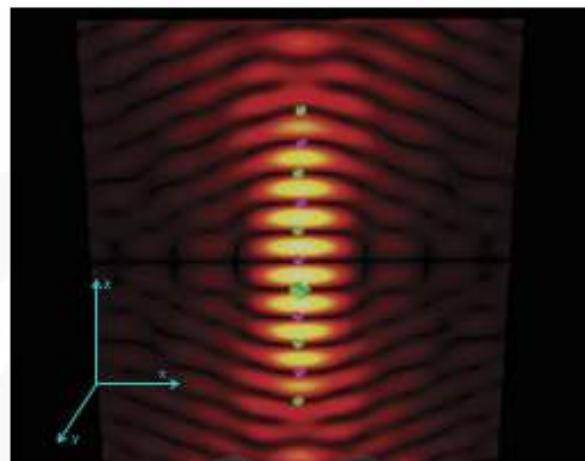
Zespół Szkół Mechaniczno-Elektrycznych w Żywcu



FPGA

Ultrasoniczne
przetworniki

Symulacja Fali
Dźwiękowej



Sterowanie
przesunięciami
fazowymi



Cel pracy - wstępne założenia

Celem naszej pracy jest **usprawnienie bezinwazyjnego transportu obiektów przy pomocy lewitacji ultradźwiękowej w ramach procesu produkcyjnego**.

Nasze nietypowe rozwiązywanie FuturFlow służy do **transportu przedmiotów**, szczególnie tych bardzo **delikatnych**, w przypadku których zastosowanie mechanicznych sposobów transportu mogłoby je uszkodzić. Jest to połączenie wcielonego w życie **innowacyjnego pomysłu** wraz z nauką towarzyszących zjawisk fizycznych, skupiające się wokół praktycznych rozwiązań w przemyśle.

Nasze główne założenia to:

- zaprojektowanie i wykonanie uniwersalnego systemu lewitacji,
- stworzenie modelu ze śledzeniem obiektu lewitującego,
- napisanie programu symulacyjnego do transportu w 3D,
- eksperymentalne dobranie optymalnych warunków do lewitacji,
- zweryfikowanie doświadczalnie uzyskanych danych, następnie określenie ograniczeń w dynamicznym modelu lewitacji ultradźwiękowej.

Naszym projektem przybliżyliśmy świat przyszłości, wyobraźni i rzeczywistości każdemu z nas.

Życzymy przyjemnej i obfitującej w nowe przemyślenia lektury naszej pracy,
Krzysztof Nowak i Krzysztof Mrózek

Spis treści

I. Podstawy teoretyczne	6
I.1. Historia lewitacji	6
I.2. Motywacja pracy	7
I.3. Rodzaje lewitacji	7
I.3.1. Lewitacja magnetyczna	8
I.3.2. Lewitacja aerodynamiczna	9
I.3.3. Lewitacja hydrodynamiczna	10
I.4. Teoria lewitacji akustycznej	11
I.4.1. Interferencja i wygaszenie fali akustycznej	11
I.4.2. Fala stojąca	12
I.4.3. Wymagania lewitacji akustycznej	14
I.4.4. Obliczanie opóźnień fazowych	14
I.4.6. Pole ciśnienia akustycznego	15
I.4.7. Piezoelektryki, czyli jak powstaje fala dźwiękowa	16
I.5. Zalety i wady transportu za pomocą fali dźwiękowej	17
I.6. Lewitacja akustyczna i jej wykorzystanie w praktyce	18
I.7. Wykorzystane programy	20
I.8. Opis elementów	22
II. Projekt techniczny	29
II.1. Pierwszy prototyp FuturFlow	30
II.1.1. Złożenie prototypu	31
II.1.2. Przygotowanie stanowiska	31
II.1.3. Elementy mechaniczne pierwszego prototypu	31
II.1.4. Przebieg pracy	32
II.1.4.1 Sprawdzenie polaryzacji głośników ultradźwiękowych	32
II.1.4.2 Montowanie elektroniki	33
II.1.4.3 Diagnostyka i testowanie głośników	34
II.1.5 Pierwsza lewitacja	35
II.1.5.1 Spalenie FPGA	36
II.1.6. Tworzenie programu do lewitacji	36
II.1.6.1. Modyfikacja układu	36
II.1.6.2 Programowe uzyskanie częstotliwości	37
II.1.6.3. Programowa realizacja przesunięć fazowych	39
II.1.6.4. Problem rozproszenia fali akustycznej	44
II.1.7. Praca z symulatorem	45
II.1.8. Testowanie działania	47
II.1.8.1. Lewitacja kilku cząsteczek na raz	48

II.1.9. Obrazowanie Schlierena	49
II.2 Finalna wersja prototypu FuturFlow	53
II.2.1. Schemat elektryczny	53
II.2.2. Projekt płytki PCB	56
II.2.3. Wykonywanie płytki PCB	59
II.2.4. Projekt programu do lewitacji	65
II.2.5. Program Quartus	70
II.2.6. Aplikacja lewitacji w Pythonie	74
II.2.6.1. Interfejs	74
III.2.6.2 Sekcja rysowania	79
II.2.7 Obudowa lewitatora	83
II.2.7.1. Projekt obudowy	83
II.2.7.2. Drukowanie obudowy	89
II.2.8. Złożenie finalnego prototypu	94
II.2.9. Uruchomienie urządzenia	99
II.2.10. Udoskonalony wzór na obliczanie przesunięć fazowych	100
II.2.11. Testowanie działania	101
II.2.12. Instrukcja obsługi prototypu	105
III. Wnioski powykonawcze	106
Link do prezentacji online : https://youtu.be/XUKNI5owW9Q	106
IV. Kosztorys	107
V. Bibliografia	108

I. Podstawy teoretyczne

I.1. Historia lewitacji

Na przestrzeni ostatnich dziesięcioleci lewitacja zyskała dużą popularność wśród naukowców, co zaowocowało różnymi metodami lewitacji cząsteczek m.in lewitacja magnetyczna, optyczna, aero- i hydrodynamiczna, elektrostatyczna, lewitacja oparta na sile wyporu oraz lewitacja akustyczna.

Pierwszy eksperyment przeprowadzony z użyciem tego zjawiska został przeprowadzony w 1933 roku przez K. Bücksa i H. Müllera. Wtedy to też unosili oni krople alkoholu między kryształem kwarcu a reflektorem. Kolejne postępy zostały poczynione przez **Taylora Wanga**, który przetestował możliwości lewitacyjne w mikrogravitacji podczas misji promu kosmicznego **Challenger STS-51-B**. Przez ostatnie 70 lat aż do roku 2017 najpopularniejszym z kolei urządzeniem do lewitacji akustycznej był tzw. Róg Langevina składający się z piezoelektrycznego materiału, nadajnika oraz reflektora. Metoda ta wymagała jednak wielkiej precyzji, gdyż odległość między nadajnikiem a reflektorem musiała być dokładną wielokrotnością długości fali, która w zależności od prędkości dźwięku zmienia się wraz z czynnikami środowiskowymi takimi jak wilgotność czy temperatura.

Problem ten częściowo został wyeliminowany poprzez wprowadzenie **PAL** (ang. Phased Array Levitator). PAL jest układem z określoną liczbą głośników, które bazując na interferencji fali dźwiękowej mogą generować i sterować polem akustycznym.

Pierwszą osobą która przedstawiła użycie PAL był **Yoichi Ochiai**. W swoim projekcie użył **1140 głośników ultradźwiękowych** (po 285 w każdym z 4 układów). Rok później w 2015 roku profesor Asier Marzo po raz pierwszy uzyskał lewitację przy pomocy jedynie jednostronnego układu z głośnikami, co spowodowało przyrost liczby nowych zastosowań lewitacji akustycznej w wielu dziedzinach nauki.

I.2. Motywacja pracy

W 2017 roku wspomniany wcześniej **profesor Asier Marzo** opublikował wraz ze swoimi współpracownikami na uniwersytecie bristolim projekt o nazwie “**TinyLev**”. Skupiające się na lewitacji akustycznej urządzenie było w stanie **lewitować obiekty szerokiego pokroju**, zarówno metale jak i całkiem zwyczajne niemagnetyczne przedmioty. Zaliczały się do nich np. kropelki płynów, a **nawet organizmy żywe** o bardzo małej masie.

Rok później opublikował on projekt “**Ultraino**”. Był już to bardziej rozbudowany system lewitacji, która mogła być sterowana w osi 3D. Również ilość sterowanych obiektów wzrosła, można było **niezależnie sterować kilkoma a nawet kilkunastoma częsteczkami** na raz. To zachęciło szereg naukowców z wielu dziedzin do opracowania dalszych zastosowań akustycznej lewitacji.

Zainspirowani pracą Profesora Asiera Marzo rozpoczęliśmy przygotowanie merytoryczne do **projektu urządzenia wykorzystującego zjawisko akustycznej lewitacji**. Temat okazał się bardziej rozbudowany niż sądziliśmy. Chcieliśmy jednak, żeby nasz projekt był **intuicyjny i przejrzysty**. Dokumenty badawcze z którymi się spotkaliśmy, zawierały wiele skomplikowanych **równań falowych**, a występujące tam operacje matematyczne daleko wybiegały poza obszar naszej wiedzy. Na przestrzeni kolejnych miesięcy zgłębiliśmy zarówno teorię fizyczną tego zjawiska, jak i zajeliśmy się jego rzeczywistą implementacją.

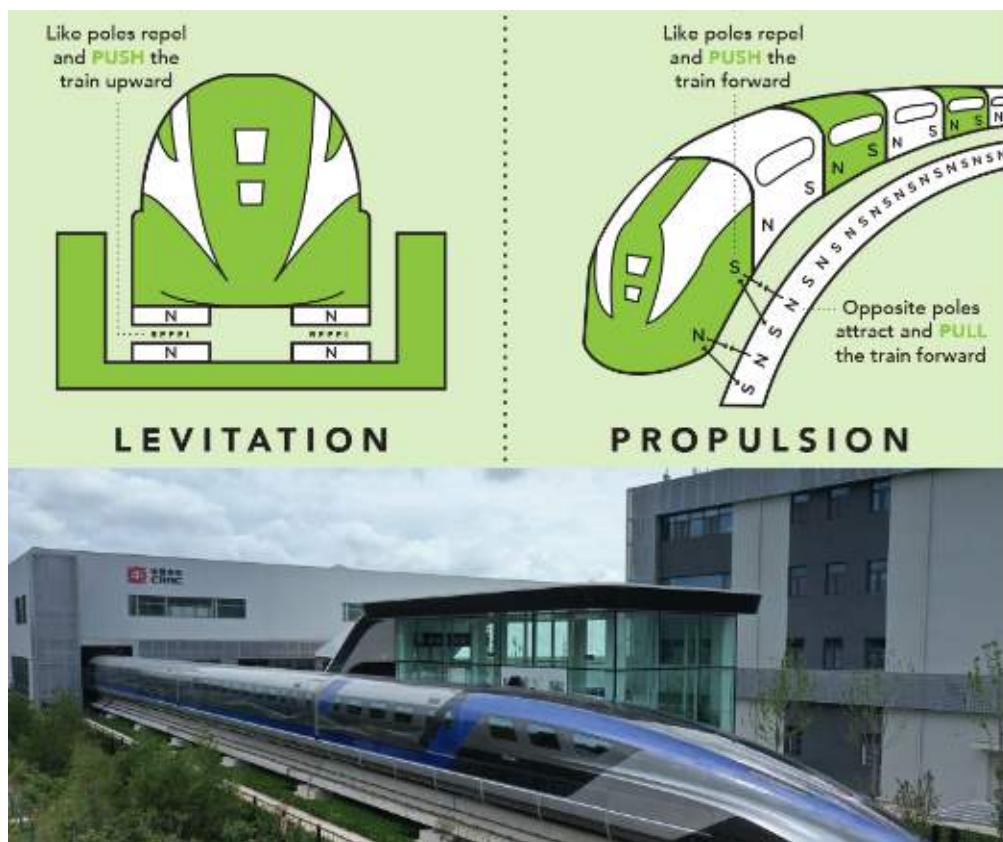
I.3. Rodzaje lewitacji

Świat który nas otacza, jest pełny niesamowitych zjawisk fizycznych. Opisywana przez nas **lewitacja akustyczna** to tylko jeden ze sposobów unoszenia obiektów w powietrzu bez fizycznego kontaktu z podłożem. Oprócz niej występują również bardziej powszechnie sposoby lewitacji takie jak np. lewitacja magnetyczna lub też mniej znana - lewitacja optyczna. Każda z tych metod charakteryzuje się unikalnymi cechami oraz inną zasadą działania, lecz w każdej chodzi o **wytworzenie siły będącej w stanie zrównoważyć siłę grawitacji działającą na obiekt**.

I.3.1. Lewitacja magnetyczna

Lewitacja magnetyczna zachodzi pod wpływem oddziaływania sił elektromagnetycznych i pola magnetycznego, które równoważą siłę wynikającą z grawitacji lub innych czynników. Znajduje ona zastosowanie w szerokich dziedzinach przemysłu np. w łożyskach magnetycznych czy też systemach szybkiego przemieszczania.

Świetnym przykładem jest zastosowanie lewitacji magnetycznej w pociągach Maglev, gdzie użyte są 2 zestawy elektromagnesów, jeden do odpychania i wypychania z toru, a drugi do przemieszczania wzniesionej maszyny do przodu. Pociąg porusza się po prowadnicach elektromagnesów, które kontrolują stabilność i prędkość pociągu. Dzięki temu pociąg może osiągać prędkości do nawet **600 km/h czyli ponad dwa razy więcej niż obecny rekord prędkości najszybszego pociągu w Polsce**.



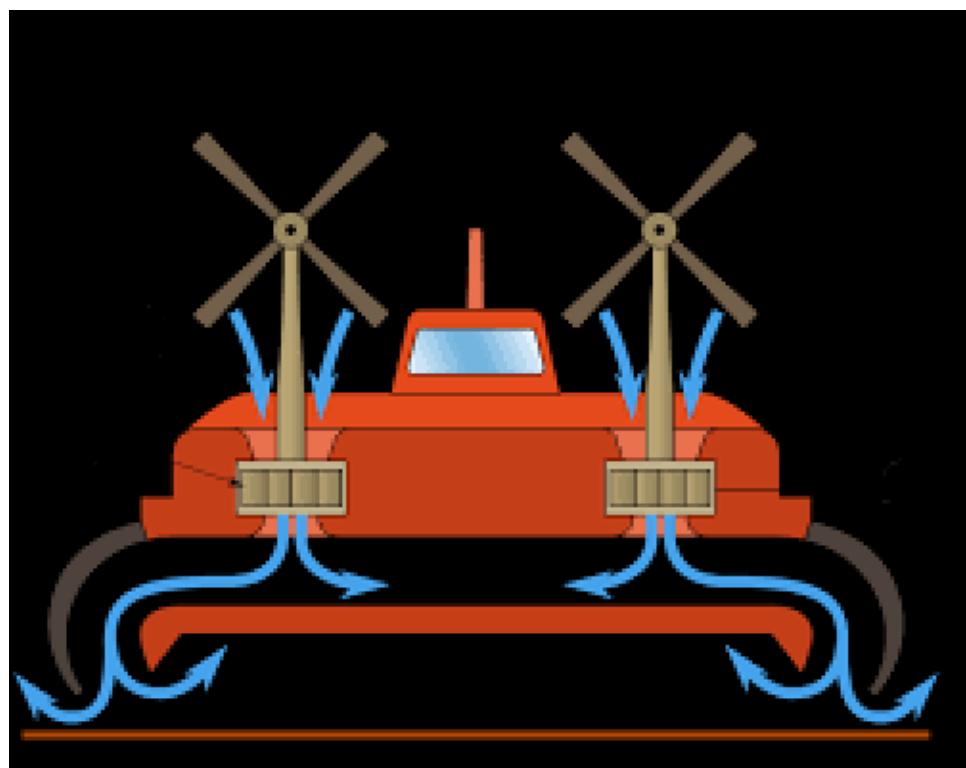
Rys. 1 MagLev - pociąg przyszłości
Źródło: energy.gov/articles/how-maglev-works

I.3.2. Lewitacja aerodynamiczna

Lewitacja aerodynamiczna wytwarzana jest przez strumień gazu skierowany na ciało tak, aby równoważyć siłę grawitacji. Obiekt może utrzymywać się stabilnie w pionowym strumieniu powietrza stabilizowanym przez siłę. Może ona być uzyskana przez działanie sprężonego powietrza na lewitujący przedmiot.

Dobrym przykładem jest Air Hockey gdzie powietrze wydmuchiwanie przez zestaw małych dysz znajdujących się na stole unosi lekki przedmiot na małej wysokości nad nim, który może być przesuwany z minimalnym oporem. Podobne, lecz dużo większe urządzenia, umożliwiają unoszenie ludzi. Są to tzw. **tuneli aerodynamiczne** wykorzystywane do np. symulacji spadania ze spadochronem.

Powietrze może być również wpychane przez obiekt w dół. Tę zasadę wykorzystuje się głównie w **poduszkowcu**, gdzie specjalny wentylator zasysa powietrze z otoczenia i wtłacza je pod spód pojazdu, patrz Rys. 2.



Rys. 2 Zasada działania poduszkowca

Źródło: [wikimedia.org/wiki/File:Hovercraft_-_scheme.svg](https://commons.wikimedia.org/w/index.php?title=File:Hovercraft_-_scheme.svg&oldid=100000000)

I.3.3. Lewitacja hydrodynamiczna

Lewitacja hydrodynamiczna jest bardzo podobna do lewitacji aerodynamicznej, z tą różnicą, że **czynnikiem wytwarzającym siłę unoszącą jest strumień cieczy**. Zjawisko to działa najlepiej gdy obiektem lewitującym jest przedmiot w kształcie zbliżonym do sfery. Warto również wspomnieć, że im silniejszy jest strumień wyrzucanej cieczy, tym cięższe obiekty mogą ulec lewitacji. Gdy strumień cieczy doryka takie ciało, wprawia je w ruch, gdzie znowu zaczyna działać siła dośrodkowa. Siły działające na sferę są w równowadze dlatego może ona się unosić. Całe zjawisko opisuje **zasada Bernoulliego**.



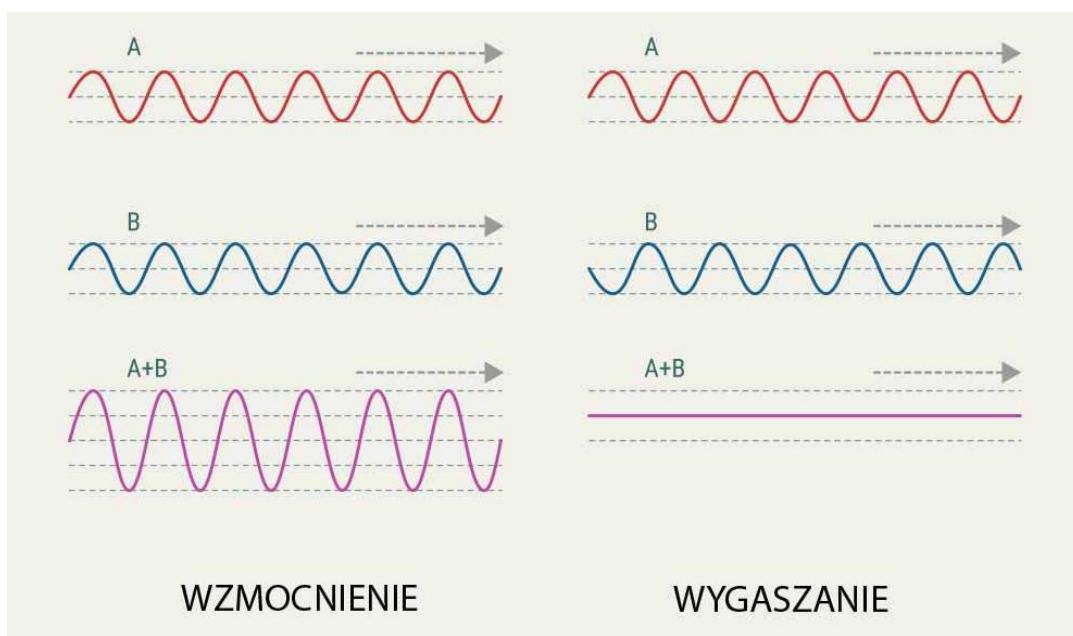
Rys. 3 Lewitacja hydrodynamiczna kulki styropianowej

Źródło: youtu.be/mNHp8iyyIjo

I.4. Teoria lewitacji akustycznej

I.4.1. Interferencja i wygaszenie fal akustycznej

Fizyczne zagadnienia związane z akustyczną lewitacją dotyczą zasadniczo **interakcji fal dźwiękowych z przedmiotami**. Aby lewitacja mogła mieć miejsce niezbędna jest odpowiednia interakcja fal dźwiękowych między sobą. Możemy wymienić dwie główne rodzaje takiej interakcji: **wzmocnienie fali oraz jej wygaszenie**.



Rys. 4 Interferencja i wygaszenie fal akustycznej
Źródło: medianauka.pl/fizyka/grafika/interferencja.jpg

Wzmocnienie interferencyjne jest **kluczowym czynnikiem umożliwiającym akustyczną lewitację**. Polega ono na zwiększeniu amplitudy fali akustycznej w miejscu, gdzie spotykają się dwie lub więcej fal. Fale interferują ze sobą w sposób konstruktywny, co prowadzi do zwiększenia siły nośnej fali wypadkowej.

1. Warunek **wzmocnienia interferencyjnego** opisuje wzór:

$$\cos 2\Pi \left(\frac{r_2 - r_1}{2\lambda} \right) = 1$$

Z którego otrzymujemy :

$$r_2 - r_1 = n\lambda$$

Gdzie:

- r_2, r_1 - odległości od 2 różnych źródeł fali
- n - całkowity współczynnik wielokrotności ($n \geq 0, n \in C$)
- λ - długość fali

Wzmocnienie występuje w miejscach, gdzie różnica odległości od źródeł jest całkowitą wielokrotnością długości fali.

2. Warunek **wygaszenia** opisuje wzór:

$$\cos 2\Pi \left(\frac{r_2 - r_1}{2\lambda} \right) = 0$$

Stąd wynika, że:

$$r_2 - r_1 = (2n + 1)\frac{\lambda}{2}$$

Gdzie:

- r_2, r_1 - odległości od 2 różnych źródeł fali
- n - całkowity współczynnik wielokrotności ($n \geq 0, n \in C$)
- λ - długość fali

Wygaszenie występuje w punktach dla których różnica odległości od obu źródeł jest równa nieparzystej wielokrotności połowy długości fali.

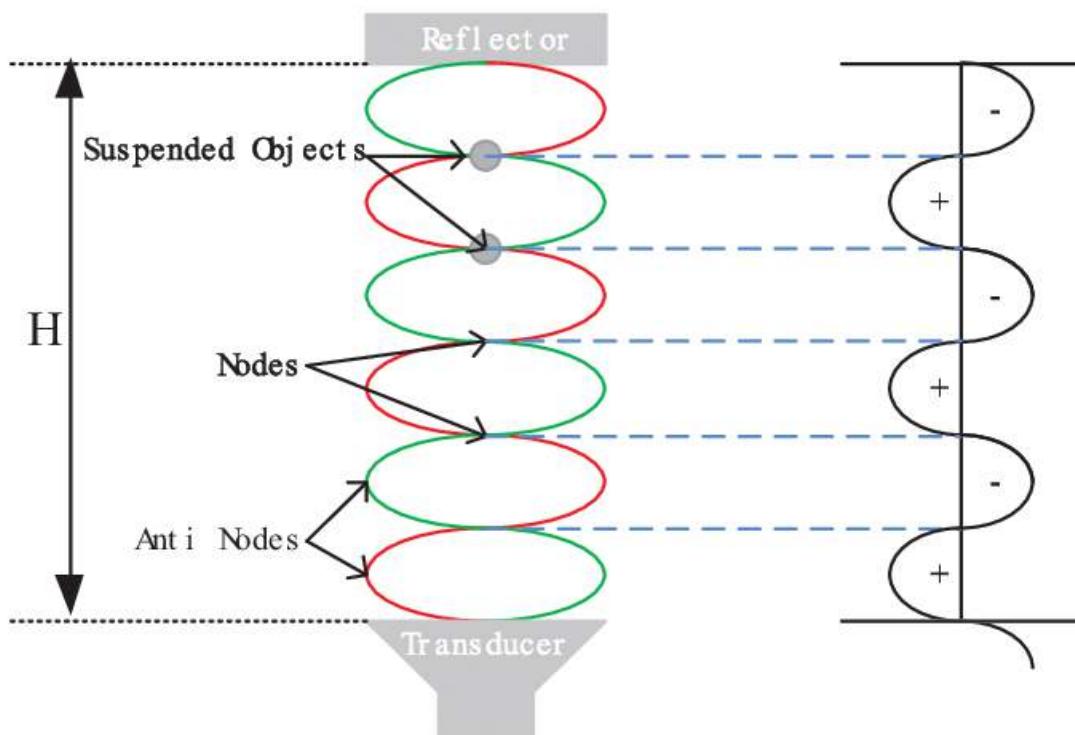
I.4.2. Fala stojąca

Działanie fali stojącej w akustycznej lewitacji polega na **utworzeniu stabilnej strefy**, w której fale dźwiękowe wytwarzane przez nadajniki nakładają się na siebie. Fala stojąca ma **węzły**, w których amplituda drgań jest równa zero, oraz punkty nazywane **strzałkami**, w których amplituda drgań jest maksymalna.

Zerowa amplituda w węzłach oznacza **zerowe drganie**, a więc obiekty umieszczone w tym położeniu nie ulegają wibracji. Jest on otoczony wysokim ciśnieniem zarówno z góry jak i dołu co powoduje jego stabilizację. To umożliwia zawieszenie obiektu w węzłach fali stojącej.

Aby fala stojąca powstawała sygnał wysyłany z górnego głośnika musi być przesunięty o 180° względem sygnału emitowanego z dolnego głośnika. Zgodnie z definicją fali stojącej, powstaje ona w wyniku interferencji dwóch fal o tej samej częstotliwości i względnego przesunięcia 180° .

Obiekt unoszący się w powietrzu z założenia powinien mieć rozmiar mniejszy od rozmiaru długości fali, tak aby zmieścił się w obszarze niższego ciśnienia. Jednak przy odpowiedniej sile wypadkowej dokonano już pierwszych, zakończonych sukcesem prób lewitacji przekraczającej tą wartość.



Rys. 5 Opis działania akustycznej lewitacji przy występowaniu zjawiska fali stojącej
 Źródło: researchgate.net/publication/320984182

Na Rys. 5 możemy zaobserwować, iż poprzez umieszczenie głośnika emitującego i reflektora naprzeciw siebie w odległości równej wielokrotności długości fali powstaje fala stojąca. W miejscach oznaczonych na schemacie jako szare koła tworzy się miejsce **zdolne do stabilnego utrzymania obiektu**. Oddalone są one od siebie o połowę długości fali i obrazują strefę niskiego ciśnienia pola akustycznego.

I.4.3. Wymagania lewitacji akustycznej

Możemy wyróżnić kilka **najważniejszych czynników** wpływających na lewitację akustyczną takie jak: rozmiar obiektu, jego masa i kształt, częstotliwość fali dźwiękowej, warunki środowiskowe (np. temperatura i wilgotność powietrza) czy też moc i konfiguracja głośników.

Do wykonania akustycznej levitacji są potrzebne:

- głośniki ultradźwiękowe (zbudowane z piezoelektryka, który pozwala na generowanie fal o wysokiej częstotliwości, jako wymóg skutecznej levitacji),
- generator sygnału,
- obiekt o odpowiednim kształcie i gęstości,
- odpowiednie środowisko (możliwie wolne od zakłóceń dźwiękowych, o odpowiedniej temperaturze i wilgotności powietrza).

I.4.4. Obliczanie opóźnień fazowych

Opóźnienie fazowe głośników ultradźwiękowych jest **kluczowe dla uzyskania fal stojących i skutecznej levitacji akustycznej**. W skrócie, jest to wzajemne przesunięcie między dwoma sygnałami wysyłanymi ze źródeł o tej samej częstotliwości.

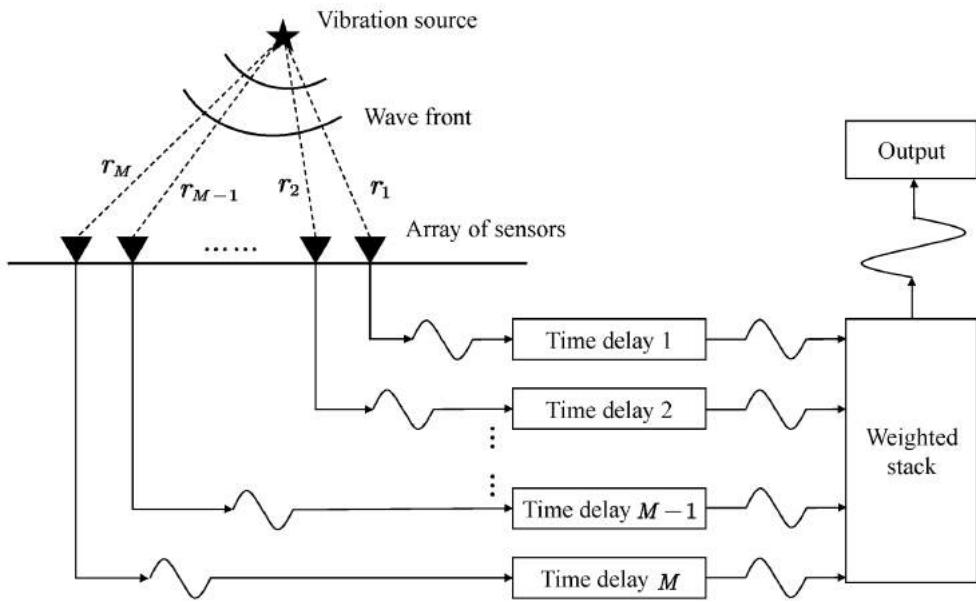
Aby obliczyć **opóźnienie fazowe** dla odpowiedniego nadajnika, należy najpierw określić odległość między nim a lewitującym obiektem. Następnie, biorąc pod uwagę prędkość dźwięku w powietrzu, obliczyć czas, który potrzebuje fala dźwiękowa, aby przebyć tę odległość. Ostatecznie różnica czasu między dwoma głośnikami odpowiada opóźnieniu fazowemu, które jest potrzebne do wytworzenia fali stojącej.

Zależność tą opisuje wzór:

$$p = \frac{(l-h) \times T}{\lambda}$$

gdzie:

- p - opóźnienie czasowe w μs ,
- l - odległość obiektu od środka głośnika w mm,
- h - prostopadła wysokość od obiektu do powierzchni z głośnikami w mm,
- T - okres sygnału wyrażony w μs ,
- λ - długość fali w mm.



Rys. 6 Obliczanie przesunięć fazowych
Źródło: mdpi.com/1424-8220/22/24/9924

W trakcie obliczania przesunięć fazowych istotna jest również **długość fali emitowanej przez głośniki ultradźwiękowe**. Wyraża się ją za pomocą prostego wzoru:

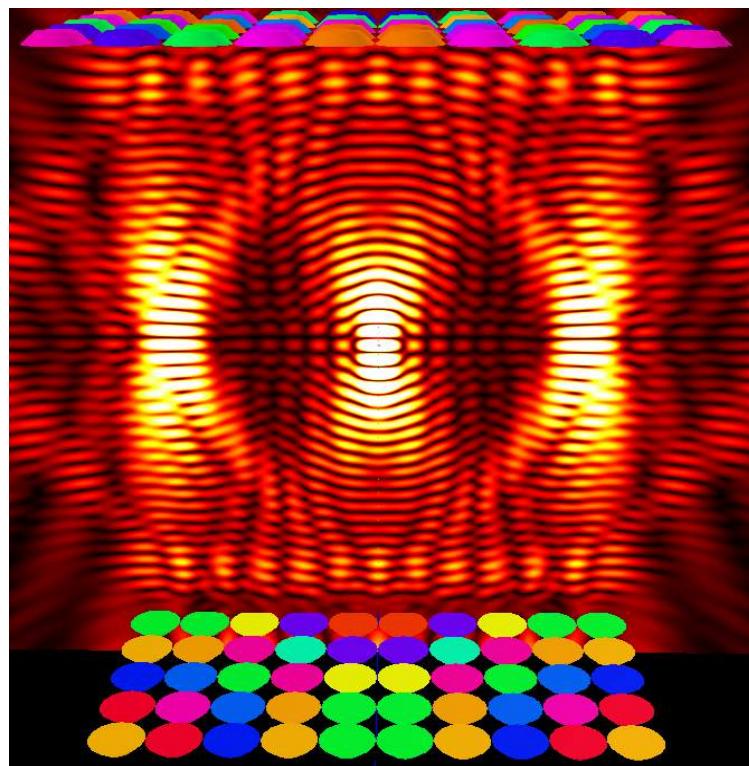
$$\lambda = \frac{v}{f}$$

gdzie:

- λ - długość fali wyrażona w metrach,
- v - prędkość medium (ośrodka),
- f - częstotliwość rezonansowa głośników.

I.4.6. Pole ciśnienia akustycznego

Wizualizacja pola ciśnienia akustycznego w modelu lewitacji pozwala na **analizę propagacji fali akustycznej emitowanej z głośników**. Dzięki temu możliwe jest określenie jak dźwięk będzie się rozchodził w przestrzeni, a także poziomów ciśnienia w różnych jej punktach czy miejsc tłumienia fali. Specjalistyczny program **3D Acoustic SIM** umożliwia wspomnianą obserwację punktów skupienia lewitatora, czego przykładem jest wizualizacja zaprezentowana na Rys. 7. **Poziom natężenia pola ciśnienia akustycznego** jest zobrazowany jako różny poziom jasności pól na wizualizacji, zakładając że im jaśniejsze, tym większa jest jego wartość.

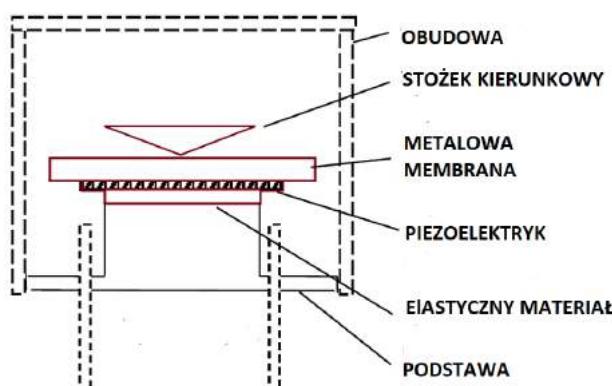


Rys. 7 Wizualizacja pola ciśnienia akustycznego dla układu 200 głośników w ustawieniu 10x10.

Źródło: opracowanie własne na podstawie programu 3D Acoustic Sim

I.4.7. Piezoelektryki, czyli jak powstaje fala dźwiękowa

Przetworniki ultradźwiękowe są wykonane z **materiału piezoelektrycznego**, który na skutek odkształcenia mechanicznego wytwarza napięcie elektryczne. Proces ten można odwrócić poprzez przyłożenie do niego napięcia. Gdy takie napięcie przykłada się z określoną częstotliwością - tworzy on falę akustyczną, poprzez przekazywanie swoich drgań cząsteczkom powietrza. Jego ruch powoduje zmiany w zagęszczeniu powietrza, na skutek czego powstają naprzemienne obszary niskiego i wysokiego ciśnienia, czyli **fala dźwiękowa**.



Rys. 8 Budowa głośnika ultradźwiękowego

Źródło: opracowanie własne na podstawie

www.researchgate.net/figure/Ultrasonic-transducer-structure_fig2_30499312

I.5. Zalety i wady transportu za pomocą fali dźwiękowej

Ogromną **zaletą** transportowania obiektów przy pomocy lewitacji akustycznej jest jej **precyzja**, która pozwala na dokładne manipulowanie obiektem i kontrolowanie jego ruchu. Oprócz tego metoda ta pozwala na **pracę z materiałami wrażliwymi** na skażenie, delikatnymi mechanicznie. Głównym plusem jest jednak **możliwość lewitacji obiektów niemagnetycznych**. Dało to m.in. rozwój obecnej medycyny w której przeprowadzony został pierwszy bezdotykowy transport komórek w żywym organizmie.

Nie można jednak zapominać o tej drugiej stronie medalu. Metoda ta bowiem wymaga **stosowania specjalistycznych elementów**, takich jak np. głośniki ultradźwiękowe, lub odpowiednie generatory sygnałów. Również w przypadku obiektów o większych rozmiarach (rzędu kilkunastu centymetrów wzwyż) lewitacja ta jest nieskuteczna lub wręcz niemożliwa ze względu na **ograniczenia natury fizycznej**. Analiza oraz wyszukiwanie problemów w tym rodzaju lewitacji jest także niemałym wyzwaniem, gdyż wymaga ona wielkiego doświadczenia i dużej wiedzy w dziedzinie akustyki. Sama lewitacja jest **wrażliwa na wszelkie zmiany** w zakresie warunków zewnętrznych.

MOCNE STRONY

S

- Precyza pozycjonowania
- Całkowita kontrola ruchu
- Szeroki zakres lewitowanych obiektów
- Praca z obiektami wrażliwymi na dotyk
- Znaczna prędkość poruszania
- Bezinwazyjność
- Wiele potencjalnych zastosowań

SŁABE STRONY

W

- Wrażliwość na czynniki środowiskowe
- Mała masa obiektów
- Specjalistyczny sprzęt
- Problematyczna detekcja błędów

LEWITACJA AKUSTYCZNA W TRANSPORCIE

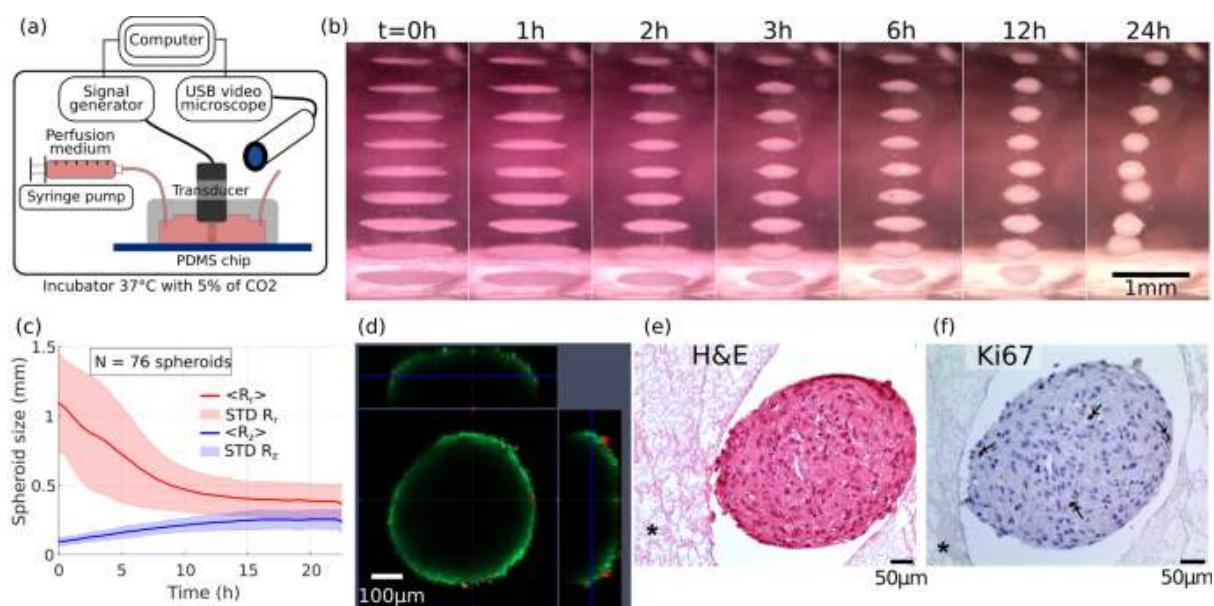


Rys. 9 Wady i zalety lewitacji akustycznej

Źródło: opracowanie własne

I.6. Lewitacja akustyczna i jej wykorzystanie w praktyce

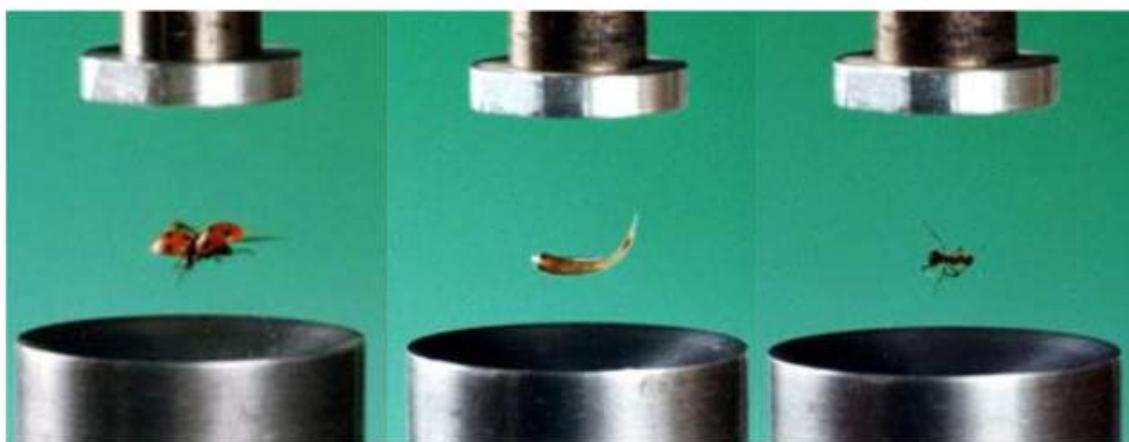
Zgodnie z badaniami opublikowanymi w czasopiśmie "Proceedings of the National Academy of Sciences", lewitacja akustyczna jest wykorzystywana w medycynie do **transportu i manipulacji komórek** w ciele pacjenta. W badaniach wykorzystano pole akustyczne o wysokiej częstotliwości, aby unosić i przesuwać pojedyncze komórki wewnątrz ciała zwierzęcia. W badaniach zaobserwowano, że metoda ta może być wykorzystana do **transportu komórek macierzystych**, które mogą pomóc w **regeneracji uszkodzonych tkanek**. Zabiegi takie są bezinwazyjne, co zwiększa bezpieczeństwo pacjenta i redukuje czas rekonwalescencji. Metoda ta mogłaby więc wnieść medycynę na zupełnie nowy poziom, a także zrewolucjonizować dziedziny takie jak np. sport.



Rys. 10 Samoorganizacja komórek poddanych akustycznej lewitacji.

Źródło: nature.com/articles/s41598-021-87459-6

Lewitacja akustyczna została również wykorzystana w **unoszeniu organizmów żywych** o małej masie. Próba takiego badania (patrz: Rys. 10) miała na celu zbadanie reakcji lewitowanych organizmów, oraz potencjalną próbę kontroli nad nimi.



Rys. 11 Lewitacja organizmów żywych
Źródło: nature.com/articles/news061127-6

Co ważne, w całym procesie **nie zaobserwowano negatywnych skutków w ciele zwierząt po ukończonym zabiegu**. Niewątpliwie dalsza praca nad tym zagadnieniem umożliwiłyby przykładowo wykonywanie precyzyjnych operacji.

Lewitacja akustyczna jest stosowana również w branży kosmicznej, do badania **fizyki ruchu kropel w stanie nieważkości**. Prace te pozwalają na lepsze zrozumienie mechaniki płynów, i jaką mają na nią wpływ takie właściwości substancji jak np. lepkość.

Jest ona obecnie rutynowo stosowana wraz z innymi technologiami lewitacji do pozycjonowania ciał stałych i cieczy bez kontaktu z innymi elementami. Di Chen i Junru Wu z University of Vermont wykazali, że lewitacja akustyczna może być stosowana do **usuwania kurzu i pyłów** z powierzchni taki jak np. panele słoneczne. Pierwsze projekty zakładają użycie tej metody w **walce z burzami piaskowymi**, zanieczyszczającymi łaziki na Marsie i Księżyku.



Rys. 12 Łazik Nasa Curiosity, zdjęcie wykonane po burzy piaskowej 15 czerwca 2018
Źródło: NASA/JPL-Caltech

Zastosowaniem, które niezwykle przykuło naszą uwagę to wykorzystanie lewitacji akustycznej do produkcji, umożliwiając tym samym bezdotykowy montaż produktów. Naszym zdaniem proces produkcji posiada jednak naprawdę wiele obszarów, w których lewitacja akustyczna mogłaby zrewolucjonizować jego funkcjonowanie. Jednym z nich, i tym którego ogromny potencjał chcielibyśmy wykorzystać, jest **bezdotykowy transport elementów, półproduktów i produktów w procesie produkcyjnym**. Logistyka w tym wydaniu weszłyby na zupełnie nowy poziom, zapewniając w dłuższym czasie **zmniejszenie kosztów** w naprawdę wielu wymiarach:

- zmniejszenie zapotrzebowania na pracowników obsługujących magazyny,
- brak konieczności inwestycji w wiele różnych rodzajów systemów transportu
- zmniejszenie ilości uszkodzonego towaru ze względu na pełną automatyzację

Wiedząc już jak ogromny potencjał jest w tej dziedzinie, nie możemy przejść obok tego obojętnie. Chcielibyśmy więc, aby nasze rozwiązanie było pierwszym krokiem w stronę ogromnej rewolucji jaką jest **logistyka oparta na lewitacji akustycznej**.

I.7. Wykorzystane programy

KiCad

Jest to oprogramowanie służące do **tworzenia schematów obwodów elektrycznych**. Pozwala on na tworzenie projektów od podstaw, a także na ich edycję i weryfikację. Do zawartości tego oprogramowania należy szereg samodzielnego programów, odpowiedzialnych za poszczególne funkcje w procesie tworzenia schematu, takie jak np. Eeschema do edycji schematów czy też Pcbnew do edycji obwodów drukowanych.

Autodesk Fusion 360

Program ten jest jednym z licznych programów oferowanych przez firmę Autodesk. Jest to kompleksowe oprogramowanie umożliwiające tworzenie intuicyjnych **projektów 3D**, zarządzanie **dużą ilością danych**, wykonywanie **skomplikowanych symulacji**, a także tworzenie dokumentacji w obrębie jednego programu. Posiada on oprócz podstawowej wersji wiele rozszerzeń przeznaczonych dla ścisłe określonej grupy odbiorców. W naszym projekcie będziemy używać podstawowej wersji programu na licencji edukacyjnej.

Apache NetBeans IDE

Jest to darmowe tzw. open-source oprogramowanie pozwalające na **tworzenie aplikacji** webowych, desktopowych a także mobilnych. Obsługuje on aplikacje w wielu językach takich jak HTML5, PHP, JAVA czy też C++. Zapewnia kompletną obsługę całego cyklu tworzenia aplikacji, od tworzenia projektu, poprzez debugowanie, symulowanie i w końcowej wersji jego wdrażanie. Aplikacja ta umożliwiła nam pracę z symulatorem, który był napisany w jej środowisku.

Visual Studio Code

Visual Studio Code (VSCode) to wieloplatformowe **środowisko programistyczne** stworzone przez Microsoft. Używane do pisania, debugowania i publikowania kodu w różnych językach programowania, takich jak: JavaScript, HTML, Python, C ++, PHP i wiele innych. VSCode jest prostym w obsłudze oprogramowaniem umożliwiającym pracę z łatwym interfejsem oraz wieloma narzędziami.

Logic Saleae

To oprogramowanie służące do **analizy sygnałów cyfrowych**. Pozwala na podłączenie analizatora stanów logicznych, pozwalającym na obserwację i analizę sygnałów cyfrowych z różnych źródeł takich jak mikrokontrolery, sensory, układy FPGA, itp. Program posiada wiele funkcji, takich jak analiza protokołów komunikacyjnych, dekodowanie sygnałów, filtrowanie, triggerowanie.

Ultimaker Cura

Jest to bezpłatne oprogramowanie do **przygotowania modeli 3D do druku**. Jest to narzędzie typu "slicer", co oznacza, że dzieli on model 3D na warstwy i generuje potrzebny do procesu druku **kod G**. Cura obsługuje szeroki zakres formatów plików, w tym STL, OBJ, X3D i 3MF, a także pozwala na łatwe dostosowanie ustawień drukowania, takich jak grubość warstwy, prędkość drukowania i infill (wypełnienie).

Quartus Altera

W ramach projektu zdecydowaliśmy się na użycie programu specjalistycznego oferowanego przez firmę Intel o nazwie Intel Quartus II. Jest to oprogramowanie CAD dedykowane dla układów FPGA i CPLD służące do **projektowania programowalnych urządzeń logicznych**. Charakteryzuje się on dość skomplikowanym interfejsem, lecz zawiera bardzo wiele możliwości, od tworzenia prostych układów logicznych w edytorze graficznym języków takich jak Verilog czy VHDL, aż do tworzenia własnych bloków funkcyjnych i opisywania sprzętowego układu.

Wersja, której użyliśmy to **Quartus II 13.0sp1 (64-bit) Web Edition**, gdyż jest ona kompatybilna z wykorzystywaną przez nas płytą Altera Cyclone II EP2C5T1448N.



Rys. 13 Listing użytych programów

Źródło: opracowanie własne

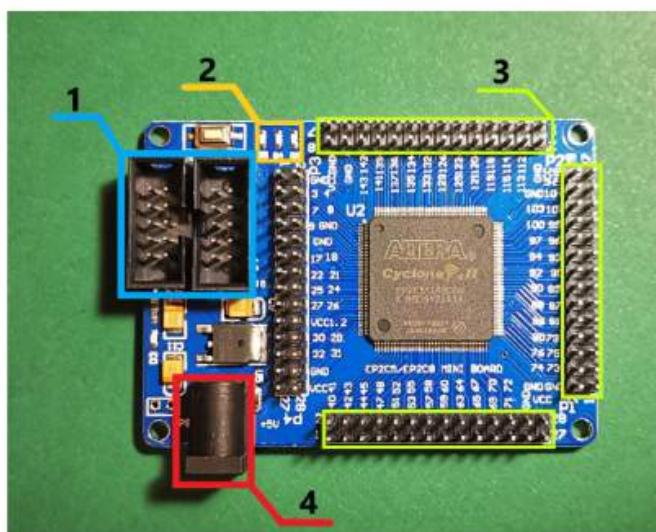
I.8. Opis elementów

FPGA EP2C5144TC8N

FPGA (Field Programmable Gate Array) to rodzaj cyfrowego układu programowalnego, często określany również jako bezpośrednio **programowalna macierz bramek**. Układy FPGA są w stanie **przetwarzać dane w sposób równoległy**, dzięki czemu operacje są wykonywane w tym samym czasie, a nie sekwencyjnie, przez co bardzo dobrze nadają się do zastosowań obliczeniowych o wysokiej wydajności.

Budowa układów FPGA w zależności od producenta różni się. Charakteryzuje się ona podstawowymi elementami (Rys. 14) takimi jak:

- programowalne interfejsy JTAG i ASP (oznaczone jako 1),
- sygnalizatory LED (oznaczone jako 2)
- wyprowadzenia GPIO (oznaczone jako 3),
- zasilanie DC 5V (oznaczone jako 4).



Rys. 14 FPGA EP2C5144C8N - wykaz podzespołów
Źródło: opracowanie własne

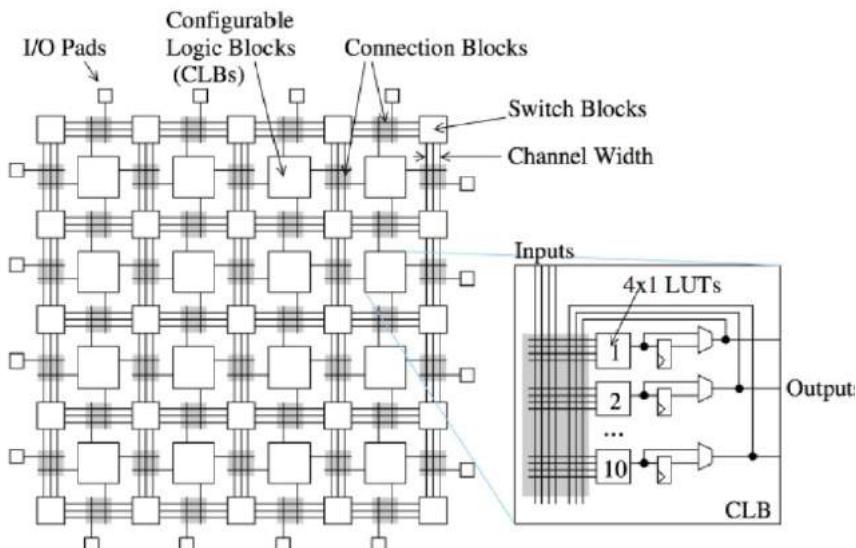
W naszym projekcie użyliśmy płytka Altera **Cyclone II generacji** - EP2C5144TC8N, która posiada:

- wyprowadzenia zasilania 3,3 V i 1,2 V,
- pojedyncze wejście do zasilania 5 V poprzez gniazdo DC 2,1 mm,
- 89 dostępnych pinów użytkowych,
- wbudowany zegar 50 MHz podłączony do m.in. pinu 17,
- interfejs JTAG,
- szeregowy interfejs programowania ASP (ang. Active Serial Port) z wbudowaną pamięcią EPROM 4MBit,
- 4608 bitów pamięci na blok (w tym 512 bitów parzystości),

- 4K x 26 bitów (razem 119 808 bitów),
- do 1,1 Mb dostępnej pamięci RAM,
- możliwość operacji na sygnałach o częstotliwości do 260 MHz.

Implementacja danego projektu do układów FPGA polega na użyciu jednego lub więcej języków programowania ścisłe dostosowanych pod te układy. Użyte przez nas języki programowania to:

- **VHDL** (ang. *Very High Speed Integrated Circuit Hardware Description Language*) stworzony w 1983 roku pod przewodnictwem departamentu obrony Stanów Zjednoczonych. Język ten posiada zestaw gotowych do użycia bibliotek stworzonych przez Instytut Inżynierów Elektryków i Elektroników (IEEE). Biblioteki te definiują zestaw wartości logicznych, które mają być używane w języku VHDL,
- **język schematów blokowych**, oparty na schematach blokowych. Umożliwia wizualizację całego programu i jest łatwiejszy w zastosowaniu w porównaniu z językami opartymi na tekście. Posiada wiele gotowych elementów logicznych, które za pomocą prostego interfejsu można ze sobą łączyć. W zakresie podstawowym istnieje możliwość zmiany niektórych parametrów w tych blokach.



Rys. 15 Struktura FPGA

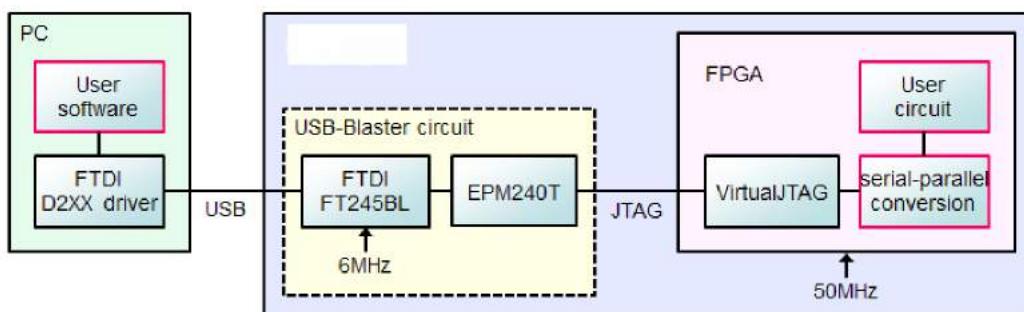
Źródło: opengenus.org/structure-of-field-programmable-gate-array-fpga

USB BLASTER

Aby zaprogramować FPGA potrzebny jest programator USB-blaster. Zapewnia on możliwość programowania urządzeń CPLD i FPGA. Umożliwia obsługę trybów JTAG, Passive Serial i Active Serial Programming. Podłączany jest on do komputera za pomocą uniwersalnej wtyczki USB, a do płytki drukowanej za pomocą 10-pinowego złącza. Jego główną zaletą jest to, że zapewnia wsparcie programistyczne dla szeregowych urządzeń konfiguracyjnych, w odróżnieniu od jego zamiennika - kabla ByteBlasterMV.



Rys. 16 Zastosowany USB-Blaster
Źródło: opracowanie własne



Rys. 17 Schemat komunikacji z USB-BLASTER
Źródło: github.com/pgate1/USB-Blaster_UART

W celu użycia USB-Blaster, należy podłączyć go do komputera za pomocą kabla USB i do układu FPGA za pomocą kabla IDC 10. Wymagane jest zainstalowanie sterowników USB-Blaster na komputerze, które umożliwiają komunikację między komputerem a USB-Blaster.

Gdy komunikacja zostanie ustawiona, użytkownik może użyć oprogramowania Quartus II. Komunikacja odbywa się za pomocą poleceń wysyłanych z oprogramowania na komputerze do USB-Blaster, który następnie przekazuje je do układu FPGA za pomocą interfejsu JTAG. Układ FPGA przetwarza te polecenia i przesyła odpowiedź do USB-Blaster, który następnie przekazuje ją do komputera.

TCT40-16T

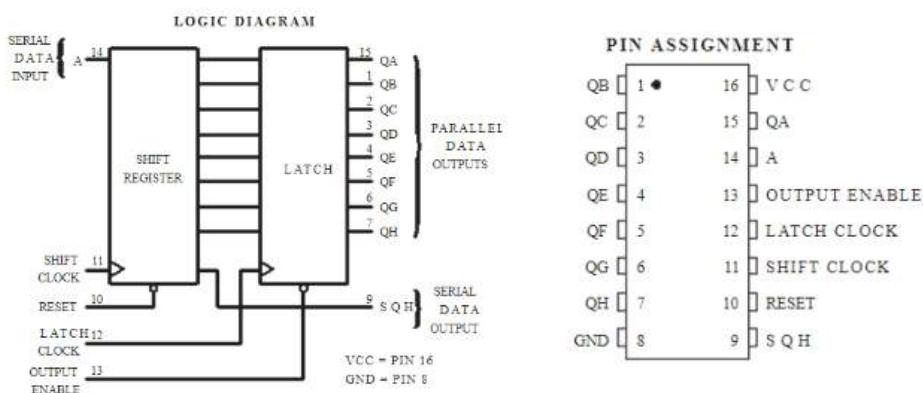
Głośniki ultradźwiękowe TCT40-16T są jednym z modeli, które są w stanie wygenerować falę dźwiękową niezbędną do lewitacji obiektu w powietrzu. Oprócz nich do uzyskania podobnego efektu można zastosować Manorshi 10 mm lub FBULS1007PT 10 mm. Wszystkie te głośniki charakteryzują się małą średnicą (10-16 mm), co sprzyja w ich zastosowaniu w układach z kilkudziesięcioma lub nawet kilkuset głośnikami. Częstotliwość rezonansowa głośników TCT40-16T wynosi 40.0 ± 1 kHz.



Rys. 18 Zastosowany głośnik ultradźwiękowy
Źródło: opracowanie własne

Rejestr przesuwny HC595

Rejestr HC595 składa się z 8-bitowego bufora przesuwnego, który umożliwia przesunięcie sekwencji bitów z jednego rejestru do drugiego. Może być sterowany przy użyciu sygnałów zewnętrznych, takich jak sygnały zegara lub sygnały danych.



Rys. 19 Opis wyjść sterowników mosfet 74HC595
Źródło: microchip.com

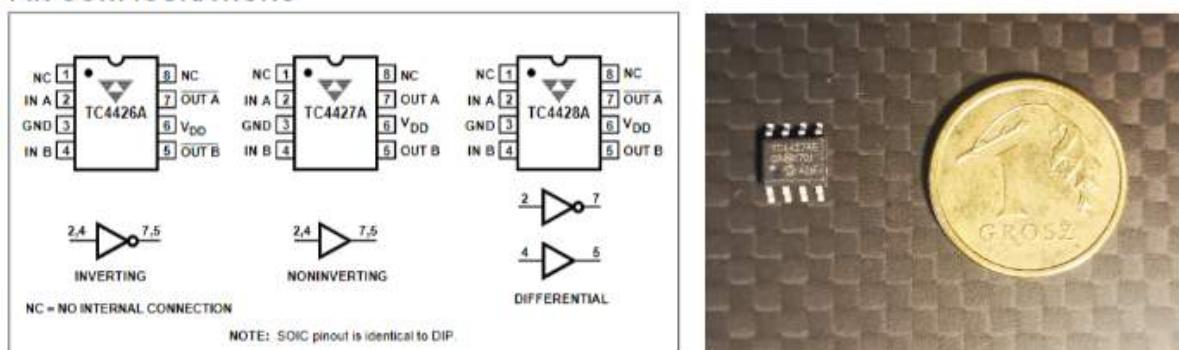
Posiada on wejście danych - ser, latch clock, który służy do aktualizowania stanów wyjść, shift clock do przesuwania bitów na kolejne wyjścia, reset do czyszczenia poziomów logicznych i output enable do regulowania możliwości odczytu danych wyjściowych.

TC4427

TC4427 to **2-kanałowy sterownik tranzystorów MOSFET**, który zapewnia szybkie przejścia stanu ON/OFF i jest w stanie operować pod napięciem potrzebnym do odpowiedniego działania głośników. Układ ten ma **małe opóźnienie propagacji sygnału** oraz **niskie zużycie energii**, co umożliwia jego efektywne stosowanie w aplikacjach wymagających szybkiego i precyzyjnego sterowania.

W prototypie jest on zastosowany jako układ pośredniczący między 74HC595 a głośnikami. Jako że 74HC595 (w pierwszej wersji FPGA) operuje na 4,4 V (FPGA 3,3 V), a głośniki do poprawnego działania potrzebują wyższego napięcia, wymagany jest więc **układ przekształcający sygnał** na wyższe napięcie - TC4427.

PIN CONFIGURATIONS

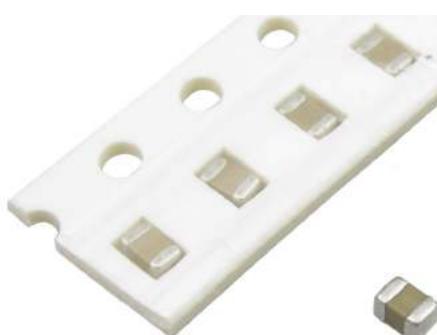


Rys. 20 TC4427 konfiguracja pinów

Źródło: opracowanie własne

Kondensatory 100 nF

Układy w trakcie pracy mogą generować wahania napięcia, przez równoczesny pobór prądu, które mają negatywny wpływ na stabilność układu i jakość lewitacji. W celu zredukowania tych zakłóceń, zdecydowaliśmy się dodać kondensatory, które będą pomagać w ustabilizowaniu napięcia zasilania przy poszczególnych układach.



Rys. 21 Zastosowany kondensator 100 nF

Źródło: product.tdk.com/en/products/capacitor

Przetwornica 230V/12V AC-DC 2A

Służy ona do **zamiany napięcia** przemiennego 230 V na napięcie stałe 12 V.

Inne z parametrów to :

- prąd wyjściowy: 2 A,
- częstotliwość pracy: 50-60 Hz,
- moc wyjściowa: 24 W,
- metoda modulacji: modulacja szerokości impulsu (PWM).

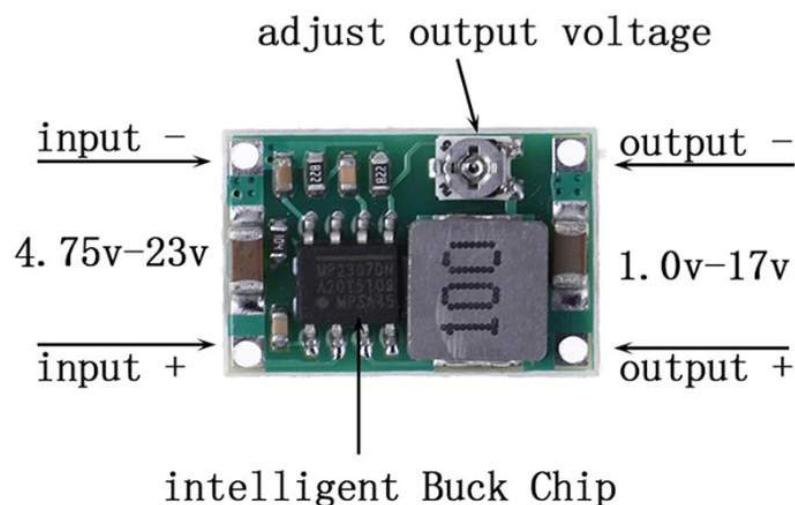


Rys. 22 Przetwornica 230V/12V AC-DC 2A

Źródło: pl.aliexpress.com/i/4000351150023.html?gatewayAdapt=glo2pol

Przetwornica Step-Down Mini360 DC-DC

Przetwornica Step-down Mini360 to urządzenie do **zamiany napięcia z wyższego poziomu na niższy**. Maksymalne napięcie wejściowe to od 4,75 V do 23 V, a maksymalne napięcie wyjściowe to od 1 V do 17 V. Prąd wyjściowy wynosi 3 A w szczytce, a ciągły jest równe 1,8 A, natomiast wydajność wynosi 96%. W naszym projekcie zastosowaliśmy go do zasilania układów hc595.



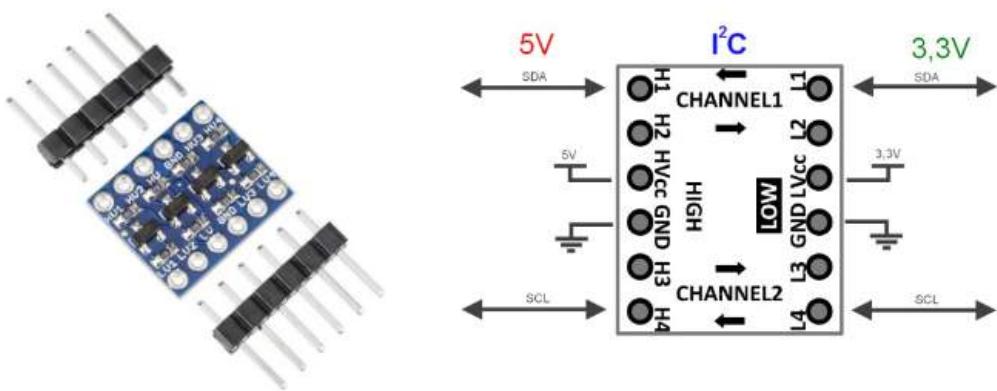
Rys. 23 Przetwornica Step-Down

Źródło: www.ebay.pl/item/155001134900

Konwerter poziomów logicznych I2C

Konwerter poziomów logicznych to układ elektroniczny, który służy do przekształcania poziomów napięcia pomiędzy różnymi standardami logicznymi. Jest to przydatne wtedy, gdy sygnały z jednego układu lub urządzenia muszą być przetworzone i przesłane do innego, który korzysta z innego standardu napięcia.

Konwerter poziomów logicznych może działać w dwóch kierunkach, przekształcając napięcie sygnału z niższego na wyższy poziom lub z wyższego na niższy. Najczęściej stosowane standardy logiczne to TTL, CMOS, LVCMOS, LVTTL i LVDS. W naszym wypadku potrzebny on jest by umożliwić komunikację programu komputera (5 V) z Fpga (3,3 V).



Rys. 24 Konwerter stanów logicznych I2C

Źródło: pl.aliexpress.com/i/1005003157047080.html?gatewayAdapt=glo2pol

II. Projekt techniczny

W trakcie naszej pracy kierowaliśmy się **praktycznością** oraz **uniwersalnością** naszego rozwiązania w ramach procesów logistycznych, ale nie bez znaczenia był dla nas komfort i prostota użytkowania czy też użyteczność urządzenia wraz z jego estetyką.

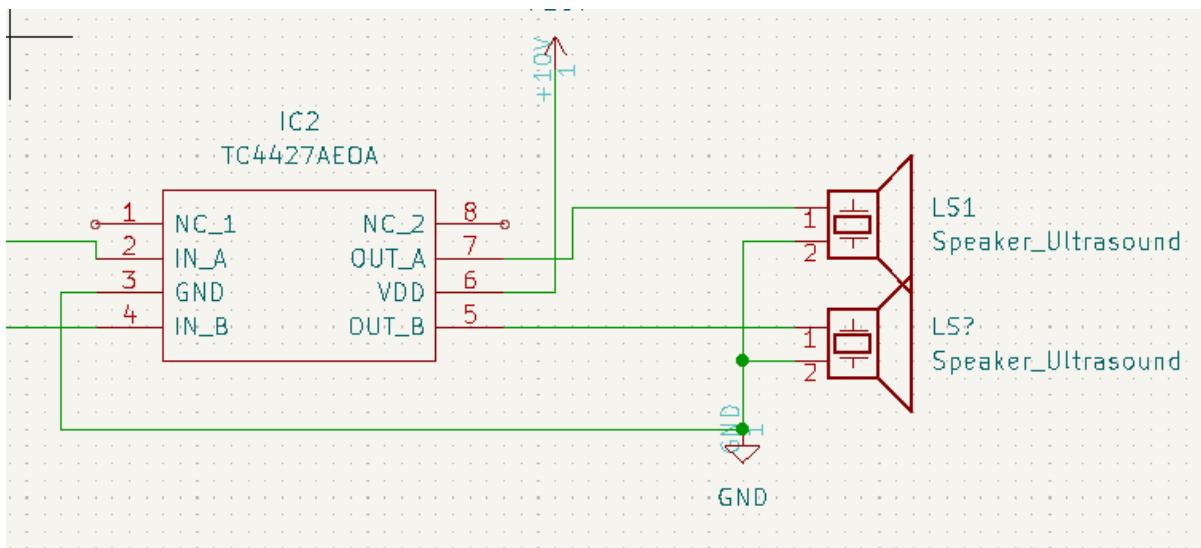
Praca przebiegała w następujących krokach:

1. zapoznanie z teorią fizycznego modelu lewitacji ultradźwiękowej,
2. opracowanie pierwszego prototypu transportu obiektów,
3. realizacja prototypu oraz testowanie urządzenia,
4. diagnostyka i naprawa usterek,
5. projekt drugiego prototypu,
6. stworzenie programu trajektorii ruchów w 3D,
7. kalibracja i finalizacja końcowego projektu.

II.1. Pierwszy prototyp FuturFlow

Pierwszy projekt prototypu składał się z:

- 32 głośników ultradźwiękowych TCT40-16T,
- mikrokontrolera FPGA Altera Cyclone II EP2C5144TC8N,
- drukowanych podstawek,
- kondensatorów 100 nF,
- programu opracowanego w Quartusie,
- sterowników tranzystorów Mosfet TC4427aea.



Rys. 25 Schemat elektryczny układu głośników

Źródło: opracowanie własne

Prototyp urządzenia transportującego został stworzony w celu zapoznania się z tematem akustycznej lewitacji i zebraniu pierwszych informacji na temat jego ograniczeń oraz możliwości, które moglibyśmy zaimplementować w prototypie finalnym. Miał on na celu stworzenie najefektywniejszej wersji końcowej (m.in. zminimalizowanie usterek) oraz charakteryzował się:

- większą mobilnością, gdyż jego rozmiary znaczaco różniły się od wersji końcowej,
- łatwiejszym dostępem do podzespołów, co umożliwiło łatwość testowania urządzenia bez konieczności zdejmowania obudowy.

Porównując możliwe opcje układu głośników doszliśmy do wniosku, że pod względem kosztu, poziomu zaawansowania oraz zakresu roboczego, najlepszym wyborem będzie ustawienie ich w **układzie kwadratowym**. Pierwszy prototyp zakładał użycie **32 głośników** (po 16 z każdej strony).

II.1.1. Złożenie prototypu

II.1.2. Przygotowanie stanowiska

Przygotowanie stanowiska zaczęliśmy od komplementowania sprzętu potrzebnego do realizacji naszego pierwszego prototypu. Użyte przez nas przyrządy to m.in.:

- generator funkcyjny DDS FY6800 60 MHz - 2 kanałowy,
- oscyloskop cyfrowy OWON 4-kanałowy 200 MHz, 1 Gsps,
- multymetr cyfrowy LCD 2x SONDY,
- zasilacz DC 12 V.



Rys. 26 Przyrządy na stanowisku pracy

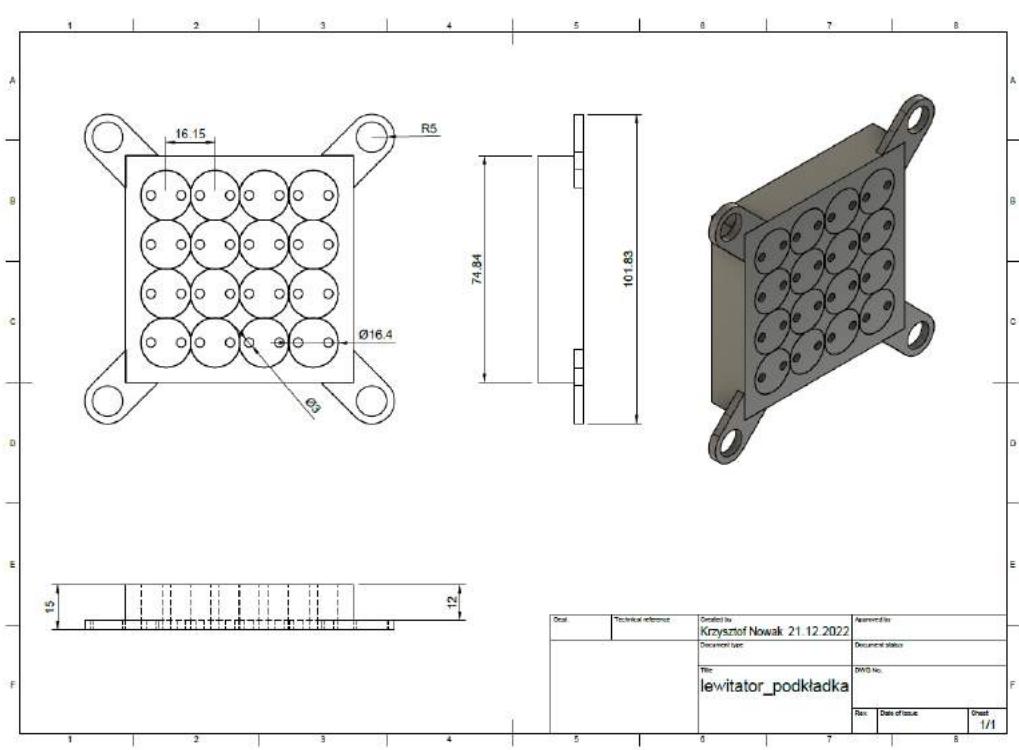
Źródło: zdjęcia własne

II.1.3. Elementy mechaniczne pierwszego prototypu

Prototyp składał się w głównej części ze śrub M8 podtrzymujących drukowane podstawki mieszczące głośniki. Poprzez dokręcanie nakrętek na śrubach mogliśmy uzyskać precyzyjną regulację podstawek względem siebie.

Do zaprojektowania drukowanej podstawki użyliśmy programu Autodesk Fusion 360. Biorąc pod uwagę wysokość głośników dobraliśmy odpowiednio wysokość podstawki, tak aby włożone w nią głośniki zrównały się z jej poziomem. Średnica użytych przez nas głośników TCT40-16T wynosiła $16+0,1$ mm. Założyliśmy, że aby uniknąć przymocowywania głośników dostosujemy otwory tak, aby wchodziły one na wciśnięcie, dlatego też ustaliliśmy średnicę każdego otworu na 16 mm.

Projekt 3D płytki przedstawia Rys. 27.

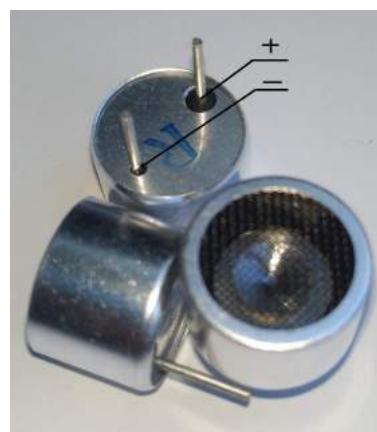


Rys. 27 Projekt drukowanej podstawki
Źródło: opracowanie własne

II.1.4. Przebieg pracy

II.1.4.1 Sprawdzenie polaryzacji głośników ultradźwiękowych

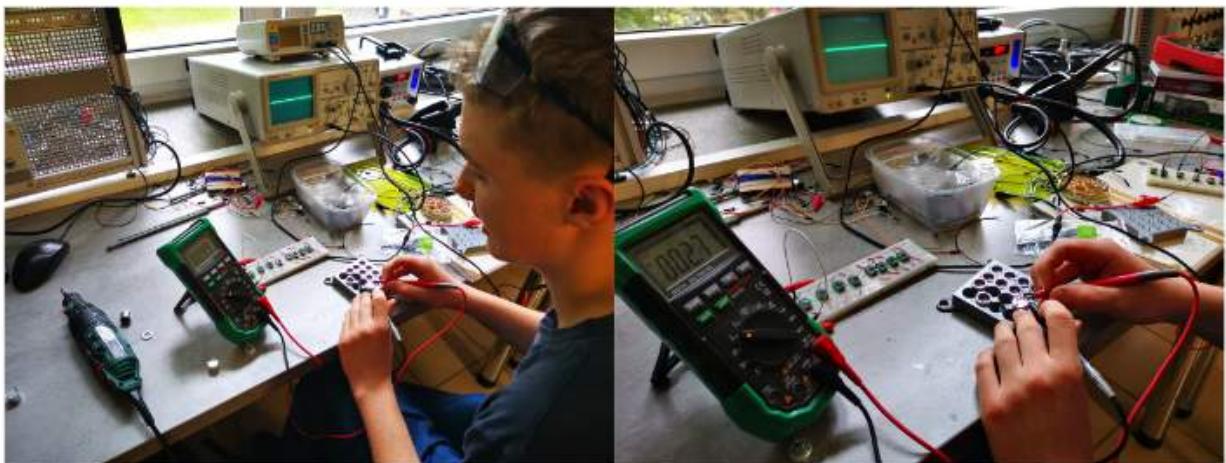
Każdy głośnik posiada oznaczenie polaryzacji na nóżkach. W przypadku głośników ultradźwiękowych polaryzacja jest oznaczana na etapie produkcji. Nóżka o polaryzacji dodatniej posiada widoczne oznaczenie w postaci czarnego oznakowania.



Rys. 28 Oznaczenie polaryzacji przez producenta
Źródło: opracowanie własne

Niestety jak przyszło nam się dowiedzieć polaryzacja oznaczona na obudowie nie zawsze odpowiada polaryzacji rzeczywistej. Każdy z głośników musiał być sprawdzony i oznaczony, aby wszystkie działały w tej samej fazie (patrz: Rys. 30). Nie ma znaczenia czy głośniki będą spolaryzowane negatywnie czy pozytywnie, lecz ważne jest żeby każdy był podłączony jednakowo.

Nóżki głośnika podłączaliśmy do sond multimetru i porównywaliśmy otrzymane sygnały oznaczając odpowiednio każdą nóżkę.

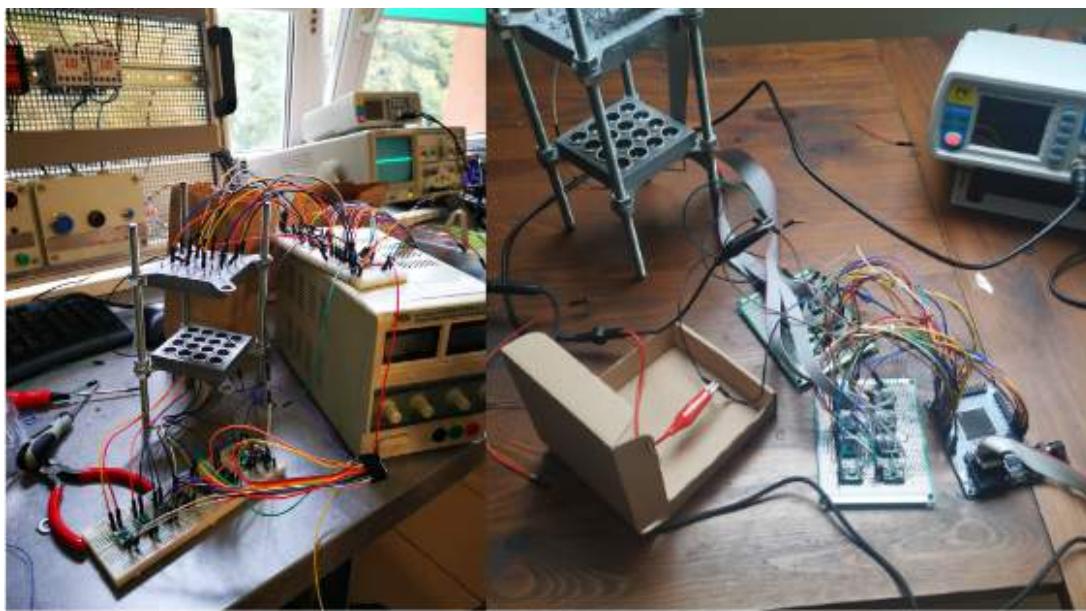


Rys. 29 Proces sprawdzania polaryzacji głośników
Źródło: zdjęcia własne

II.1.4.2 Montowanie elektroniki

Po sprawdzeniu głośników, zamontowaliśmy je w podstawkach, nóżki głośników oznaczone minusem podpięliśmy razem, a drugą nóżkę poprowadziliśmy na wyjście sterowników tc4427.

Początkowo wszystkie elementy były podłączone do uniwersalnej listwy. Niestety, takie rozwiązanie było bardzo niedokładne, gdyż w czasie testowania sygnałów z głośników przy pomocy oscyloskopu występowało wiele przerw między połączeniami, co skutkowało zniekształconymi przebiegami lub ich brakiem. Zdecydowaliśmy się wymienić ją na drukowaną płytę PCB.



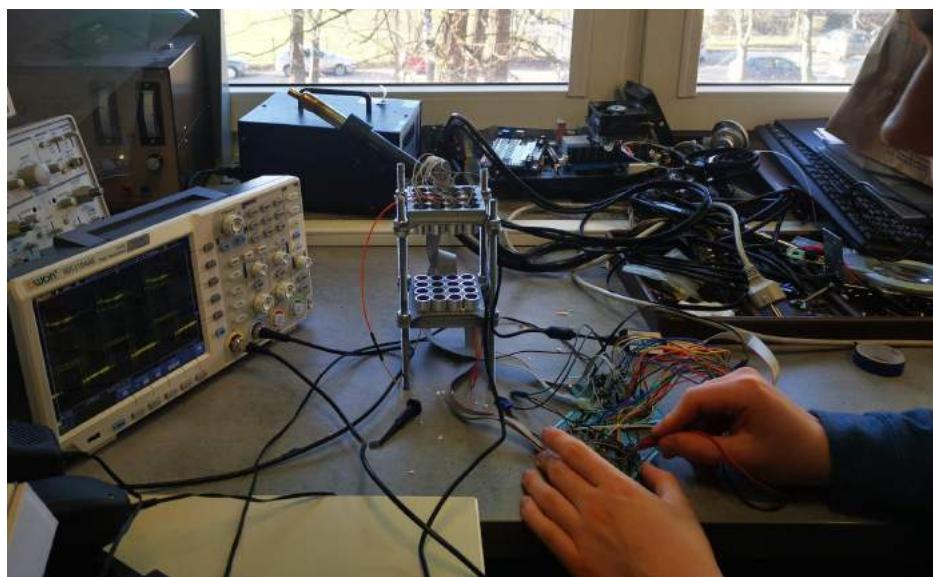
PRZED

PO

Rys. 30 Optymalizacja połączenia elementów
Źródło: zdjęcia własne

II.1.4.3 Diagnostyka i testowanie głośników

Przy użyciu oscyloskopu cyfrowego sprawdzaliśmy uzyskiwane przebiegi sinusoidalne na głośnikach. Jeżeli były one zniekształcone lub odbiegały od wzorcowego sygnału, był to najczęściej problem samego głośnika lub spalonych układów scalonych. Po dogłębnej analizie i naprawie usterek, wszystkie sygnały były jednakowe.



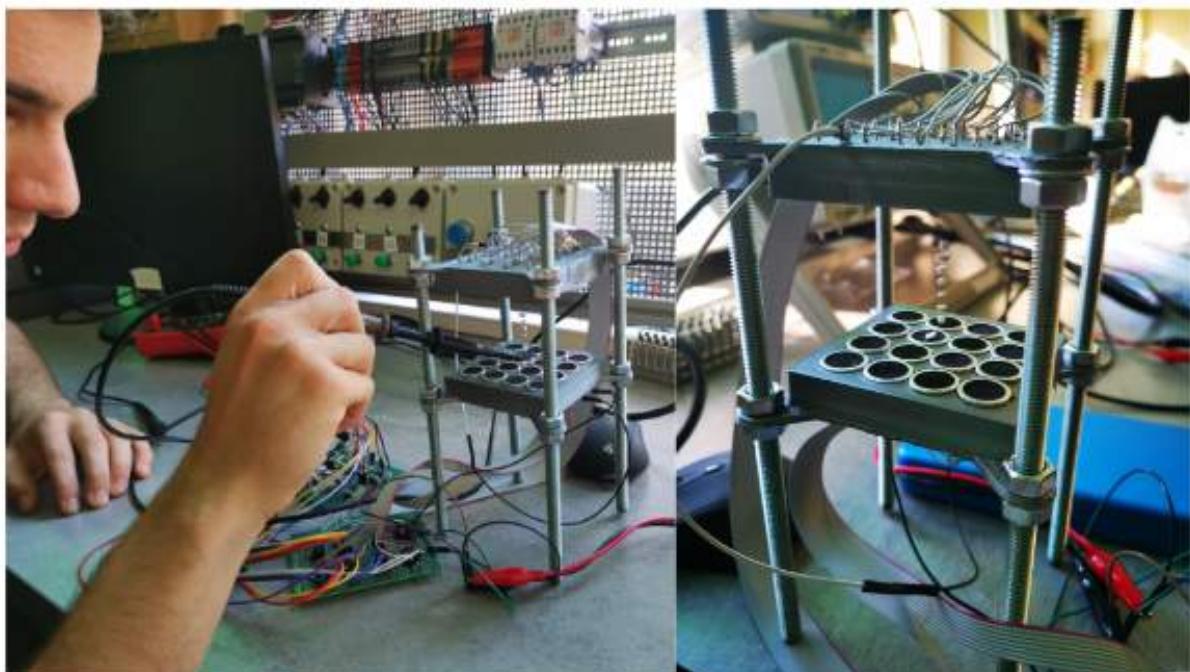
Rys. 31 Sprawdzanie przebiegów sygnałów prostokątnych
Źródło: zdjęcie własne

II.1.5 Pierwsza lewitacja

Gdy wszystkie głośniki emitowały pożądany sygnał przeszliśmy do testowania lewitatora. Naszym pierwszym pomysłem było **załączenie 4 środkowych głośników z dołu i z góry, tak aby uzyskać punkt skupienia pośrodku lewitatora.**

W tym celu podaliśmy sygnał prostokątny 5 V z generatora na sterowniki tc4427, które wzmacniały sygnał i podawały go na odpowiednie głośniki. **Odległość między głośnikami ustawiliśmy doświadczalnie.** Przy odległościach zbliżających się do wielokrotności długości fali obserwowałyśmy znaczące drgania unoszonego styropianu. Przy innych odległościach z kolei nie obserwowałyśmy żadnych właściwości lewitacyjnych.

Po paru godzinach testowania dobraliśmy odległość równą ok. 51.5 mm. Jest to całkowita wielokrotność długość fali, ponieważ $51.45/8,575 = 6$. 51.5 mm to wysokość lewitatora, a 8,575 mm to długość emitowanej fali. Wynik równy 6 to całkowity stosunek wielokrotności długości fali (warunek wzmacnienia interferencyjnego).



Rys. 32 Testowanie działania lewitatora i aplikowanie styropianu na odpowiednie położenia
Źródło: zdjęcia własne

Użyty przez nas obiekt lewitacji to styropianowe kulki. Sygnały interferowały i obiekty lewitowały w osi równe oddalonej od każdego głośnika, na całej ich długości. Było to możliwe, ponieważ fale ze względu na małą odległość głośników względem siebie były prawie równoległe. Chociaż efekt wyglądał widowiskowo, to był łatwiejszy do wykonania niż lewitacja w jednym punkcie.

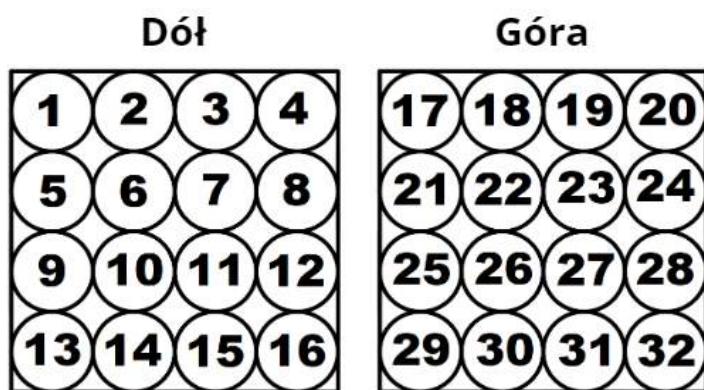
II.1.5.1 Spalenie FPGA

W trakcie naszych prób zauważaliśmy, że lewitacja jest znacznie uzależniona od odległości między czołami głośników, a fala z każdym centymetrem znacznie słabnie. Wiedzieliśmy, że chcąc zwiększyć odległość i ilość unoszonych obiektów musimy zwiększyć amplitudę fali. Ze względu na to, że jest ona proporcjonalna do napięcia, postanowiliśmy zmienić zasilacz. Pierwotny był ograniczony do podawania maksymalnie 12V, więc zdecydowaliśmy się na zasilacz laboratoryjny z regulowanym napięciem. Po ustawieniu go na 14 V oraz podłączeniu do zasilania zobaczyliśmy chmurę dymu, więc natychmiast wyłączyliśmy wszystko z zasilania. Zastosowany przez nas zasilacz podał 2 A spalając układy tc4427 i finalnie niszcząc FPGA. W efekcie analiz okazało się, że napięcie o wartości 14 V cofnęło się na FPGA, za pośrednictwem niesprawnego układu TC4427. Błądem z naszej strony okazało się niewykorzystanie ograniczenia natężenia na zasilaczu, który miał taką możliwość. Po sprawdzaniu i wymianie spalonej układowej problem niezwłocznie naprawiliśmy.

II.1.6. Tworzenie programu do lewitacji

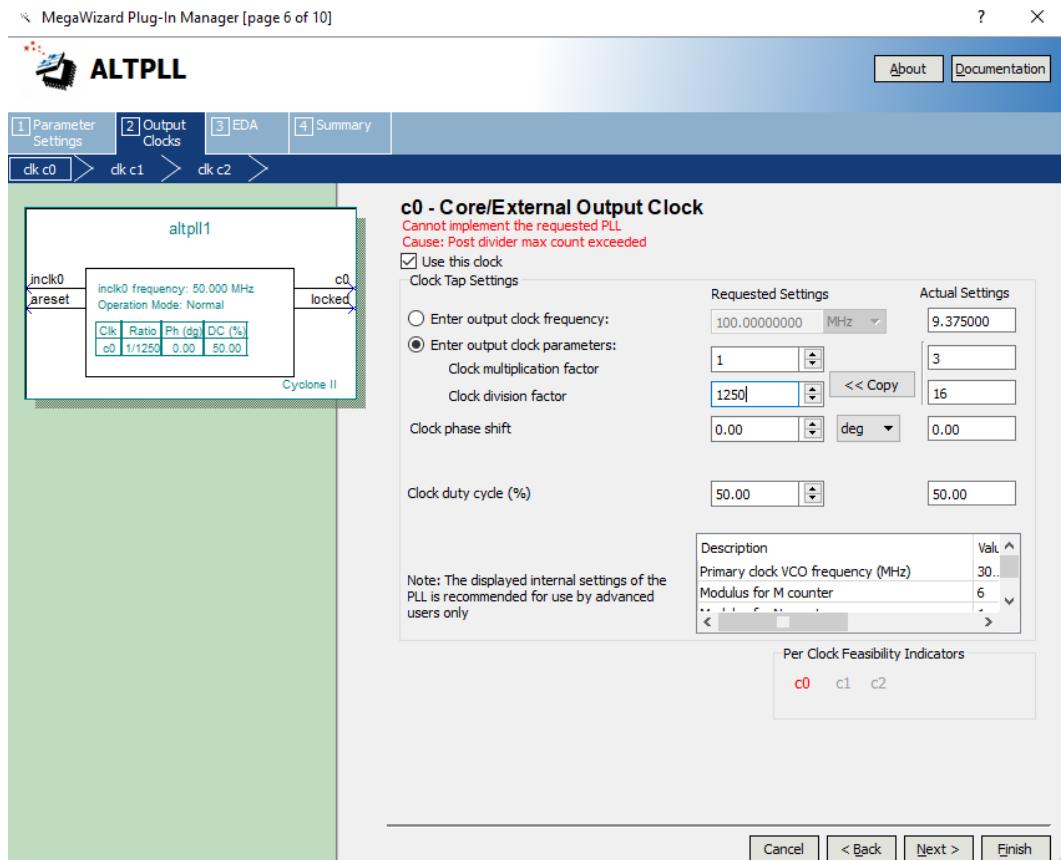
II.1.6.1. Modyfikacja układu

Po przetestowaniu lewitatora przy użyciu generatora i pierwszych pozytywnych efektach, przeszliśmy do projektowania programu w Quartusie. Celem było podanie sygnału 40 kHz na głośniki w takim samym ustawieniu jak dotychczas (głośniki środkowe zarówno z dołu jak i góry). Przelutowaliśmy więc układ oraz dodaliśmy konektory na płytę drukowaną, a ich wyprowadzenia podłączyliśmy do mikrokontrolera FPGA. Wyjścia z każdego głośnika przechodzące przez sterowniki tc4427 podpięliśmy do FPGA. Odległość między płytami zwiększyliśmy do 64 mm. Głośniki zostały przez nas ponumerowane w sposób zaprezentowany na Rys. 33.



Rys. 33 Numeracja głośników
Źródło: opracowanie własne

II.1.6.2 Programowe uzyskanie częstotliwości

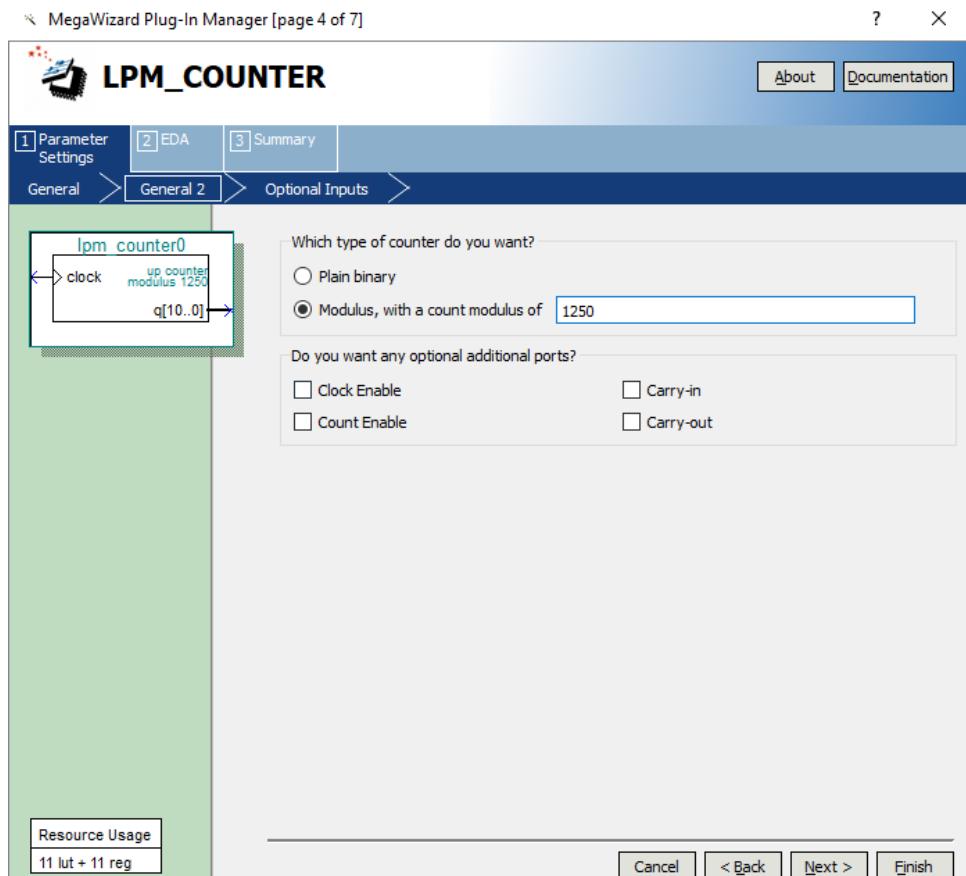


Rys. 34 Modyfikacja parametrów zegara ALTPPLL

Źródło: zdjęcie własne

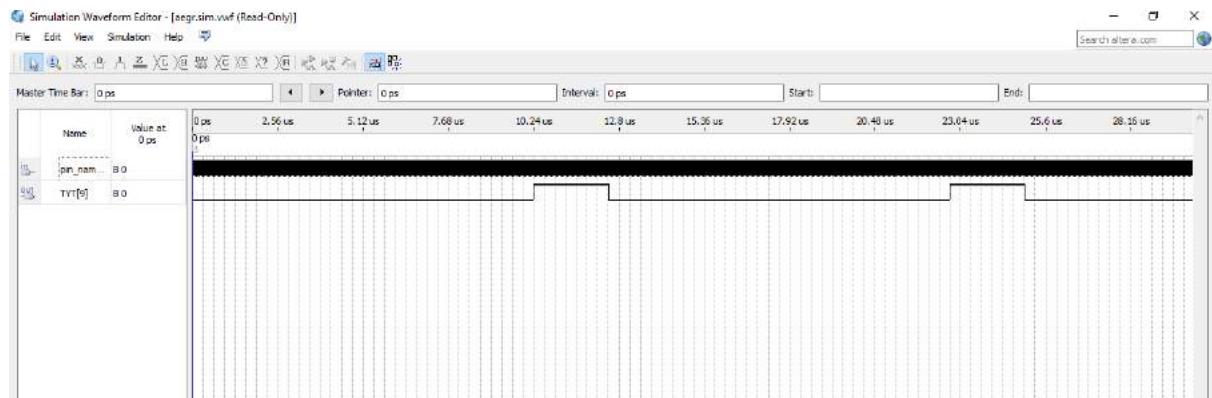
Pierwszym podejściem było **zastosowanie zegara ALTPPLL** (patrz: Rys. 34), który pozwala na zmianę częstotliwości wyjściowej wraz z opóźnieniem czasowym sygnału i jego wypełnieniem.

Niestety jak się okazało, zegar ten posiada wiele ograniczeń. Głównym jest zakres częstotliwości wyjściowej VCO, a także zakres częstotliwości PFD. Jest on obsługiwany przez detektor fazy i częstotliwości, który steruje napięciem VCO.



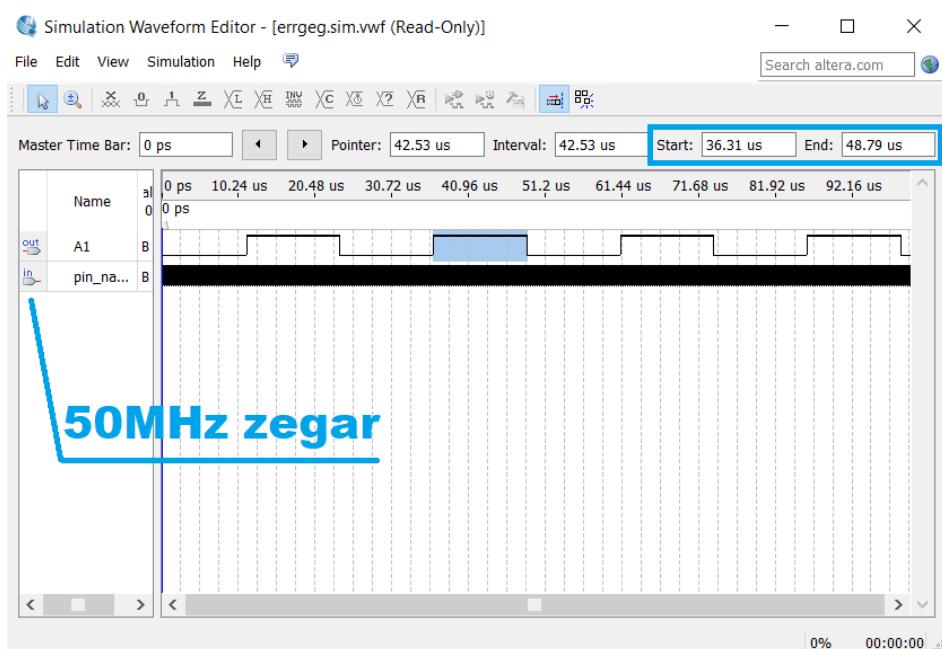
Rys. 35 Modyfikacja parametrów *lpm_counter*
Źródło: zdjęcie własne

W drugim podejściu wykorzystaliśmy dostępną w bibliotekach megafunkcję **lpm_counter** (patrz: Rys. 35), aby uzyskać pożdaną częstotliwość. Blok ten posiada parametr **modulus**, gdzie po wprowadzeniu wartości decimalnej, na wyjściu otrzymuje się częstotliwość podzieloną przez tą wartość. W naszym przypadku **modulus miał wynosić 1250, gdyż $50 \text{ MHz}/1250 = 40 \text{ kHz}$** .



Rys. 36 Symulacja przebiegu z zastosowaniem funkcji *lpm_counter*
Źródło: zdjęcie własne

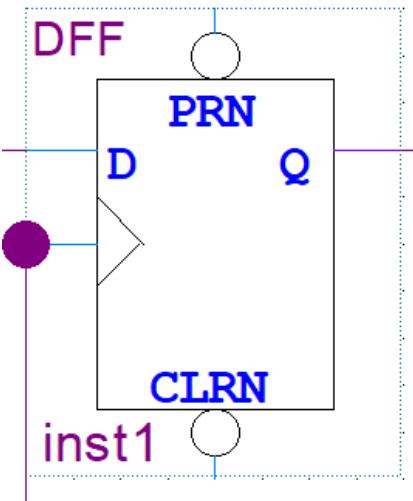
Otrzymany w ten sposób sygnał miał małe wypełnienie, podczas gdy pożądaną wartością było 50%. Aby uzyskać ten efekt zastosowaliśmy **przerzutnik JKFF**. Najpierw w counterze wartość modulus ustaliliśmy na 625 (ponieważ przerzutnik dzieli częstotliwości przez 2): $50 \text{ MHz}/625 = 80 \text{ kHz}$. Po przetestowaniu symulacji uzyskaliśmy na wyjściu **40 kHz z wypełnieniem 50%**. Czas pełnego okresu 40 kHz po przeliczeniu na mikrosekundy wynosi 25 μs . Ze wzoru na okres $T = 1/f$, z tego $T = 1/40000 = 25 * 10^{-6} = 25 \mu\text{s}$.



Rys. 37 Symulacja przebiegu o częstotliwości 40 kHz i wypełnieniu 50%
Źródło: opracowanie własne

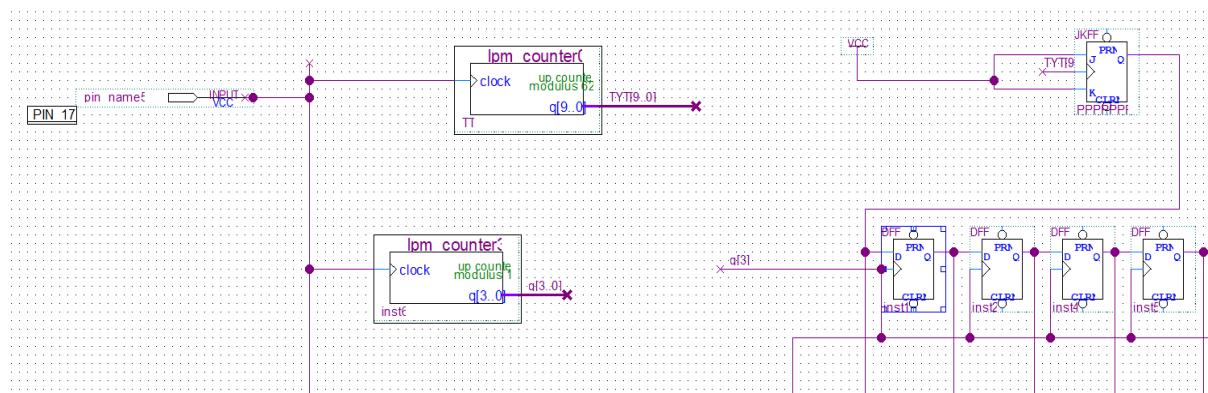
II.1.6.3. Programowa realizacja przesunięć fazowych

Następnym wyzwaniem było znalezienie sposobu na **programową realizację przesunięć fazowych między głośnikami** w celu możliwości sterowania lewitacją. Z pomocą przyszedł nam nasz opiekun mgr. inż Leszek Szpila który zaproponował użycie **przerzutników typu D**. Przerzutnik ten posiada jedno wejście danych, jedno wejście zegarowe i w zależności od rodzaju 1 lub 2 wyjścia komplementarne Q i $\neg Q$ (proste i zanegowane).



Rys. 38 Przerzutnik D
Źródło: opracowanie własne

Wiedząc to przeszliśmy do implementacji rozwiązania w programie. Początkowo nie byliśmy pewni jakiej liczby przerzutników D użyć, wszystko bazowało na rozdzielcości jaką chcieliśmy uzyskać. Z początku użyliśmy 50 przerzutników, z których każdy podawał sygnał przesunięty o $0,5 \mu\text{s}$ względem poprzedniego. Po namyśle zwiększyliśmy liczbę przerzutników do 125 w celu **zwiększenia dokładności przesunięć fazowych**. Tym razem przesunięcie na wyjściu każdego z przerzutników było równe $0,2 \mu\text{s}$, gdyż $25 \mu\text{s}/125 = 0,2 \mu\text{s}$. Ważna była również częstotliwość podawana na wejście zegarowe przerzutnika. Zegar 50 MHz z pinu 17 podaliśmy na licznik o parametrze modulus 10, uzyskując na wyjściu częstotliwość równą 5 MHz . Dzieląc ją przez liczbę przerzutników otrzymujemy $5 \text{ MHz}/125 = 40 \text{ kHz}$.



Rys. 39 Programowa realizacja przesunięć fazowych
Źródło: opracowanie własne

Następnym krokiem były **multipleksery**. Element taki mógł mieć dowolną ilość wejść do których mogliśmy wpisać "na sztywno" wymagane przesunięcia. Znaczco ułatwiało to **detekcję błędów** oraz skróciło czas pisania programu. Wybierały one wyjście

z przerzutnika D (oznaczone od 0 do 125, gdzie 0 to zerowe opóźnienie sygnału) zgodnie z wartością przypisaną na kanał sel (kanał wyboru). Aby otrzymać punkt lewitacji w dowolnym miejscu, musieliśmy obliczyć najpierw opóźnienia i później wprowadzić je do multiplekserów. Do obliczeń przyjęliśmy punkt zerowy znajdujący się na lewej dolnej krawędzi podstawki drukowanej. Aby ułatwić proces obliczeń skorzystaliśmy z arkuszu kalkulacyjnego.

Każdy z głośników ma określona odległość od punktu zerowego, a odległość od głośników liczymy umownie od środka głośnika. Aby znaleźć szukaną odległość liczymy wektor różnicę 3D między współrzędnymi obiektu lewitowanego a współrzędnymi głośnika. Następnie bierzemy wysokość obiektu od czoła głośników i odejmujemy ją od uprzednio obliczonej odległości. Tak uzyskany wynik zamieniamy na przesunięcie fazowe. Korzystając z prostej zależności $25 \mu\text{s} = 8,575 \text{ mm}$ (okres 25 μs , przy $v = 343 \text{ m/s}$, odpowiada długości fali 8,575 mm), obliczamy przesunięcie fazowe dla głośników w μs . Mnożymy wcześniej uzyskaną różnicę przez 100 i następnie dzielimy przez 34,3. Ponieważ niektóre z uzyskanych wyników są większe od 25 μs (a sygnałem jest funkcja okresowa) używamy funkcji MOD (modulus), przyjmującej 2 parametry: dzielną i dzielnik. Zwraca ona resztę z dzielenia liczby przez dzielnik. Następnie używamy funkcji część całkowita z dzielenia i dzielimy otrzymany wcześniej wynik przez 0,2. Otrzymana wartość jest umownym numerem kanału z jakiego sygnał jest podawany na głośnik.

$$\left[\left(\sqrt{(x - x_s)^2 + (y - y_s)^2 + H^2} - H \right) * \frac{100}{34.3} \right] \% 125 = N_s$$

gdzie:

- x, y - współrzędne lewitacji,
- x_s, y_s - odległość od punktu zerowego środka głośnika,
- H - wysokość lewitacji,
- N_s - numer kanału z przesunięciem.

Wartość 0,2 wynika z zastosowanym w naszym programie rozwiązaniu polegającym na użyciu przerzutników **DFF (Data flip-flop)**. Opóźniają one podawany sygnał proporcjonalnie do ilości przerzutników, których w naszym przypadku zastosowaliśmy 125. W związku z tym przy okresie 25 μs otrzymujemy: $25 \mu\text{s}/125 = 0,2$.

Niezwyczajne było dla nas również zwiększenie dokładności otrzymanego wyniku. Wynik funkcji całkowitej z dzielenia mnożymy więc razy otrzymane 0,2 i całe wyrażenie odejmujemy od czasu w μs uzyskanego w funkcji MOD. Następnie sprawdzamy czy otrzymany wynik dzieli się przez 0,1. Jeśli tak, do wartości kanału dodajemy 1, jeśli nie liczbę kanału zostawiamy bez zmian. Następnie do każdego

wyniku dodajemy jeszcze raz 1. Wynika to z tego, że kanał 0 to kanał bez przesunięcia fazowego, a licząc od zera mamy 126 kanałów, a tylko 125 przerzutników DFF.

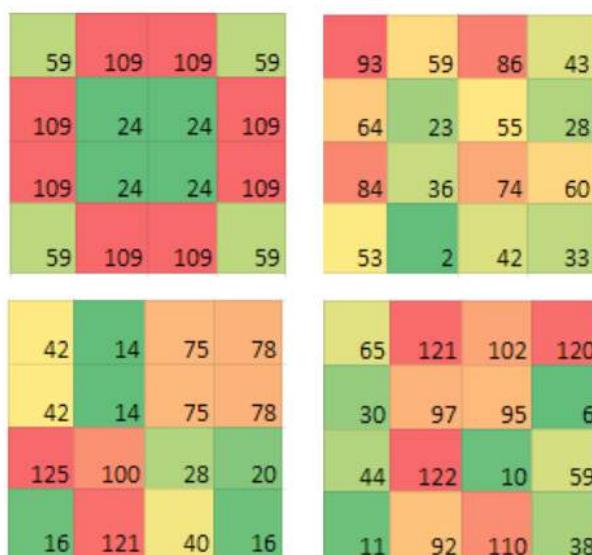
Opisany powyżej proces przedstawia program stworzony przez nas w arkuszu kalkulacyjnym (patrz: Rys. 40). Oblicza on wyniki automatycznie, na podstawie danych podanych w białych polach X i Y oraz wysokość dla pożądanych wartości.

X	Y	Wysokość													
POŁOŻENIE		WARTOŚCI													
X	Y	Nr	Wysokość	x	y	KWADRAT-X	KWADRAT-Y	DŁUGOŚĆ	RÓŻNICA	CZAS US	CZAS USS	ILORAZ	SPRAWDZENIE	DODAJ +	KANAŁ
32	32	1	29,5	8	8	576	576	44,9694	15,4694	45,4983	20,49834	102	0,09833548	0	103
32	32	2	29,5	24	8	64	576	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	3	29,5	40	8	64	576	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	4	29,5	56	8	576	576	44,9694	15,4694	45,4983	20,49834	102	0,09833548	0	103
32	32	5	29,5	8	24	576	64	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	6	29,5	24	24	64	64	31,5951	2,09509	6,16204	6,162043	30	0,16204281	1	32
32	32	7	29,5	40	24	64	64	31,5951	2,09509	6,16204	6,162043	30	0,16204281	1	32
32	32	8	29,5	56	24	576	64	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	9	29,5	8	40	576	64	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	10	29,5	24	40	64	64	31,5951	2,09509	6,16204	6,162043	30	0,16204281	1	32
32	32	11	29,5	40	40	64	64	31,5951	2,09509	6,16204	6,162043	30	0,16204281	1	32
32	32	12	29,5	56	40	576	64	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	13	29,5	8	56	576	576	44,9694	15,4694	45,4983	20,49834	102	0,09833548	0	103
32	32	14	29,5	24	56	64	576	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	15	29,5	40	56	64	576	38,8619	9,36194	27,5351	2,535103	12	0,13510325	1	14
32	32	16	29,5	56	56	576	576	44,9694	15,4694	45,4983	20,49834	102	0,09833548	0	103

Rys. 40 Obliczanie przesunięć fazowych za pomocą programu Excel

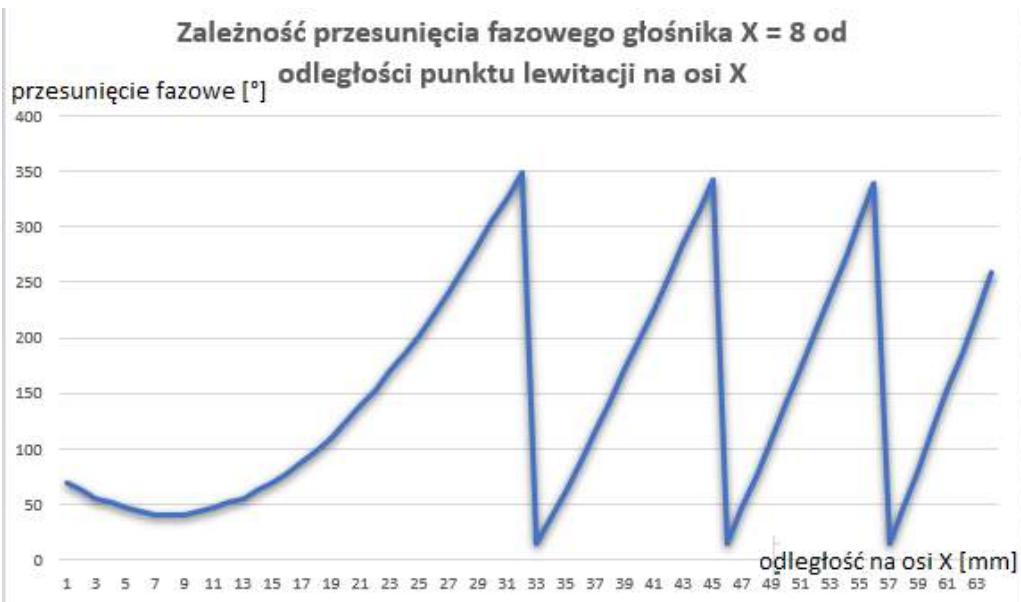
Źródło: zdjęcie własne

Do programu powstała wizualizacja pokazana na Rys. 41. **Każdy kolor obrazuje inne przesunięcia na głośnikach**, ten sam kolor na kilku głośnikach oznacza więc to samo przesunięcie przez nie wysyłane. Tabela automatycznie zmienia kolor po wpisaniu położenia X, Y i wysokość (Z) w tabelce we wspomnianym wcześniej arkuszu.

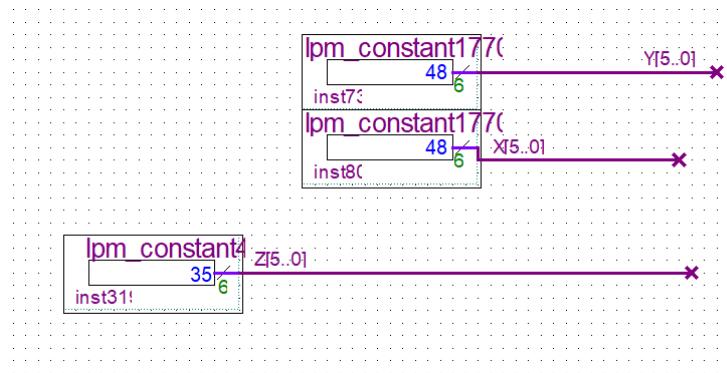


Rys. 41 Wizualizacja przesunięć fazowych dla różnych wartości współrzędnych X, Y, Z

Źródło: zdjęcie własne



Rys. 42 Wykres opóźnienia fazowego
Źródło: opracowanie własne



Rys. 43 współrzędne x, y, z punktu skupienia
Źródło: zdjęcie własne

Po wpisaniu wartości opóźnień do multiplekserów a następnie wgrywając program do mikrokontrolera, otrzymujemy punkt skupienia dokładnie w pożądanym miejscu.



Rys. 44 Otrzymywanie punktu skupienia w dowolnym miejscu
Źródło: opracowanie własne

II.1.6.4. Problem rozproszenia fali akustycznej

Podczas pracy nad lewitacją zaobserwowałyśmy, że zjawisko lewitacji charakteryzuje się sporą **delikatnością**. W obecnym układzie głośników w podstawkach nie były one jednolicie wypoziomowane, dlatego też głośniki obróciliśmy i ponownie złutowaliśmy. Po wielu godzinach testów jednak nie byliśmy w stanie poddać lewitacji żadnych częstek. Okazało się, że **obszary niewypełnione między głośnikami powodowały rozproszenie fali** i przez to niemożliwość powstania fali stojącej. Głośniki wypełniliśmy gorącym klejem tak aby zapełnić te obszary. Lewitator znów zaczął działać.

Potwierdziliśmy tym samym, że **fala akustyczna odbijając się od nierównych powierzchni tworzy niepożądane punkty skupienia i może zaburzyć właściwości lewitacyjne układu.**



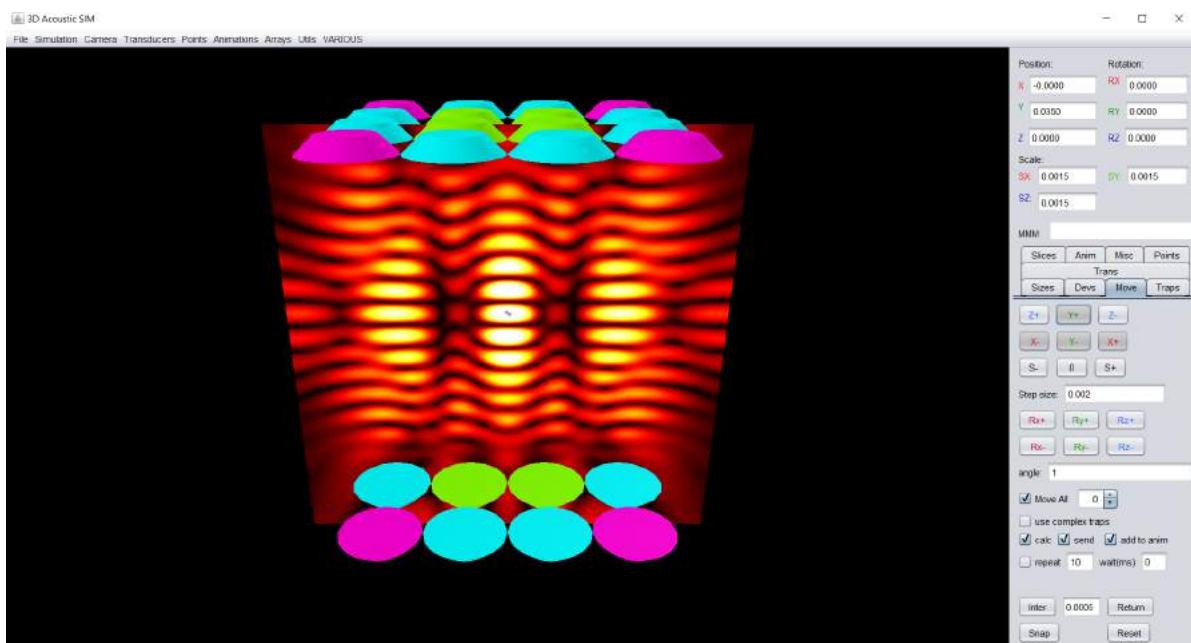
Rys. 45 Gorący klej między głośnikami
Źródło: zdjęcie własne

II.1.7. Praca z symulatorem

W trakcie szukania alternatywnych rozwiązań obliczania przesunięć fazowych natknęliśmy się na program, który przykuł naszą uwagę. Była to aplikacja stworzona na Uniwersytecie Bristolskim, pozwalająca na symulowanie lewitacji akustycznej. Aby ją użyć musieliśmy zainstalować program **Apache Netbeans** oraz bibliotekę **Java JDK 11**. Następnie udostępniony kod źródłowy wprowadziliśmy do programu Netbeans i uruchomiliśmy go.

Zaznajamiając się z programem odkryliśmy wiele fascynujących funkcji. W programie można stworzyć swój własny model lewitacyjny, ze swobodnym określeniem ilości głośników z każdej strony, odległości między nimi, czy też kształt ich układu (heksagonalny, kolisty).

Rys. 46 przedstawia jak po przeniesieniu wyglądał nasz układ do symulatora.

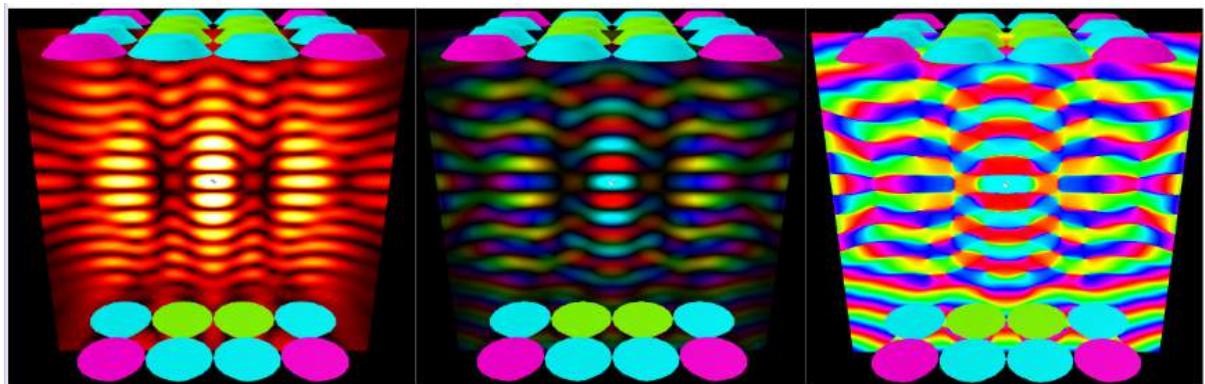


Rys. 46 Prototyp urządzenia w symulacji
Źródło: zdjęcie własne

Parametr częstotliwości głośników ustaliliśmy na 40 kHz, a ich średnicę na 16 mm. Kolor każdego z głośników jest ustalany według następujących reguł:

- ten sam kolor oznacza takie samo przesunięcie sygnału wysyłanego z głośników;
- różny kolor to różne opóźnienia fazowe wysyłane z głośników.

Zakładka slices umożliwiła nam dodanie przekroju lewitatora z **wizualizacją zagęszczeń i rozrzedzeń powietrza** w nim powstających. Dostępne są trzy opcje:

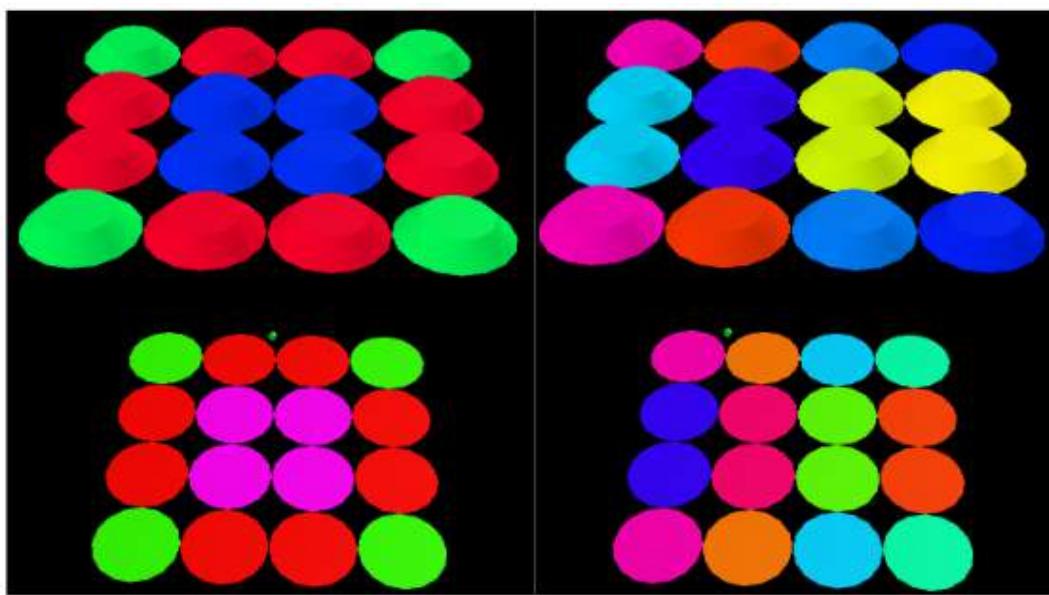


Rys. 47 Wizualizacja pola ciśnienia akustycznego w lewitorze
Źródło: zdjęcia własne

Opcje te pokazują następująco: amplitudę, opóźnienie fazowe oraz połączenie amplitudy i opóźnienia fazowego. W zakładce opisanej points dodawaliśmy obiekt do symulacji. Po jego dodaniu, w zakładce move znajdował się panel sterowania cząsteczką opisany za pomocą współrzędnych x, y, z. Aby nią poruszać trzeba ją wybrać tak, aby jej kolor zmienił się na zielony a następnie wybrać odpowiednie współrzędne na panelu sterowania lub też ręcznie wprowadzić w ich pola wartości. Okienko o nazwie Step Size to dystans w metrach o jaki w każdym kroku ma poruszać się obiekt. W naszym przypadku ustaliliśmy go na 0,002 czyli 2 mm.

W zakładce trans będącej skrótem od angielskiego transducerów czyli po polsku głośników, widzimy szereg parametrów. Tym który w głównej mierze nas interesuje to **parametr PH** czyli **opóźnienie fazowe wyrażone w rad/π**. Naciskając na głośnik możemy w każdej chwili podejrzeć jego opóźnienie. Parametr ten przyjmuje wartości od 0 do 2, gdzie zero oznacza zerowe przesunięcie, natomiast 2 - maksymalne. Oprócz tego istnieje możliwość numeracji każdego z głośników, a także przypisania im odpowiednich pinów wyjściowych.

Przy procesie poruszania obiektem w symulacji potwierdziła się poprawność naszego wcześniejszego rozumowania. Podczas ruchu góra - dół na środku, głośniki grupowały się w sekcje, które wysyłyły sygnał z tym samym opóźnieniem. Tak samo w ruchu horyzontalnym, głośniki łączyły się w pary zarówno na dole jak i na górze i wysyłając sygnał z tym samym opóźnieniem (Rys. 48).



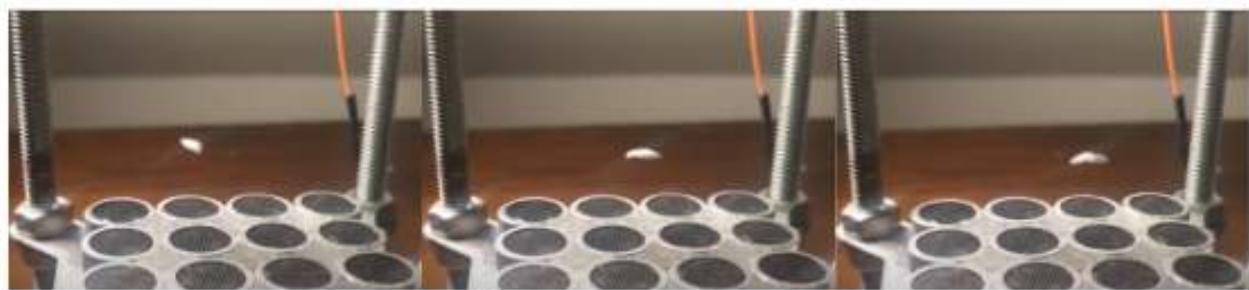
Rys. 48 Obraz zmiany faz wysyłanych przez głośniki

Źródło: opracowanie własne

Parametr PH, wspomniany wcześniej w opisie opcji, wprowadziliśmy ręcznie do programu opracowanego w Quartusie, który oscylował między początkową a końcową wartością. Jak zauważyliśmy efekt nie różnił się znaczco od tego uzyskanego naszym własnym podejściem. Lewitujący obiekt był stabilny, nie było również problemów z lewitacją jego większej ilości.

II.1.8. Testowanie działania

Wykonując odpowiednie obliczenia, zaczeliśmy **programować ruchy po obranych torach ruchu w osi poziomej**. Praca ta była wymagającej, gdyż wymagała od nas precyzji ustawienia obiektu lewitującego w położeniu początkowym i odpowiednim zgraniu programu. Efekty jednak były niesamowite.



Rys. 49 Poruszanie obiektem lewitującym w osi poziomej

Źródło: zdjęcia własne

Uzyskaliśmy precyzyjne odzwierciedlenie zadanego ruchu. Cząsteczka oscylowała między położeniem początkowym i końcowym. W czasie testów sprawdzaliśmy **najdłuższy czas lewitacji obiektu.** Był on równy ok. kilku minut, aż do powstanie zakłóceń (czy to otoczenia czy samego sygnału), po czym cząstka spadała.

II.1.8.1. Lewitacja kilku cząsteczek na raz



Rys. 50 Lewitacją większej ilości obiektów w formie trójkąta
Źródło: zdjęcie własne

Obliczając wypadkowe interferencje fali dźwiękowej mogliśmy precyzyjnie określić miejsca skupienia fali. To pozwoliło nam na umieszczeniu kilku rozproszonych obiektów w tym samym czasie i ich lewitację po określonym torze ruchu. Lewitacja ta była bardziej niestabilna niż w przypadku tej z pojedynczym obiektem, lecz efekt był naprawdę satysfakcjonujący.



Rys. 51 Poruszanie kilkoma obiekty na raz
Źródło: zdjęcia własne

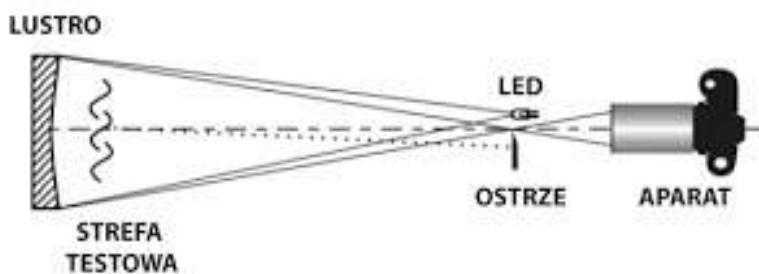
II.1.9. Obrazowanie Schlierena

Proces uzyskiwania lewitacji był bardzo czasochłonny i detekcja ewentualnych błędów w naszym rozumowaniu była ciężka. Ze względu na to, że fali akustycznej nie można zobaczyć, ciężko ustalić istnienie punktu zdolnego do lewitacji. Aby efektywniej wyznaczyć miejsce skupienia oraz dokonać odpowiedniej synchronizacji programu postanowiliśmy więc poszukać jak najlepszej metody zobrazowania fali stojącej. Dzięki temu natrafiliśmy na ciekawą technikę zwaną obrazowaniem smugowym czy też obrazowaniem Schlierena.

Jest to technika optyczna, która pozwala dostrzec niewielkie zmiany współczynnika załamania powietrza a także innych przezroczystych ośrodków. Załamanie to spowodowane niejednorodnością powietrza jest widoczne dzięki układowi optycznemu z pojedynczym parabolicznym lub sferycznym zwierciadłem. Ta niejednorodność może być spowodowana zmianami gęstości, temperatury lub ciśnienia powietrza. Za pomocą kamery z podglądem cyfrowym możemy zobaczyć ciepłe ruchy konwekcyjne unoszące się np. z płomienia świecy lub gorącej herbaty, czy też zimne powietrze opadające np. z suchego lodu. Za pomocą tej techniki można również wizualizować przepływ gazów innych niż powietrze.

Naszym celem było **zobrazowanie fali stojącej powstającej w lewitatorze**. Aby stworzyć nasz własny układ obrazowania smugowego potrzebowaliśmy:

- zwierciadła sferycznego lub parabolicznego,
- ostrej żyłetki,
- kamery z podglądem cyfrowym,
- punktowego źródła światła.



Rys. 52 Układ sprzętu do obrazowania Schlierena
Źródło: kopernik.org.pl

Ważne jest aby źródło światła było punktowe, by ograniczyć szerokość wiązki światła, a tym samym kąt w którym są one nadawane. Ostrze znajdujące się w ogniskowej reguluje ilość światła wpadającego do aparatu.



Rys. 53 Stanowisko do obrazowania Schlierena
Źródło: zdjęcia własne

Gdy poznaliśmy teorię tego zjawiska rozpoczęliśmy ustawianie całego układu. Z ogólnodostępnych informacji dowiedzieliśmy się że do najlepszego efektu potrzeba **znacznej precyzji**. Dobre ustawienie zwierciadła, lampki LED i żyletki może zająć nawet cały dzień.

Aby uniknąć uszkodzenia zwierciadła złożyliśmy podstawkę, w którą umieściliśmy przyrząd.. Z tyłu zamontowany został gwint do regulacji nachylenia. Z pierwszego ułożenia zauważylismy, że układ nie może być ułożony na drewnianej podłodze. Chociaż może wydawać się solidna, to podczas chodzenia występują mikro odkształcenia, które uniemożliwiają precyzyjne umieszczenie promienia na żyletce.

Pierwsze próby przeprowadziliśmy sprawdzając wizualizację ciepła emitowanego z powierzchni ręki czy też gazu wydobywającego się z zapalniczki (Rys. 54).



Rys. 54 Widoczność gazu ulatniającego się z zapalniczki
Źródło: zdjęcia własne

Efekty obrazowania smugowego były dla nas zaskakujące - dojrzaliśmy to co niewidoczne. Jednak zobrazowanie ciepła z powierzchni ręki czy gazu z zapalniczki nie było naszym zasadniczym celem. Było nim **zobrazowanie fali stojącej**.

Układ ze zwierciadłem, ostrzem i kamerą zmodyfikowaliśmy więc dodając nasze urządzenie w odległości kilku centymetrów od zwierciadła. Po obserwacji i próbie zmiany jego odległości nie byliśmy jednak w stanie zaobserwować żadnych zmian w podglądzie cyfrowym kamery.



Rys. 55 Brak widoczności fal akustycznych
Źródło: zdjęcia własne

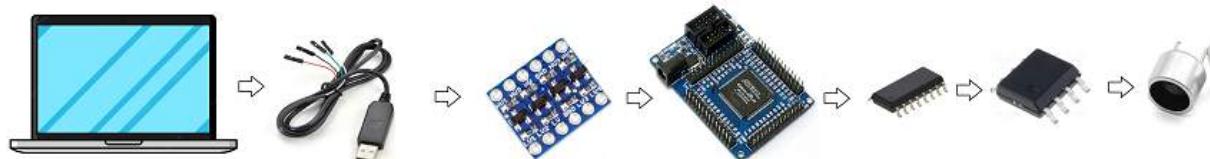
Obrazowanie smugowe okazało się metodą, która nie sprostała naszym potrzebom. Zmiany ciśnienia w powietrzu wywołane przez interferujące fale dźwiękowe załamywały światło w zbyt małym stopniu, aby można było je dostrzec. Ogniskowa zwierciadła o wielkości 0,75 m była zbyt mała, aby móc zobaczyć takie niewielkie odchylenia. W przestudiowanych przez nas badaniach obrazowania fal stojących stosowano zwierciadła o ogniskowej rzędu kilku metrów i znacznie większej średnicy niż nasze. Kosztują one dużo więcej i przekraczają znacząco nasz budżet. Co więcej, nawet one wymagają do zauważalnego efektu dobrego aparatu z cyfrowym podglądem.

Dokładne ustawienie zwierciadła jest w przypadku tego obrazowania kluczem, dlatego nie wykluczamy, że przy idealnym rozłożeniu i warunkach mogłoby się to udało. Postanowiliśmy jednak skupić się na udoskonalaniu projektu, a temat na ten moment odłożyliśmy na bok. Mamy nadzieję, że w przyszłości będzie szansa na dalszy rozwój w tym zakresie.

II.2 Finalna wersja prototypu FuturFlow

Projekt następnej wersji prototypu składał się z:

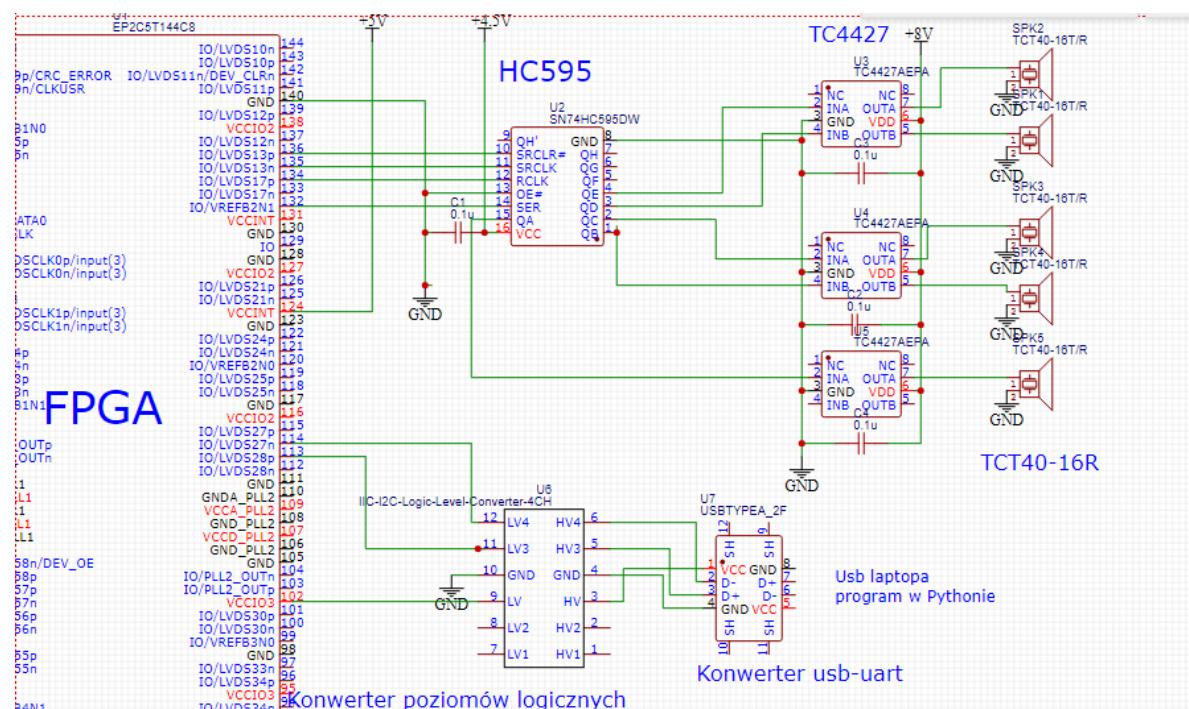
- 200 głośników ultradźwiękowych TCT40-16T,
- mikrokontrolera FPGA Altera Cyclone II EP2C5144TC8N,
- drukowanej obudowy,
- programu do lewitacji opracowanego w Pythonie,
- programu opracowanego w Quartusie,
- rejestrów przesuwnych HC595,
- driverów tranzystorów Mosfet TC4427aea,
- własnoręcznie zrobionej płytki PCB,
- przetwornic step-down,
- przetwornicy AC/DC - 230/12 V.



Rys. 56 Schemat graficzny ciągu komunikacyjnego

Źródło: opracowanie własne

II.2.1. Schemat elektryczny



Rys. 57 Schemat elektryczny układu wykonawczego

Źródło: opracowanie własne

Pracę zaczęliśmy od opracowania schematu ideowego budowy tej wersji urządzenia. Naszym celem było zaprojektowanie układu tak, aby lewitator składał się z 200 głośników, którym każdym z osobna będziemy w stanie sterować. Ze względu na tak dużą liczbę nie można było przypisać każdemu pinowi FPGA osobnego głośnika. Musieliszy więc z jednego wyjścia nadawać sygnał na kilka głośników w postaci zakodowanej. Ze względu na to potrzebne były układy, które potrafią taki sygnał odczytać i rozdzielić na poszczególne nadajniki. Do tego celu wybraliśmy rejestyry przesuwne 74HC595. Ich zaletą są optymalne czasy opóźnienia sygnału (które są ważne przy regulowaniu opóźnień fazowych) oraz posiadanie latches, niezbędnego, aby podczas przesuwania bitów wyjścia nie były aktywne. Mają one jednak pewne ograniczenia.

Po pierwsze, maksymalne prędkości jakie mogą przyjąć zależą od napięcia. FPGA nie jest w stanie podać więcej niż 3,3 V jako sygnał wysoki, co ogranicza możliwe napięcia zasilania rejestru, które w przypadku stanu wysokiego 3,3 V wynosi około 4,5 V.

AC ELECTRICAL CHARACTERISTICS($C_L=50\text{pF}$, Input $t_r=t_f=6.0\text{ ns}$)

Symbol	Parameter	V_{CC} V	Guaranteed Limit			Unit
			25 °C to -55°C	≤85 °C	≤125 °C	
f_{max}	Minimum Clock Frequency (50% Duty Cycle) (Figures 1 and 7)	2.0 4.5 6.0	6.0 30 35	4.8 24 28	4.0 20 24	MHz

Rys. 58 Dokumentacja układu HC595 - zależność częstotliwości taktowania od napięcia zasilania
Źródło: <https://www.alldatasheet.com/datasheet-pdf/pdf/46165/SLS/HC595.html>

Przy takim zasilaniu układy są w stanie przyjąć sygnał o maksymalnej częstotliwości 30 MHz. Ogranicza to więc szybkość komunikacji FPGA - 74HC595, a tym samym dokładność w ustalaniu przesunięć fazowych.

V_{OH}	Minimum High-Level Output Voltage, $Q_A = Q_H$	$V_{IN} = V_{IH}$ or V_{IL} $ I_{OUT} \leq 20\text{nA}$	2.0 4.5 6.0	1.9 4.4 5.9	1.9 4.4 5.9	1.9 4.4 5.9	V
----------	--	---	-------------------	-------------------	-------------------	-------------------	---

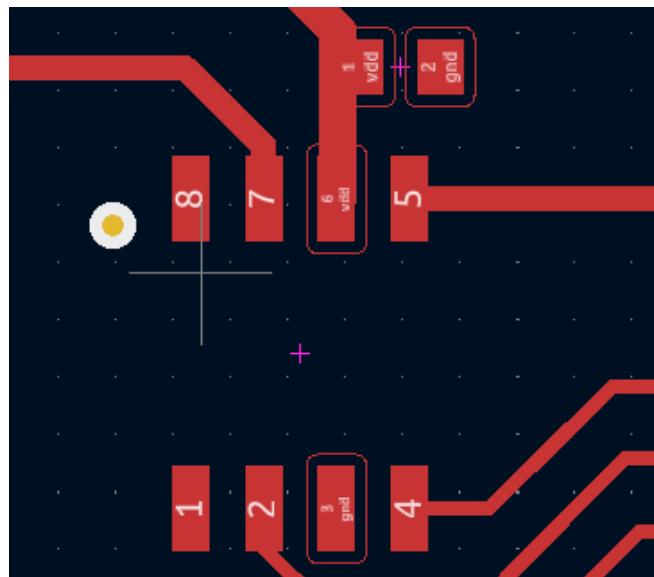
Rys. 59 Dokumentacja układu HC595 - zależność napięcia wyjściowego od napięcia zasilania
Źródło: <https://www.alldatasheet.com/datasheet-pdf/pdf/46165/SLS/HC595.html>

74HC595 zasilany z 4,5 V ma wyjściowy stan wysoki około 4,4 V, co jest wartością zdecydowanie za małą do optymalnej mocy głośników. Nie jest on również w stanie zapewnić odpowiedniego natężenia. Ze względu na to konieczne było użycie sprawdzonych już układów TC4427, które mogą operować bardziej odpowiednim napięciem - w naszym wypadku 8 V.

Output	V_{OH}	High Output Voltage	$V_{DD} - 0.025$	—	—	—	V
--------	----------	---------------------	------------------	---	---	---	---

Rys. 60 Dokumentacja układu TC4427 - zależność napięcia wyjściowego od napięcia zasilania

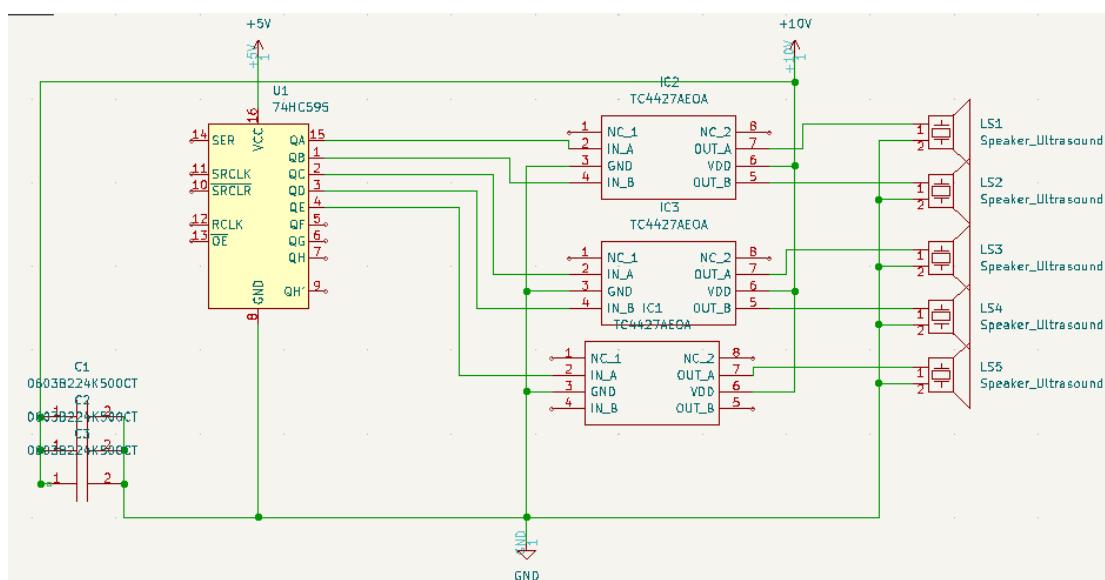
Źródło: alldatasheet.com/datasheet-pdf/view/26028/TELCOM/TC4427.html



Rys. 61 Pola lutownicze na kondensator blokujący bezpośrednio przy zasilaniu układu

Źródło: zdjęcie własne

W celu **wyeliminowania zakłóceń**, które w tym wypadku stanowiłyby problem, zastosowaliśmy kondensatory blokujące. Są one połączone równolegle z wyprowadzeniami każdego układu TC4427 i HC595, tak aby odseparować je od innych. Zasilanie dla głośników i układów wyniosło 8 V, dla 74HC595 4,5 V, a dla FPGA 5 V.

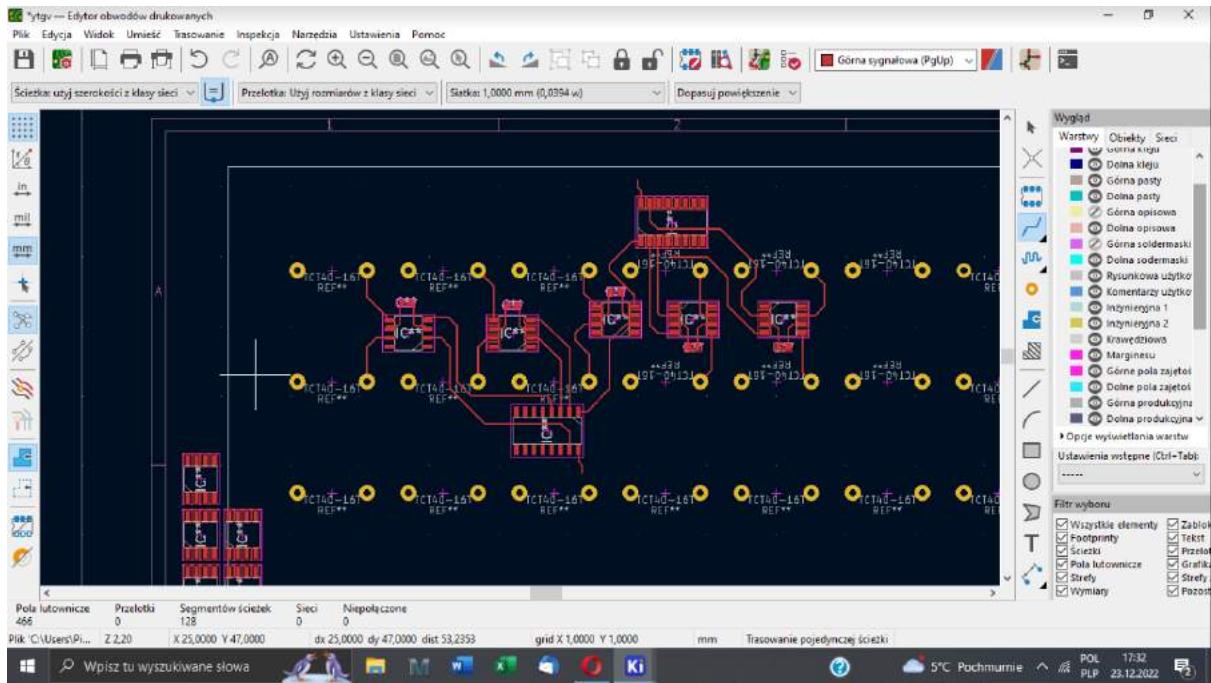


Rys. 62 Schemat elektryczny połączeń układów scalonych (bez FPGA)

Źródło: zdjęcie własne

II.2.2. Projekt płytka PCB

Nasza płytka PCB została wykonana w celu optymalizacji połączeń elektrycznych oraz zminimalizowania usterek, wynikających z przerw między połączeniami.

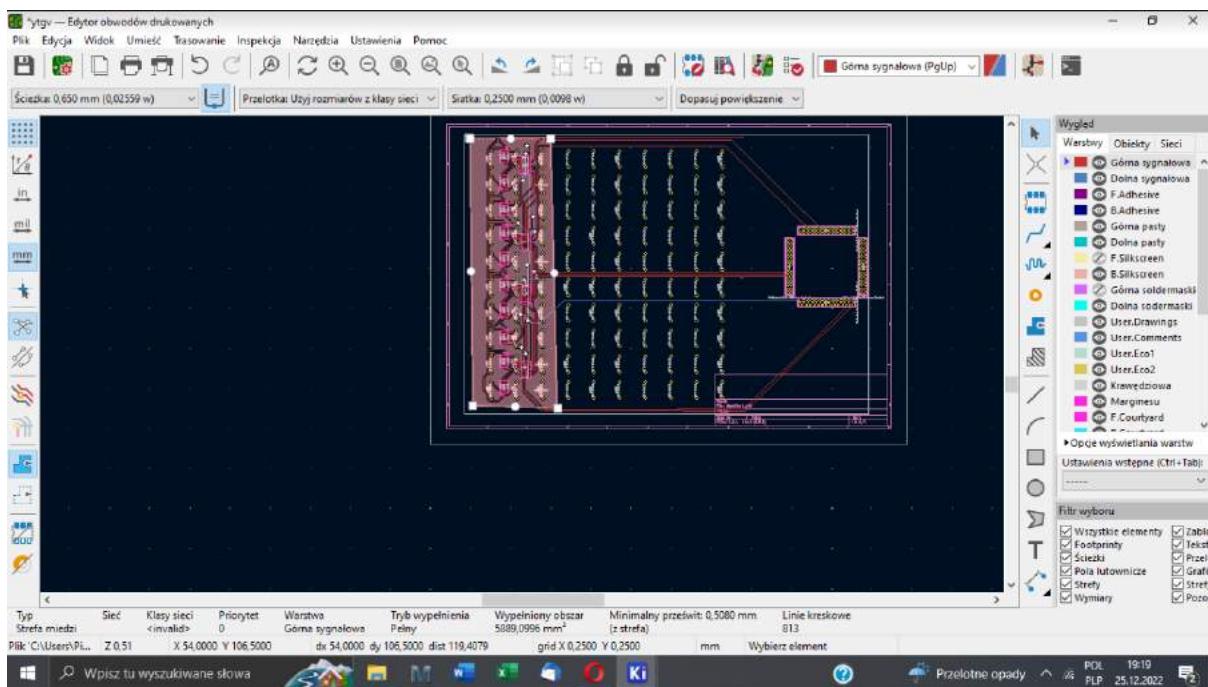


Rys. 63 Projektowanie połączeń na płytce

Źródło: zdjście własne

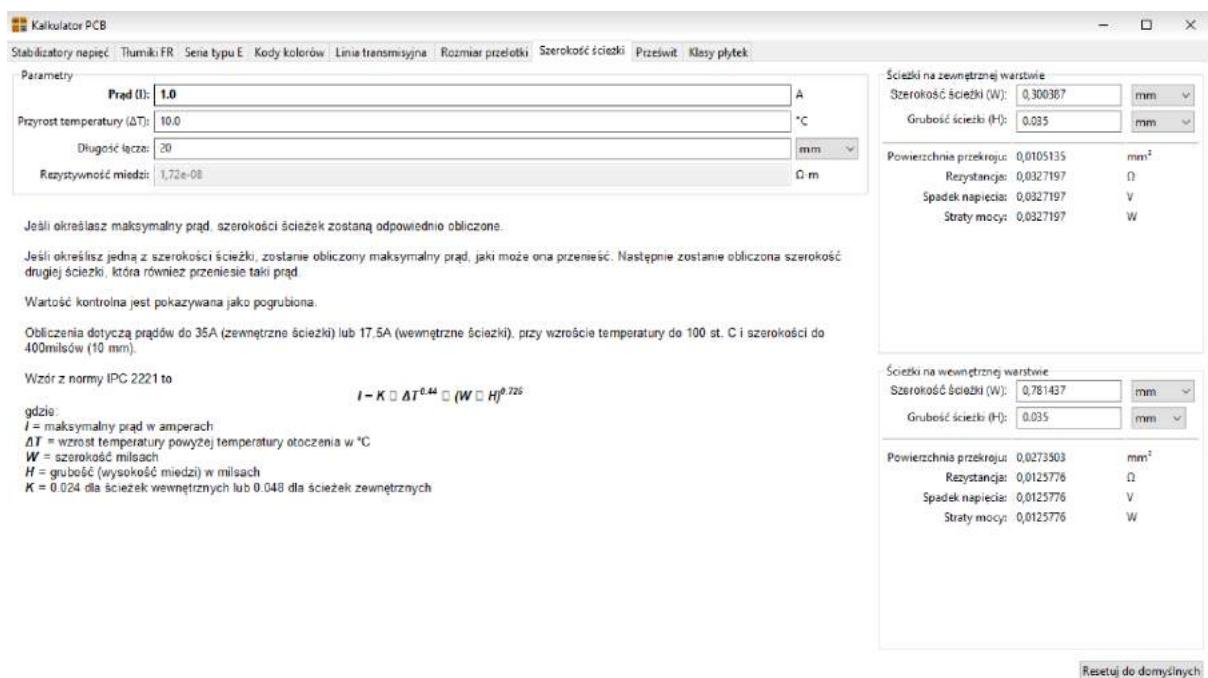
Było to jednak pewne wyzwanie, gdyż między otworami nie ma dużo miejsca. Rozłożenie wyprowadzeń wejściowych rejestru z jednej strony również nie ułatwia zadania. Z kolei połączenie HC595 z 5 kolejnymi głośnikami w kolumnie nie jest możliwe, wyprowadzenia zabierają za dużo miejsca i odcinają ścieżki od innych układów. Rejestry i sterowniki muszą znajdować się w różnych długościach miejsc wolnych. Biorąc to pod uwagę do jednego rejestru przypisaliśmy 2 rzędy głośników.

Gdy znaleźliśmy optymalny wzór, reszta została powielona. Do każdego rejestru została doprowadzona osobna ścieżka na kanał sel, a reset, latch i shift zostały połączone razem dla każdego rejestru przesuwnego.



Rys. 64 Tworzenie ścieżek na płytce oraz dodanie wyprowadzeń pod FPGA
Źródło: zdjęcie własne

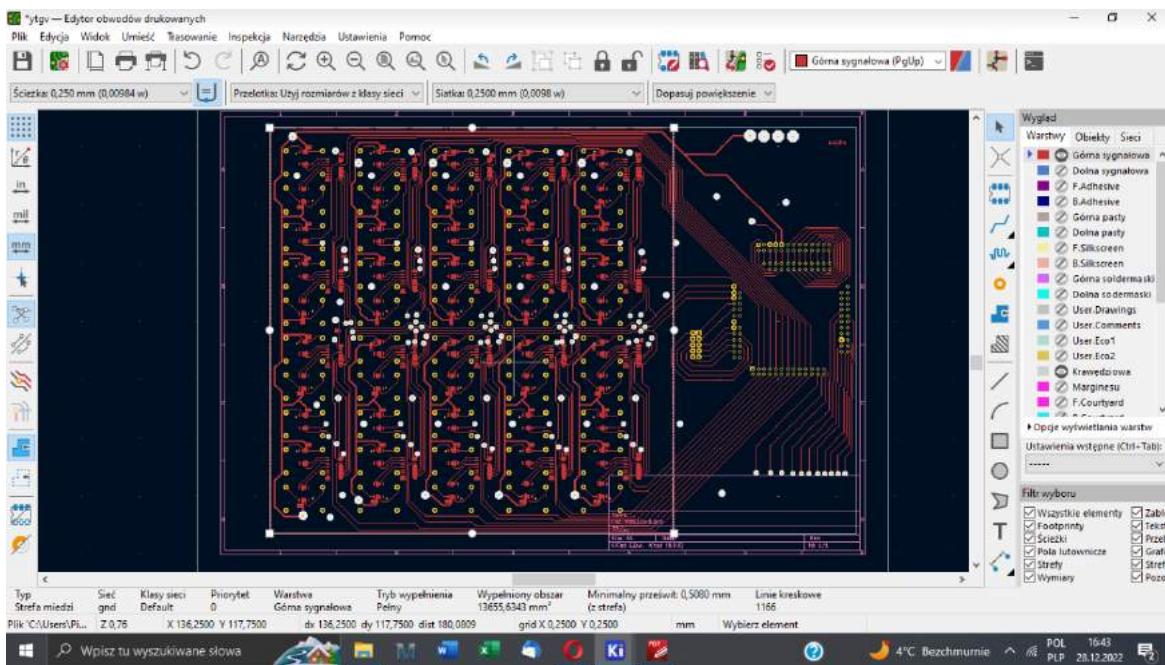
FPGA z założenia miało wchodzić na wcisk do płytka aby uniknąć zbędnego okablowania. Dlatego na płytce dodaliśmy gniazda goldpin.



Rys. 65 Kalkulator PCB - wykonywanie obliczeń w celu uzyskania optymalnej grubości ścieżek
Źródło: zdjęcie własne

Do obliczenia grubości linii zasilających użyliśmy wbudowanego w Kicad kalkulatora PCB (Rys. 65). Przy poprzedniej wersji 32 głośniki pobierały prąd przy wszystkich

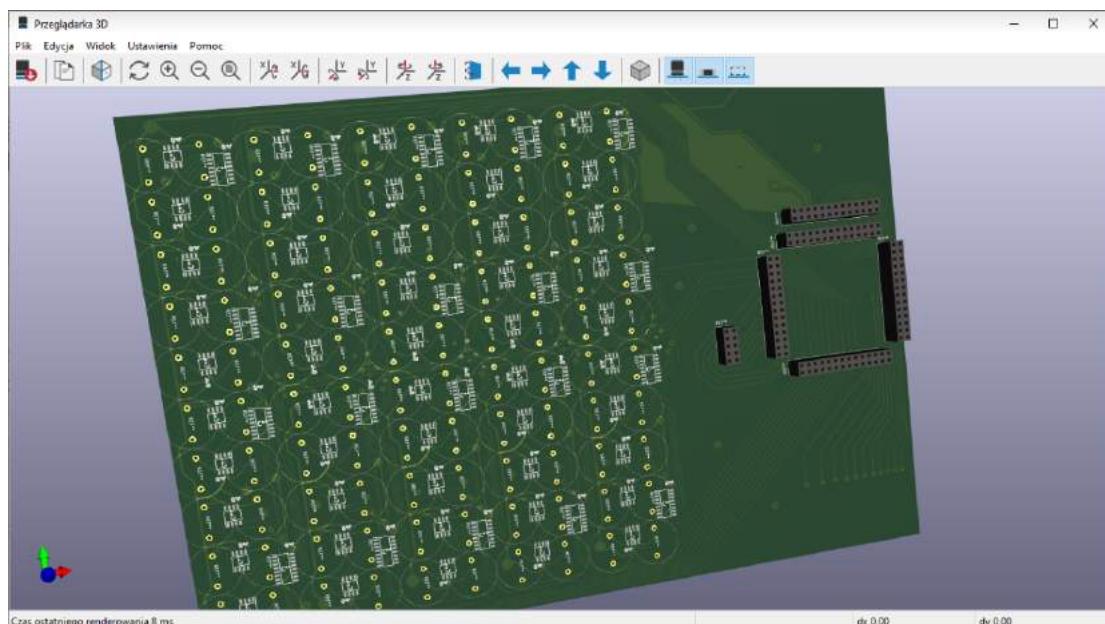
włączonych 120 mA. W tej wersji, na jedną płytke (100 głośników) przypadałoby około 370 mA. Dobierając odpowiedni zapas, ścieżki zasilające układy TC4427 i głośniki zaprojektowaliśmy na 0,5 A.



Rys. 66 Finalny projekt dolnej płytke

Źródło: zdjcie własne

Ostatnim krokiem było zrobienie miejsca na kondensatory i wyjścia sygnałowe na drugą płytke. Dla przejrzystości postanowiliśmy wykorzystać **kabel taśmowy 34-przewodowy**.



Rys. 67 Podgląd wyrenderowanego modelu płytki

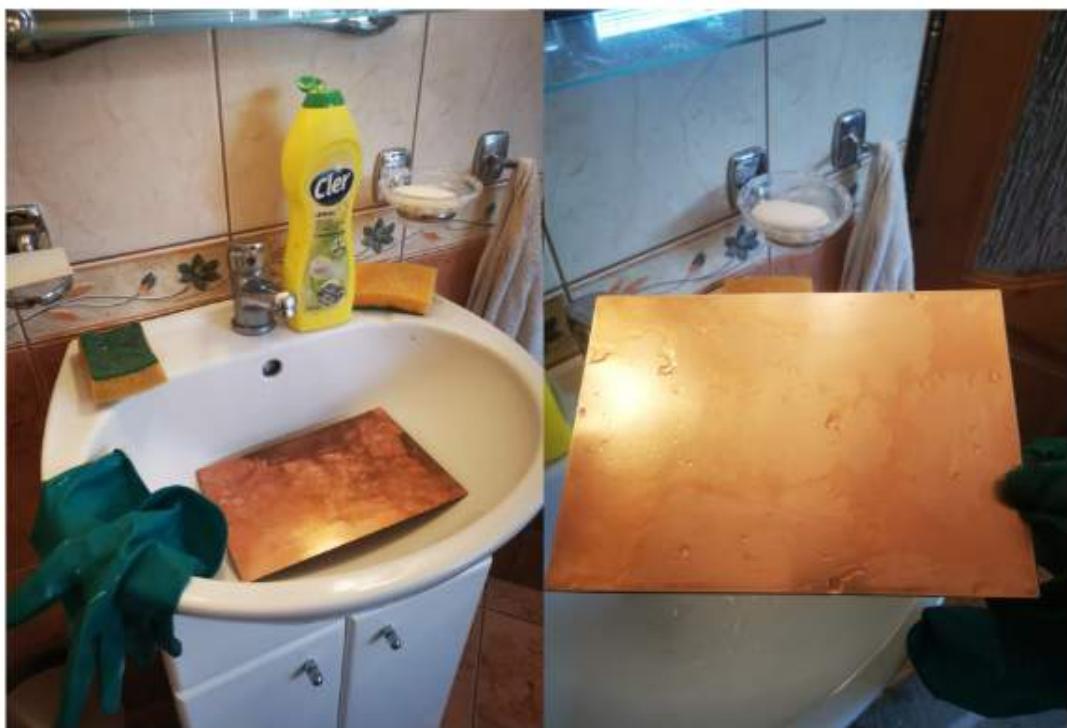
Źródło: zdjcie własne

II.2.3. Wykonywanie płytki PCB

Po zaprojektowaniu płytki przeszliśmy do jej wykonania. Potrzebne były następujące materiały:

- laminat PCB,
- środki odtłuszczające (rozpuszczalnik, aceton, płyn do mycia naczyń),
- druki z wzorem płytki,
- papier ścierny o gradacji 600,
- nadsiarczan sodu.

Laminat PCB w pierwszej kolejności musiał być **dokładnie wyczyszczony**. Płytkę najpierw przetarliśmy ostrą stroną gąbki do mycia naczyń. Następnie laminat pocieraliśmy papierem ściernym o gradacji 600, tak aby wszystkie odbarwienia, a także wszystkie nieczystości zostały usunięte.



Rys. 68 Stanowisko czyszczenia laminatu PCB

Źródło: zdjęcia własne

Kolejnym bardzo ważnym krokiem było **odtluszczenie laminatu**. W tym celu przy użyciu płynu do mycia naczyń wyszorowaliśmy powierzchnię w tym samym czasie polewając ją z kranu ciepłą wodą, tak aby po nalaniu wody na płytę, utrzymywała się ona na jej powierzchni i nie tworzyła "tłustych" oczek. Po ok. 10 min mycia laminatu przeszliśmy do kolejnego kroku. **Przygotowaliśmy wzory z nadrukami płytka** na różnych rodzajach kartek. Ze względu na brak papieru kredowego, który w tym zastosowaniu jest najlepszy, staraliśmy się wyszukać jego zamienników. Kartka miała

być śliska i zarazem nie lepiącą się. Przygotowaliśmy kilka opcji kartek: kartki z czasopism, kalendarza, papier fotograficzne, a także kalkę.



Rys. 69 Przygotowane wydruki

Źródło: zdjęcie własne

Mając nadruki, wycięliśmy je wzdłuż wzoru tak aby pasowały do laminatu. W poradniku dowiedzieliśmy się, że najlepszym sposobem na otrzymanie własnoręcznie płytki PCB jest wykorzystanie **techniki z żelazkiem**. Inne to np. własnoręcznie rysowanie ścieżek przy pomocy permanentnego markeru. Żelazko w tej metodzie powinno być bez dziur, aby przylegało do płytki jak największą powierzchnią. Niestety jedynie starsze żelazka miały taką budowę, co nie ułatwiało znalezienia odpowiedniego narzędzia. Znalezione przez nas pasujące żelazko było bez wtyczki, więc znaleźliśmy nieużywaną i przykręciliśmy ją do przewodów żelazka, a następnie nagrzaliśmy do ok. 200 °C.



Rys. 70 Stanowiska przenoszenia druku na laminat

Źródło: zdjęcia własne

Na dokładnie dopasowany druk do powierzchni laminatu i do niego przytknięty zaczeliśmy przykładać żelazko przez kolejne ok. 5 min. Czas musiał być dobrany optymalnie, gdyż **zbyt krótkie trzymanie żelazka na druku powodowało złe odejście tuszu od kartki, a zbyt długie - jego stopienie**. Po tym procesie należało wrzucić laminat do zimnej wody i odczekać ok. 10 min, tak aby woda przesiąkła całą kartkę. Gdy czas minął można było w końcu zacząć powoli odklejać kartkę. Trzeba było robić to bardzo delikatnie aby nie uszkodzić już dobrze odbitych ścieżek na laminacie.

Pierwsze próby były nieudane (Rys. 71). Proces musielismy zaczynać od początku, gdyż nie wszystkie ścieżki zostały odbite. Problem wynikał z niedokładności odtłuszczenia płytka jak i zastosowanego materiału kartki. W etapie moczenia pojawiło się pęcherzyków z powietrzem między kartką a laminatem sugerowało, że tusz w tym miejscu nie odszedł od kartki.



Rys. 71 Nieudane próby
Źródło: zdjęcia własne

Gdy wreszcie wszystkie ścieżki się odbiły mogliśmy zacząć czyszczenie laminatu. Połączenia były tak małe, że milimetrowe kawałki papieru zostawały między ścieżkami. Wszystko to dokładnie **wyczyściliśmy za pomocą szczoteczki do mycia zębów**, zważając na to, aby nie uszkodzić połączeń. Nie wszystkie jednak udało się usunąć za pomocą szczoteczki, dlatego w tym celu użyliśmy skalpelu aby dokładnie oczyścić płytke. Gdy wreszcie płytka nadawała się do trawienia - **przygotowaliśmy odpowiedni roztwór** rozrabiając **250 g nadsiarczanu sodu z 1,25 l wody w temp. 50 °C**. Następnie zmieszaliśmy go tak, aby uzyskać przezroczystą barwę i wlaliśmy go do pojemnika, a następnie zanurzyliśmy w nim naszą płytke (Rys. 72).



Rys. 72 Proces trawienia płytki
Źródło: zdjęcia własne

Proces trawienia trwał ok. 40 min, przy czym co jakiś czas roztwór był wprawiany w ruch, aby przyspieszyć proces trawienia. Po wyjęciu płytki dokładnie oczyściliśmy ją pod gorącą wodą. Ostatnim etapem było zmycie pozostałego tuszu z płytki. Zrobiliśmy to za pomocą czystego acetonu, nakładając go w małych ilościach na papier i następnie wycierając dokładnie płytę. Cały proces trzeba było powtórzyć 2-krotnie gdyż potrzebowaliśmy 2 płyt, po jednej na góre i na dół lewitatora. Efekt końcowy przedstawia Rys. 73.



Rys. 73 Płytki dolna i górna po ukończonym procesie trawienia
Źródło: zdjęcia własne

Po skończonym procesie tworzenia płytek przeszliśmy do jej **wiercenia**. W tym celu zaopatrzyliśmy się w wiertła 1 mm do wywiercenia otworów pod nóżki głośników oraz

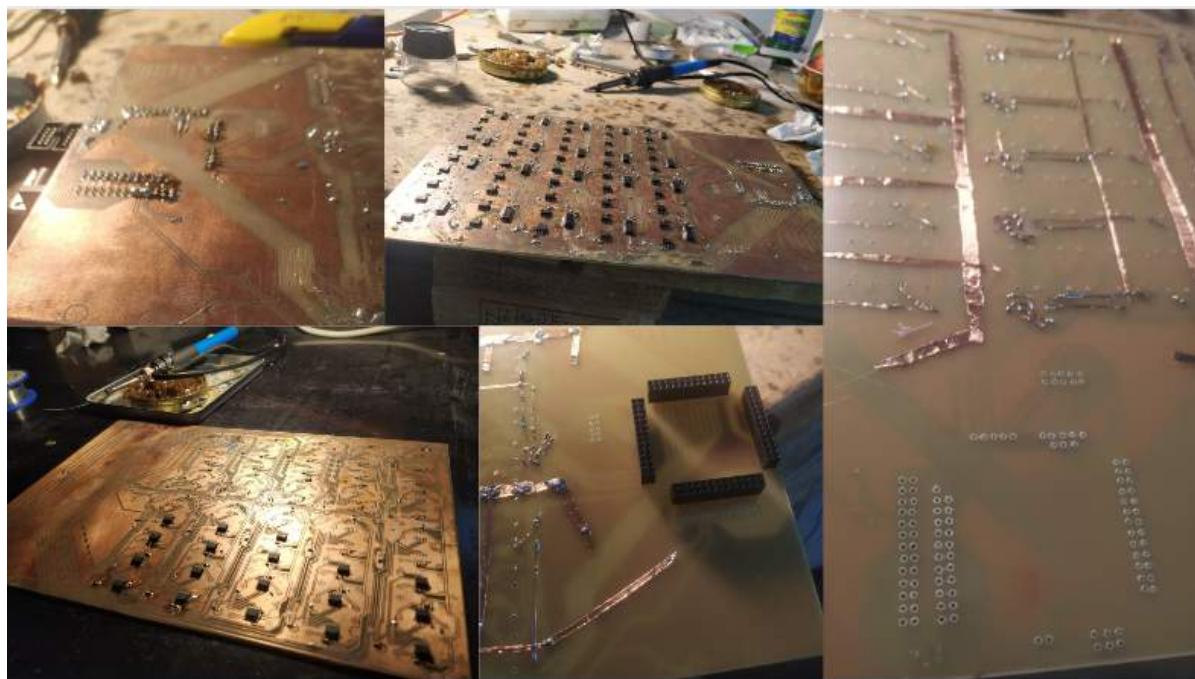
0,5 mm w celu umożliwienia przeprowadzenia połączenia z góry płytki na jej spód (jedna płytka była jednostronnie pokryta miedzią - dlatego było konieczne dorobienie ścieżek za pomocą miedzianej taśmy, druga płytka natomiast była już dwustronnie pokryta).



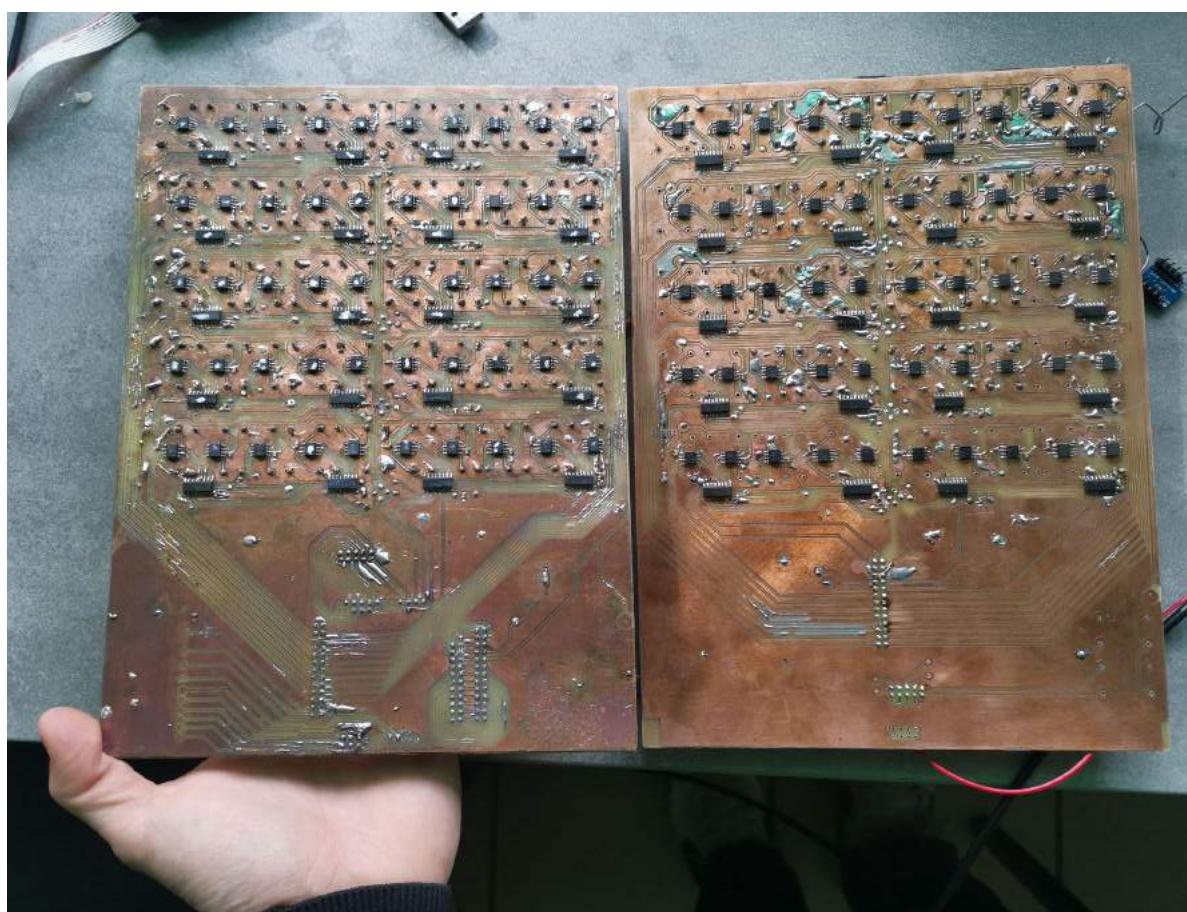
Rys. 74 Wywiercanie otworów 1,5 mm oraz 1 mm na płytce
Źródło: zdjęcia własne

Po czasochłonnym przewierceniu całej płytka, zabraliśmy się za lutowanie układów TC4427, HC595 oraz kondensatorów 100 nF na płytke, co również pochłonęło sporo czasu. **Całkowity czas potrzebny na wykonanie jednej płytka wyniósł ok. 20 h.** Było to niezwykle czasochłonne i trudne technicznie, ale wiedza, nabycie doświadczenia oraz satysfakcja z efektu końcowego przewyższały trud włożony w cały proces.

Po ukończeniu obydwu płytka zweryfikowaliśmy wszystkie połączenia, również pod względem przerw w połączeniach. Zauważylismy, że kilka układów TC4427 było spalonych, więc wymieniliśmy je. **Płytki były gotowe do montażu głośników (Rys. 76).**



Rys. 75 Etapy wykonywania połączeń
Źródło: zdjęcia własne

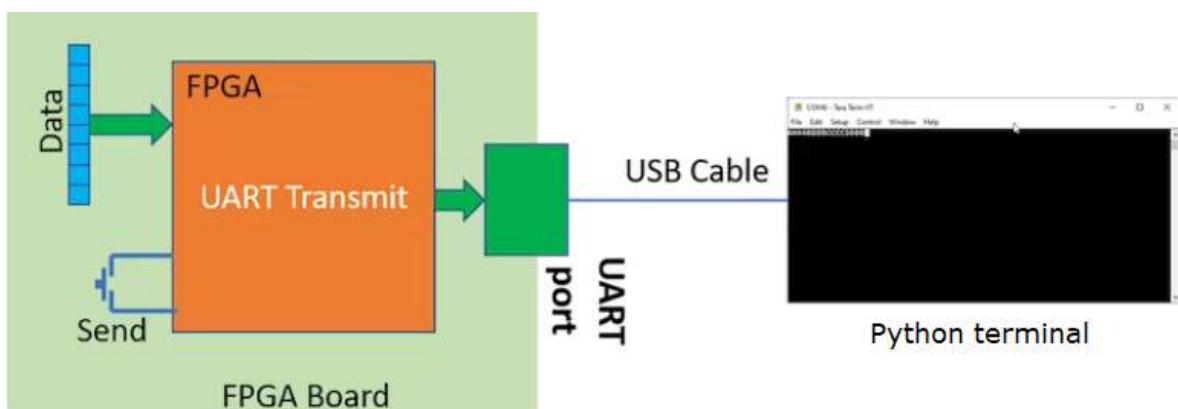


Rys. 76 Ukończone płytki PCB
Źródło: zdjęcie własne

II.2.4. Projekt programu do lewitacji

Podczas pracy z ogólnodostępnym programem 3D Sim Asiera Marzo, zauważaliśmy wiele czynników wykluczających jego użycie w naszym projekcie. Postanowiliśmy więc opracować swój własny program do lewitacji, zwracając uwagę, aby był łatwy w obsłudze a zarazem użyteczny. Program miał głównie służyć do realizacji lewitacji po skomplikowanych torach ruchu a także do jego wizualizacji.

Program obliczający przesunięcie w FPGA zawierał dużo miejsca, a dla 32 głośników ledwo starczało mocy obliczeniowej. Postanowiliśmy zmienić koncepcję i zamiast obliczać przesunięcia w FPGA zdecydowaliśmy się na zrobienie tego za pomocą Pythona. FPGA miało już tylko komunikować się z programem, który obliczał przesunięcia fazowe i wysyłać odpowiedni sygnał do rejestrów przesuwnych.



Rys. 77 Schemat komunikacji między Pythonem a FPGA

Źródło: hackster.io/mohammad-hosseinabady2/uart-transmit-with-hls-for-fpga-ab1d51

Komunikacja pomiędzy Pythonem a układem FPGA (Rys. 77) wymaga programu po obydwu stronach. Istnieje wiele sposobów takiej komunikacji, między innymi przez interfejsy SPI, I2C, Ethernet czy też USB. W naszym projekcie zdecydowaliśmy się na wybór komunikacji UART (ang. Universal Asynchronous Receiver-Transmitter), ze względu na dysponowanie ograniczonymi możliwościami sprzętowymi. Komunikacja ta odbywa się za pomocą dwóch linii szeregowych: linii Tx (Transmit) oraz linii Rx (Receive). Linia Tx służy do przesyłania danych z Pythona do FPGA a linia Rx do przesyłania danych z FPGA do Pythona.



Rys. 78 Analizator stanów logicznych oraz przewód USB-UART
Źródło: abc-rc.pl

Pracę zaczeliśmy od zakupu **analizatora stanów logicznych**, kabla **USB-UART** oraz zainstalowania programu **Logic Saleae**. Narzędzia te posłużyły do testowania komunikacji między Pythonem a FPGA.

Bauds	Bits/s	Bit duration	Speed	Actual speed	Actual byte duration
230400 bauds	230400 bits/s	4.340 µs	28800 bytes/s	23040 bytes/s	43.403 µs
460800 bauds	460800 bits/s	2.170 µs	57600 bytes/s	46080 bytes/s	21.701 µs
576000 bauds	576000 bits/s	1.736 µs	72000 bytes/s	57600 bytes/s	17.361 µs
921600 bauds	921600 bits/s	1.085 µs	155200 bytes/s	92160 bytes/s	10.851 µs

Rys. 79 Tabela prędkości transmisji szeregowej
Źródło: lucidar.me/en/serialib/most-used-baud-rates-table

Istnieją różne standardowe prędkości transmisji danych dla interfejsu UART, takie jak 9600 bps, 115200 bps, 57600 bps, aż do nawet 921600 bps. Im większa wartość baudratu, tym większa szybkość przesyłania danych między portem szeregowym a urządzeniem. Istnieją jednak pewne wady zbytniego zwiększenia szybkości transmisji. Jednym z problemów jest możliwa większa liczba błędów w transmisji, zwłaszcza w przypadku większych odległości lub zakłóceń elektromagnetycznych. Dodatkowo aby komunikacja ta działała poprawnie oba urządzenia (do wysyłania i odbierania) muszą działać z tą samą prędkością transmisji danych.

Podłączylismy analizator stanów logicznych z USB-UART i podpieliśmy go do portu USB komputera. Następnie zaczeliśmy od napisania prostego programu w Pythonie (Rys. 80), który miał zakodować liczbę i wysłać ją przez USB-UART do analizatora stanów logicznych. Dalej program Logic Saleae miał odczytywać tą liczbę i pokazać ją na symulacji przebiegów.

```

nowy.py > ...
1  import struct
2  import serial
3  import time
4  ser = serial.Serial(port='COM3', baudrate=921600)
5
6
7  a56 = struct.pack('>B', 56)
8  a10 = struct.pack('>B', 10)
9  a100 = struct.pack('>B', 100)
10 a250 = struct.pack('>B', 250)
11
12 ser.write(a56)
13 ser.write(a10)
14 ser.write(a100)
15 ser.write(a250)
16 ser.write(a250)
17 ser.write(a56)
18 ser.write(a250)
19 ser.write(a10)
20 ser.write(a100)
21 ser.close()

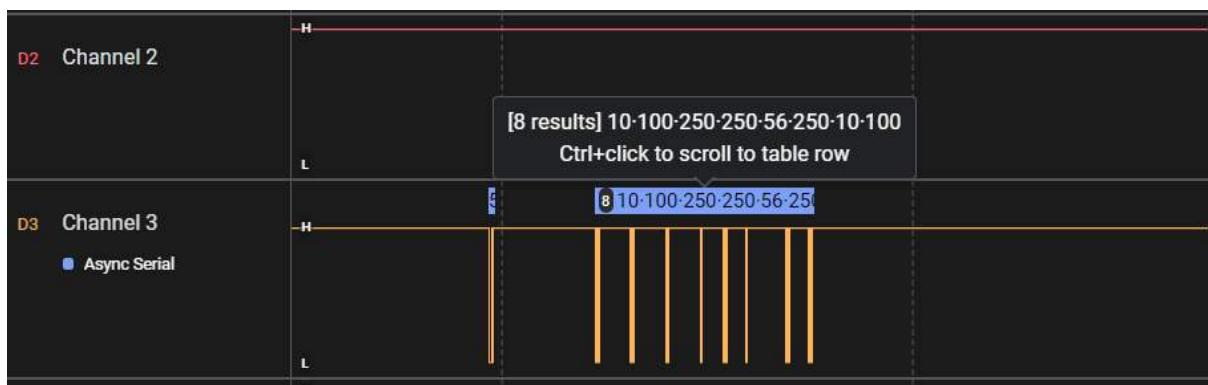
```

Rys. 80 Program testowy komunikacji szeregowej z USB-UART

Źródło: zdjęcie własne

Program składa się z czterech ważnych sekcji. Linia czwarta **otwiera port szeregowy COM3** z prędkością transmisji 921600 baudów. Linia od 7-10 wykorzystuje funkcję `struct.pack()` do **zamiany kolejno liczb 56, 10, 100 oraz 250 na ich binarną reprezentację** w postaci 1-bajtowego stringu w kolejności big-endian (najbardziej znaczący bajt pierwszy). Następnie funkcja `ser.write()` **wysyła te dane przez port szeregowy**, a funkcja `ser.close()` **kończy transmisję** poprzez zamknięcie portu szeregowego.

Tak stworzony program uruchomiliśmy wraz z programem logic saleae i obserwowaliśmy rezultat.



Rys. 81 Logic Saleae- testowanie odbieranych wyników

Źródło: zdjęcie własne

Jak widzimy na Rys. 81, **komunikacja się powiodła** - program logic saleae odebrał wysłane dane przez port szeregowy USB-UART.

Następnie przeszliśmy do implementacji programu komunikacji w kodzie odpowiedzialnym za obliczanie przesunięć fazowych. Po modyfikacji, uruchomiliśmy go i obserwowałyśmy na nowo odbierane wartości w programie (Rys. 82).



Rys. 82 Testowanie kodu obliczającego przesunięcia fazowe
Źródło: zdjęcie własne

Zauważaliśmy, że przerwy między odbieraniem sygnału między dwoma kolejnymi wartościami potrafiły sięgać do 1000 μ s czyli równo 1 ms. Biorąc pod uwagę bardziej skomplikowane i dłuższe tory ruchów **opóźnienie to było bardzo znaczące**. Gdyby chcieć zaimplementować to do głównego programu prosty ruch o 5 cm, mógłby trwać od kilku do kilkunastu sekund, a więc parametr szybkość ruchu w opcjach wyboru nie miałaby żadnego sensu. Również **ograniczyłoby to bardzo możliwości testów szybkości lewitacji**. Postanowiliśmy znaleźć problem i go rozwiązać.

Pierwszym rozwiązaniem, które postanowiliśmy przetestować była **zmiana obecnych funkcji torów ruchu**. Obliczają one współrzędne każdego położenia, następnie wzywają funkcję obliczającą przesunięcia fazowe, która później konwertuje je na numery kanałów. Obliczanie przesunięć jest dokonywane cały czas na nowo, więc czas potrzebny na wykonanie ruchu automatycznie wydłuża się ze względu na przeciążenie obliczeniowe. Zmiana ta miała polegać na **dodaniu do istniejących funkcji torów ruchu funkcji, która jako pierwsza oblicza wszystkie numery kanałów i zapisuje je w liście**. Następnie wzywana jest funkcja która wysyła każdy element listy poprzez USB-UART. Ruch w takiej postaci odbywa się dzięki odczytywaniu gotowych wartości z pamięci, które wcześniej zostały obliczone. **Program najpierw oblicza przesunięcia, a potem korzysta z nich aż do zmiany toru ruchu**.

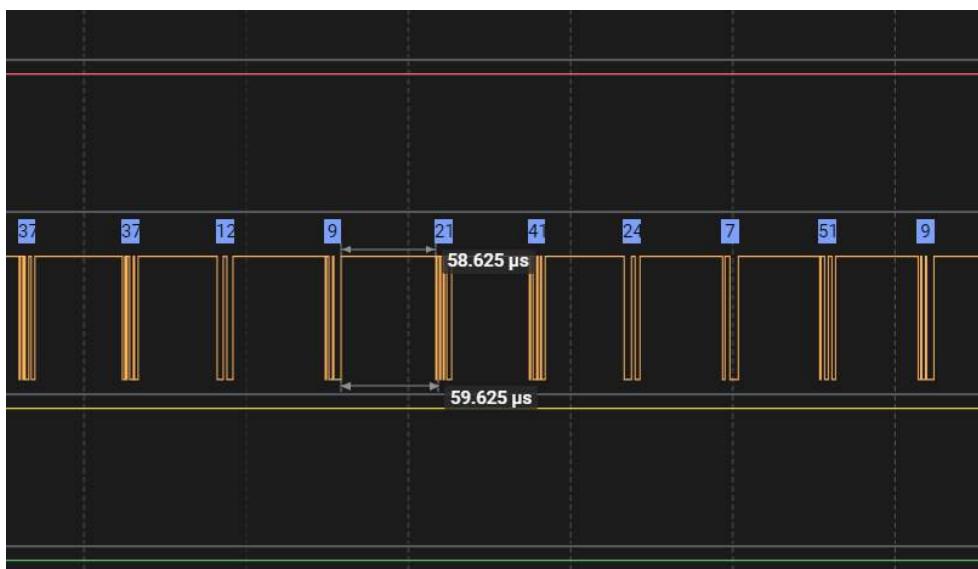
```
def infinite_loop():
    for phase_init in phases_init:
        result_send = struct.pack('>B', int(phase_init))
        ser.write(result_send)
        time.sleep(0.001)
```

Rys. 83 Implementacja funkcji wysyłającej przesunięcia fazowe poprzez iterowanie

każdego elementu z listy zawierającej numer kanału

Źródło: zdjęcie własne

Analiza różnicy w wysyłaniu (Rys. 84) pokazała, że opóźnienie zmalało kilkakrotnie. Był już to bardzo obiecujący wynik który pozwoliłby nam na **szybsze sterowanie lewitacją**.



Rys. 84 Różnica w wysyłaniu po implementacji zmodyfikowanego programu z zastosowaniem iteracji z listy

Źródło: zdjęcie własne

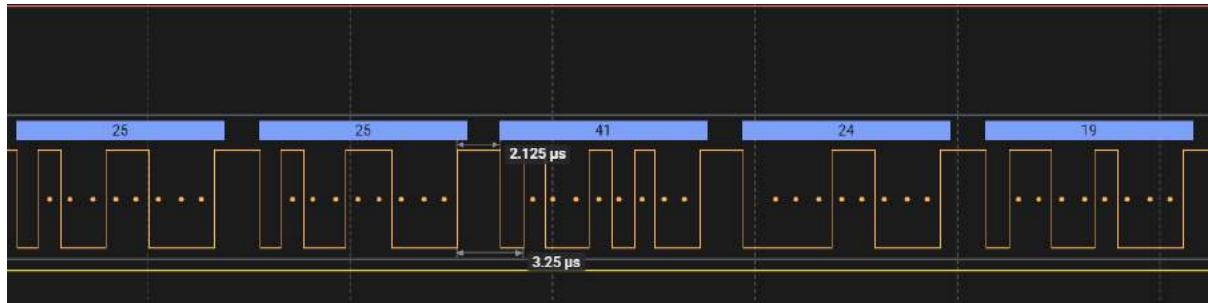
Poszukując informacji na temat opóźnień natrafiliśmy na artykuł mówiący o stosowaniu **wyrażeń generatorowych** do tworzenia tablic. Temat zgłębiliśmy i stworzyliśmy nową wariację kodu. Iterację tablic zamieniliśmy na kod ukazany na Rys. 85.

```
init_data = b''.join([struct.pack('>B', int(phase_init)) for phase_init in phases_init])
ser.write(init_data)
phase_data = b''.join([struct.pack('>B', int(phase)) for phase in phases])
ser.write(phase_data)
```

Rys. 85 Zapis danych za pomocą wyrażeń generatorowych

Źródło: zdjęcie własne

Program skompilowaliśmy, ustawiliśmy program odbierający i obserwowałyśmy rezultat. Otrzymany wynik był zaskakujący. **Opóźnienie udało nam się zmniejszyć kilkaset razy.**



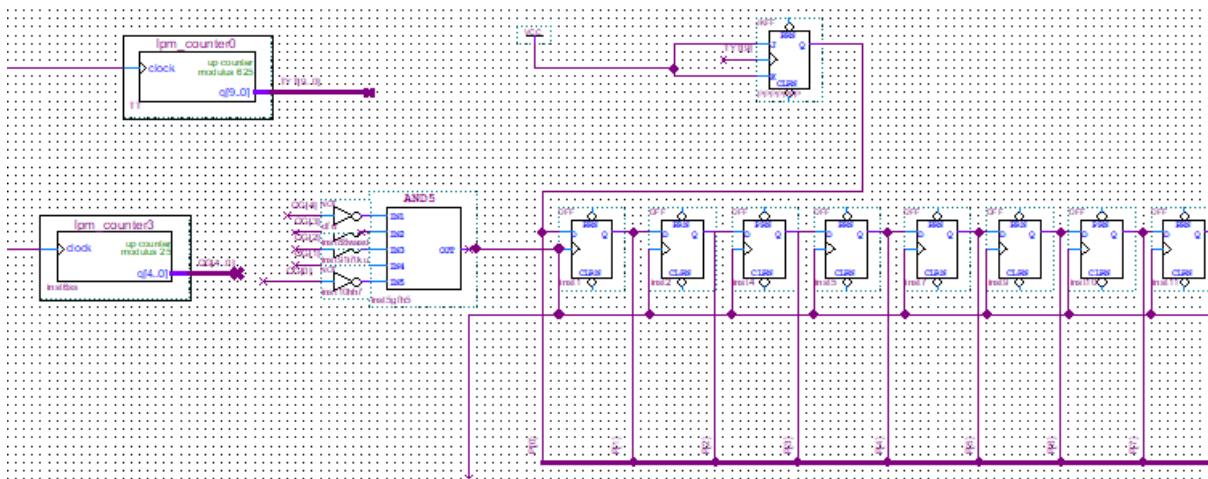
Rys. 86 Opóźnienie względem następujących po sobie danych po zastosowaniu wyrażeń generatorowych w kodzie do komunikacji z USB-UART

Źródło: zdjęcie własne

II.2.5. Program Quartus

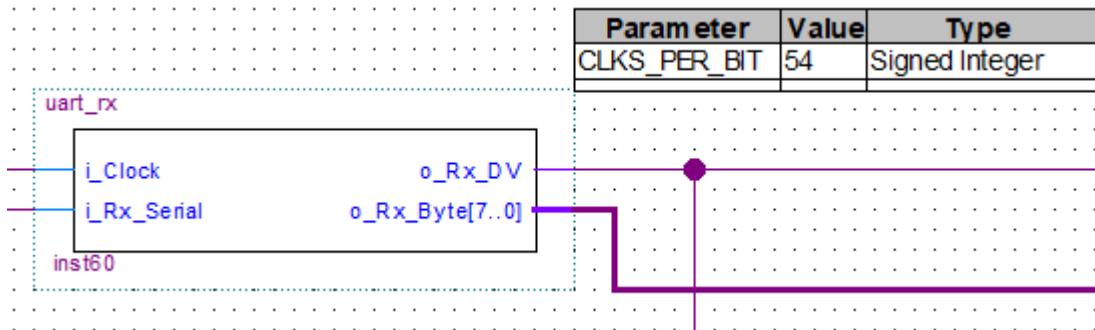
Od tego momentu FPGA ma za zadanie tylko **zapamiętywać opóźnienie** przypisane do konkretnego głośnika i **nadawać na niego odpowiedni sygnał**, kodując go wcześniej odpowiednio do 74HC595. Program podzielony jest na kilka głównych sekcji:

- wytwarzanie sygnałów 40 kHz w pełnej gamie przesunięć do 180 stopni,
- odczyt danych i konwersja liczb z zapisu szeregowego na równoległy,
- odpowiednie rozmieszczenie opóźnień dla głośników i ich zapamiętanie,
- układ kodujący przesunięcia do hc595,
- układ dobierający odpowiedni sygnał 40 kHz do liczby,
- moduł komunikacji FPGA do Pythona.



Rys. 87 Układ wytwarzający 40 kHz i przerzutniki D opóźniające sygnał
Źródło: zdjęcie własne

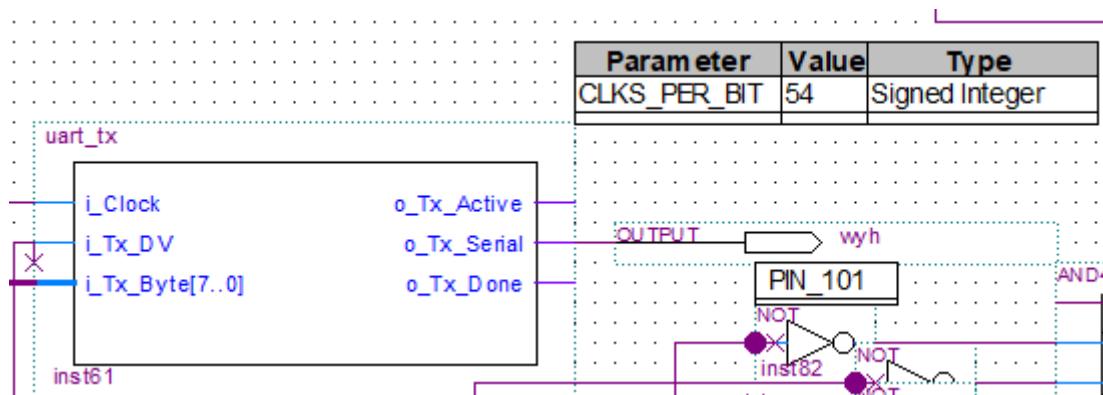
Nowy program jako, że opiera się na innej zasadzie niż poprzedni musielismy przebudować. Ze względu na ograniczenia pamięci i mocy obliczeniowej **obniżyliśmy ilość generowanych przesunięć z 125 do 50**. Zauważylismy, że jest to wartość w zupełności wystarczająca, a lewitacja nie wymaga aż takiej dokładności przesunięć.



Rys. 88 Moduł odbierający UART Rx

Źródło: zdjęcie własne

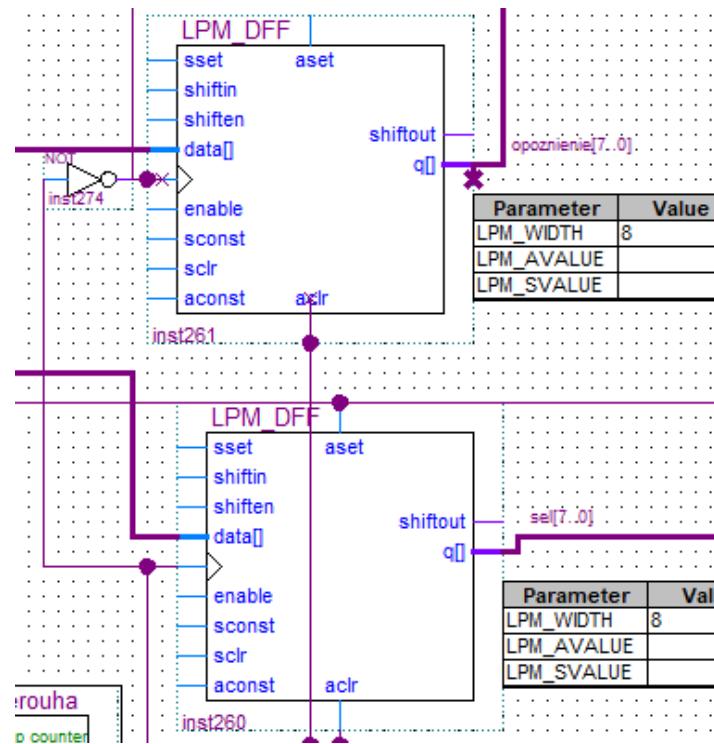
W programie potrzebny jest **moduł dekodujący sygnał UART** (Rys. 88) nadawany z komputera, który zamienia bity wysypane szeregowo w zapis równoległy z poszczególnymi rangami bitów. Wartość 54 jest stosunkiem częstotliwości zegara do niego przypisanego i prędkości transmisji (50 MHz/921600 Bd). Po odebraniu 8 bitów (licząc bez bita startowego) ustawia on wartość 1 na jeden cykl zegara. Ma to za zadanie sygnalizować odebranie pełnej liczby.



Rys. 89 Moduł wysyłający UART Tx

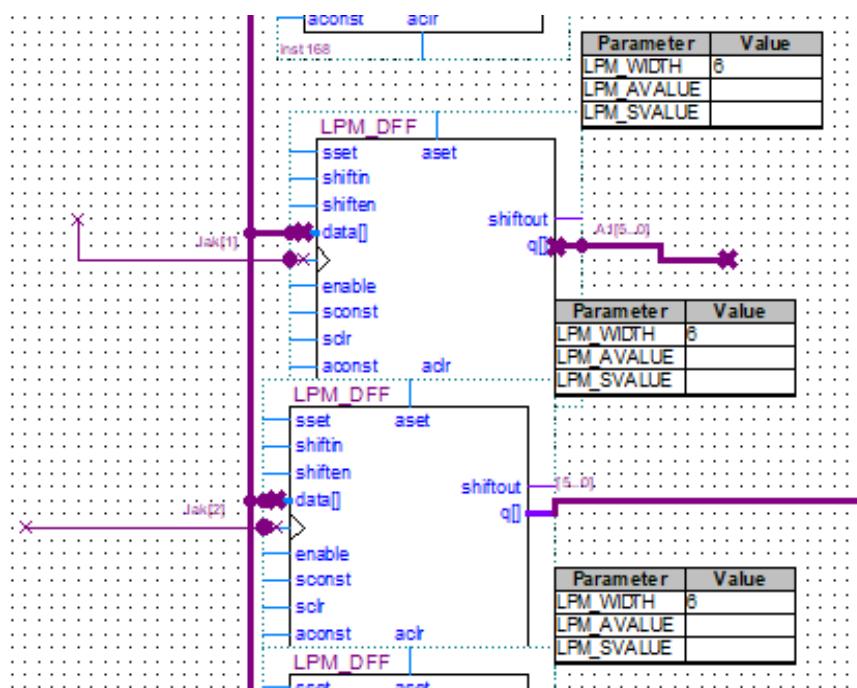
Źródło: zdjęcie własne

Do sprawdzenia poprawności odbieranych danych zastosowaliśmy moduł UART Tx (Rys. 89). Działa odwrotnie do poprzedniego - koduje liczbę z zapisu równoległego na szeregowy.



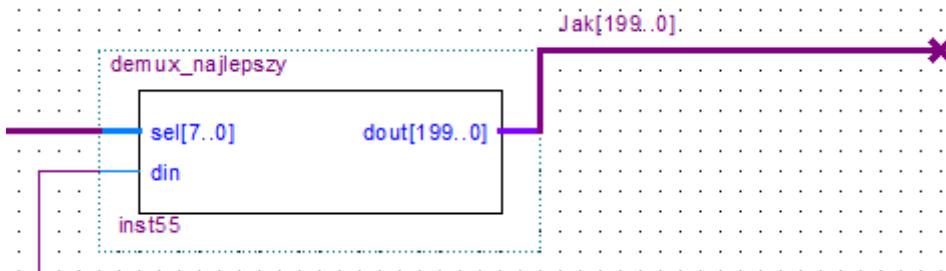
Rys. 90 Przerzutniki D zapamiętujące numer głośnika i kanał opóźnienia
Źródło: zdjęcie własne

Po odebraniu wartości program ją odpowiednio zapisuje - albo jako numer głośnika, albo numer opóźnienia fazowego. Dane te są przesyłane po kolejno, w zależności czy odebrana liczba jest parzysta lub nie, a w zależności od kolejności odebrania program zapisuje ją do odpowiedniego przerzutnika D (Rys. 90). Jest to potrzebne aby otrzymać dwie liczby w jednym czasie.



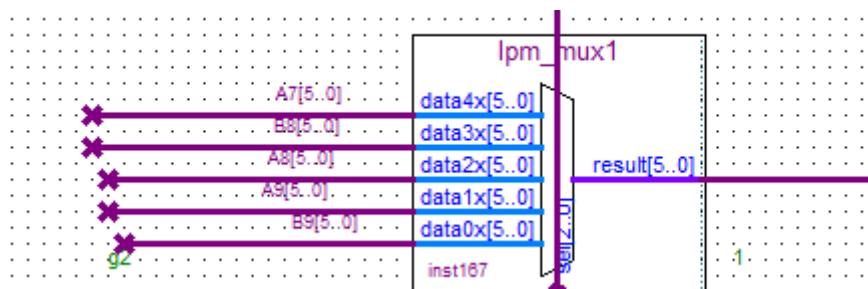
Rys. 91 Część przerzutników D, które zapamiętują opóźnienie dla danego głośnika
Źródło: zdjęcie własne

Numer opóźnienia przypisany jest jako dana wejściowa na 200 przerzutników D. Każdy z nich ma przypisany do siebie jeden głośnik. Jest on dla niego pamięcią opóźnienia.



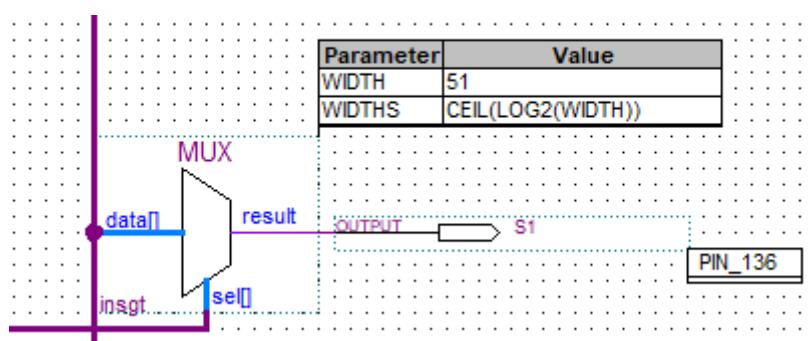
Rys. 92 Demultplekser wybierający przerzutnik D do którego nastąpi zapis opóźnienia
Źródło: zdjęcie własne

Numer głośnika jest przesyłany na demultiplexer (Rys. 92), który wybiera jeden spośród 200 kanałów w zależności od odebranej liczby. Przesyła on na to wyjście sygnał din, który jest odpowiednio zsynchronizowany zegarem uruchamiającym się przy odebraniu przez UART Rx liczby. Inicjuje on zapis do pamięci odpowiedniego przerzutnika D numeru kanału opóźnienia.



Rys. 93 Demultplekser kodujący sygnał do rejestru przesuwnego
Źródło: zdjęcie własne

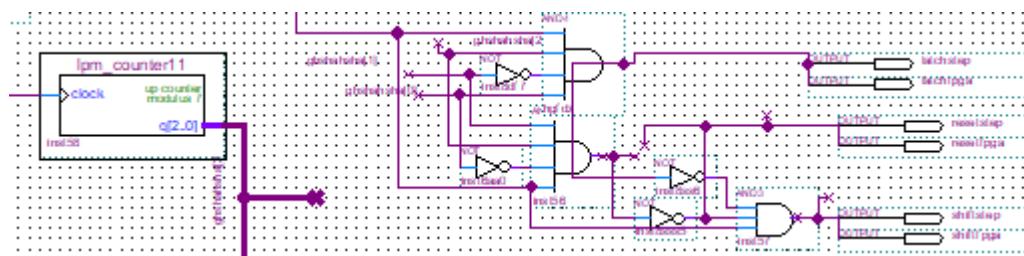
Przed zamianą liczby na sygnał 40 kHz należy odpowiednio zakodować numery, aby rejestr przesuwny znajdujący się fizycznie na płytce poprawnie przesunięcie odczytał. Numery kanałów opóźnień są przypisane po 5 do jednego demultiplexera, tak aby 74HC595 miał również 5 tych samych numerów głośników.



Rys. 94 Demultplekser dobierający odpowiedni sygnał 40 kHz do wartości

Źródło: zdjęcie własne

Numer ostatecznie zostaje zamieniony na sygnał 40 kHz z odpowiednim przesunięciem fazowym.



Rys. 95 Układ przesyłania danych do rejestrów przesuwnych

Źródło: zdjęcie własne

Układy hc595 do działania potrzebują również sygnałów latach, shift i reset. Wykonane jest to za pomocą licznika zliczającego impulsy zegara 50 MHz aż do wartości równej 6. Kolejno, numery od 0 do 4 wybierają odpowiedni kanał opóźnienia na demultiplekserze i wysyłają sygnał shift (przesuwający bity na fizycznym rejestrze), 5 wysyła sygnał latch (wyświetlenie przesuniętych bitów), natomiast 6 resetuje zapamiętane na rejestrze przesuwnym liczby.

II.2.6. Aplikacja lewitacji w Pythonie

II.2.6.1. Interfejs

Kolejnym krokiem było stworzenie odpowiedniej aplikacji, pozwalającej na łatwe sterowanie obiektem. Niezbędnym było stworzenie interfejsu graficznego widzianego przez użytkownika (GUI) oraz zimplementowanie potrzebnych funkcji obliczających. Do tego celu wybraliśmy jedną z bibliotek Pythona, w którym napisany był przez nas cały program. **Biblioteka Tkinter** to jeden z najbardziej znanych pluginów do tworzenia GUI. Posłużył on nam do stworzenia całego interfejsu aplikacji, ale wraz z jej tworzeniem doinstalowaliśmy jeszcze kilka pakietów zwiększających możliwości programu (patrz: Rys 96). Cała aplikacja została napisana w edytorze VSCode.

```

➊ kopia.py > ...
 1  from tkinter import ttk
 2  from tkinter import *
 3  from customtkinter import *
 4  import math
 5  from PIL import ImageTk, Image
 6  import threading
 7  import matplotlib.pyplot as plt
 8  import matplotlib.patches as patches
 9  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
10  from matplotlib.figure import Figure
11  from tkinter import messagebox
12  import numpy as np
13  from tkinter import font
14  import time
15  import tkinter.colorchooser as colorchooser
16  from mat import *
17  from mat1 import *
18  import struct
19  import serial
20  import time

```

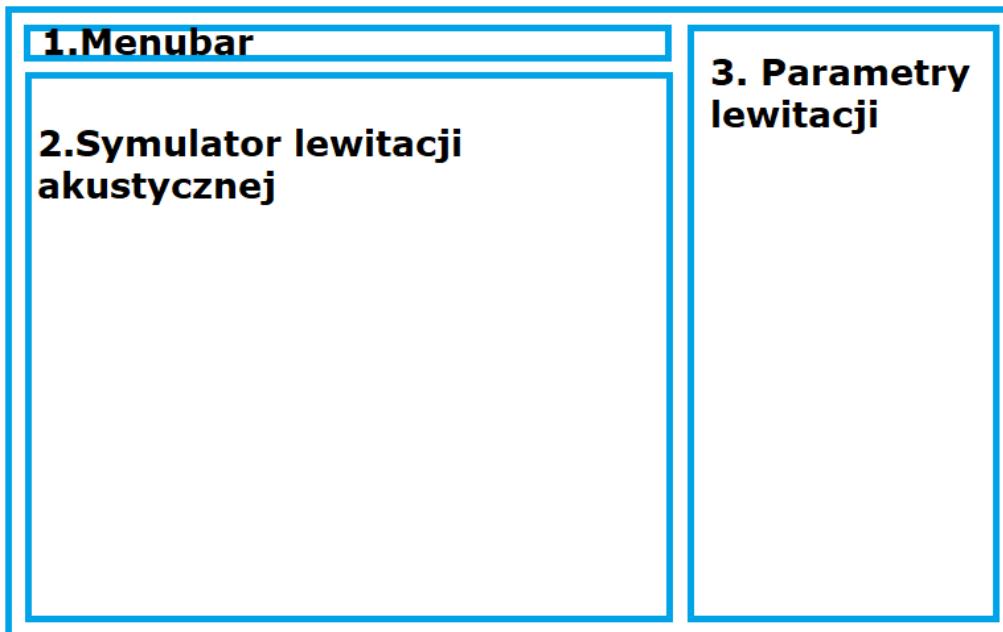
Rys. 96 Lista wszystkich użytych bibliotek

Źródło: zdjécie własne

Główne biblioteki interfejsu :

- **Tkinter** - standardowy moduł w Pythonie do tworzenia Tcl/Tk GUI interfejsu użytkownika,
- **Customtkinter** - dodatek bazujący na pakiecie Tkinter, zapewniający nowe i w pełni konfigurowalne widgety. Niestety jest to dość nowa biblioteka więc brakuje w niej wielu zmiennych, stąd nasze połączenie jej z biblioteką Tkinter.

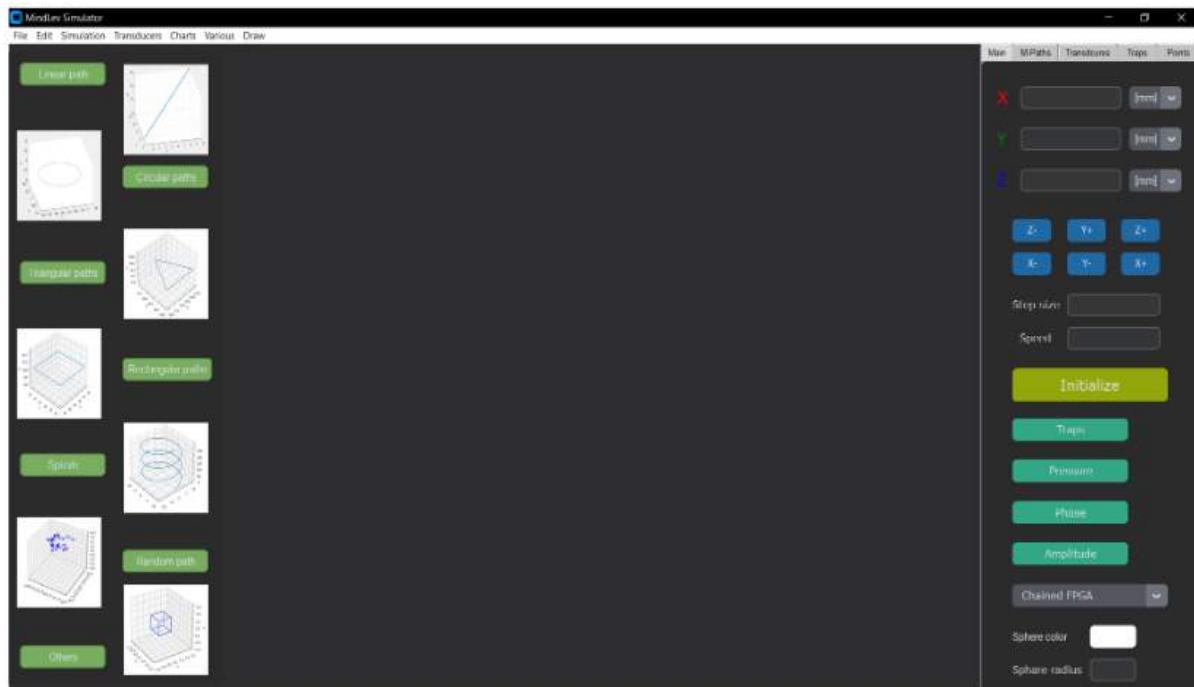
Rys. 97 przedstawia poglądowy zamysł aplikacji, którym kierowaliśmy się w dalszych pracach.



Rys. 97 Poglądowy szkic aplikacji

Źródło: opracowanie własne

Nasz pomysł na aplikację zaczęliśmy wcielać w życie. Struktura programu po kilku tygodniach przedstawia Rys. 98.



Rys. 98 Program lewitacji akustycznej
Źródło: zdjęcie własne

Zakładka po lewej stronie umożliwia wybór opcji ruchu po obranym torze w ramach siedmiu zaproponowanych przez nas opcji:

1. linear path (ścieżki po linii prostej),
2. circular path (ścieżki po okręgu),
3. triangular path (ścieżki po torze trójkątnym),
4. rectangular path (ścieżki po torze prostokątnym),
5. spirals (spirale),
6. random path (losowy ruch),
7. others (inne).

Tworzenie torów ruchu cząsteczki było nie lada wyzwaniem. W pierwszej kolejności musieliśmy zrozumieć zasadę działania biblioteki matplotlib, a także operacji tworzenia ruchu cząsteczki w samym Pythonie. Zgranie wszystkich procesów w jedno okno wyboru wymagało **ustawienia wątkowania** (threadingu) dla każdego z okien, tak aby nie zakłócać głównego wątku programu i tym samym zapobiec **brakowi responsywności** programu.

Każda z opcji ruchu różni się nieco opcjami wyboru. Do standardowych opcji należy wybór początkowych współrzędnych x, y, z, wielkość kroku, liczba oscylacji ruchu, oś poruszania a także prędkość obliczania/wysyłania danych.

Dodatkowo do każdego toru ruchu była dodana **inicjalizacja położenia obiektu**. Aby umożliwić ruch zgodnie ze współrzędnymi określonymi przez użytkownika, obiekt musiał w pierwszej kolejności się tam znaleźć, dlatego został ustanowiony punkt początkowy inicjalizacji. Program automatycznie obliczał najkrótszą drogę między tym punktem a punktem początkowym ruchu. Po ukończonej inicjalizacji ruch po danym torze ruchu został uruchamiany.

```

start = np.array((72, 72, 100))
end = np.array((start_x, start_y, start_z))

distance = np.linalg.norm(end - start)
direction = (end - start) / distance

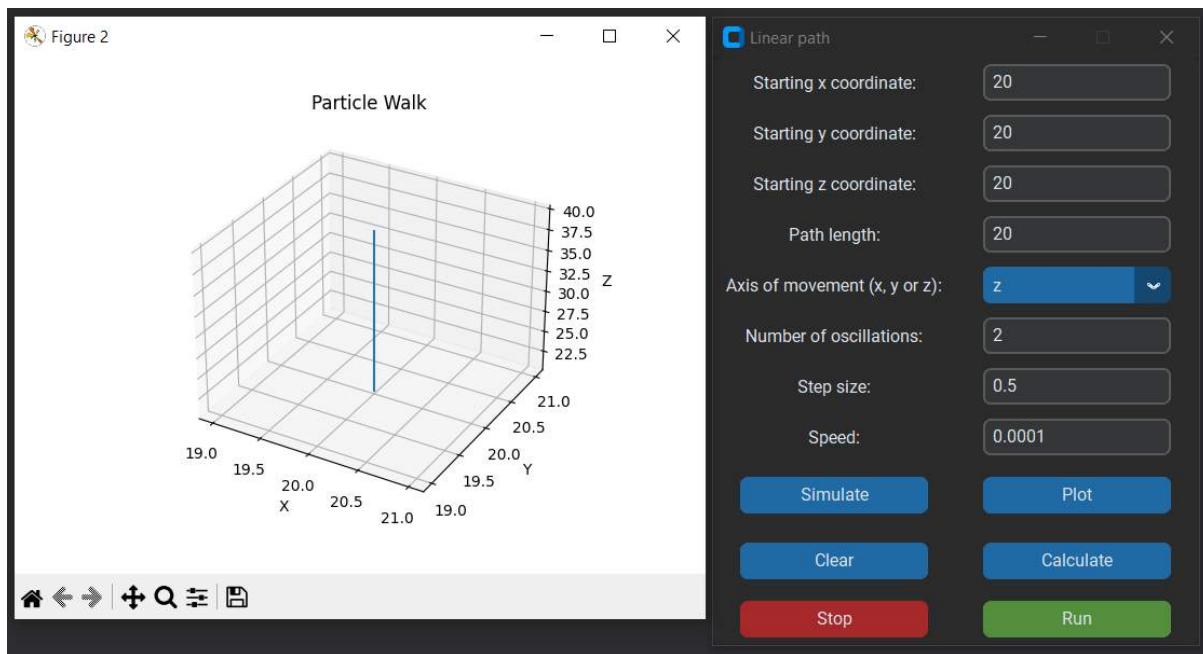
steps = np.arange(0, distance + 0.2, 0.2)

print("Steps of the walk:")
for step in steps:
    x,y,z = np.round(start + direction * step, 2)
    print(x,y,z)
    calculate_phases_ini(x,y,z)

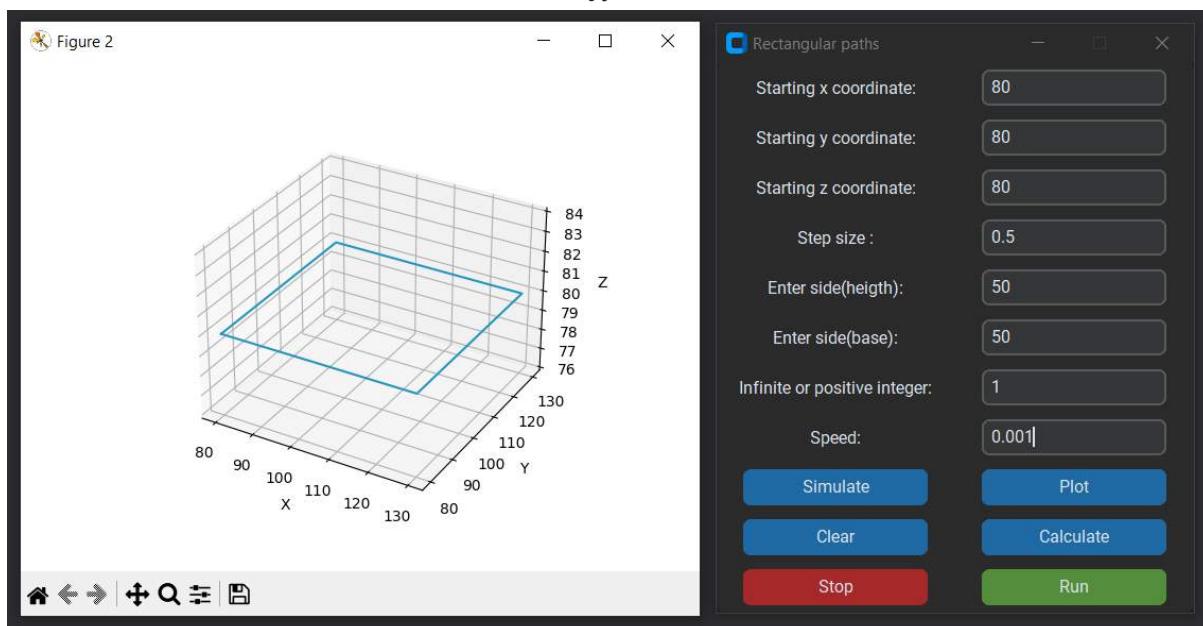
```

Rys. 99 Inicjalizacja położenia obiektu
Źródło: zdjęcie własne

Na Rys. 100 i Rys. 101 znajdują się przedstawione dwie z opcji poruszania: liniowy oraz po prostokącie wraz z utworzonymi przez program wykresami.

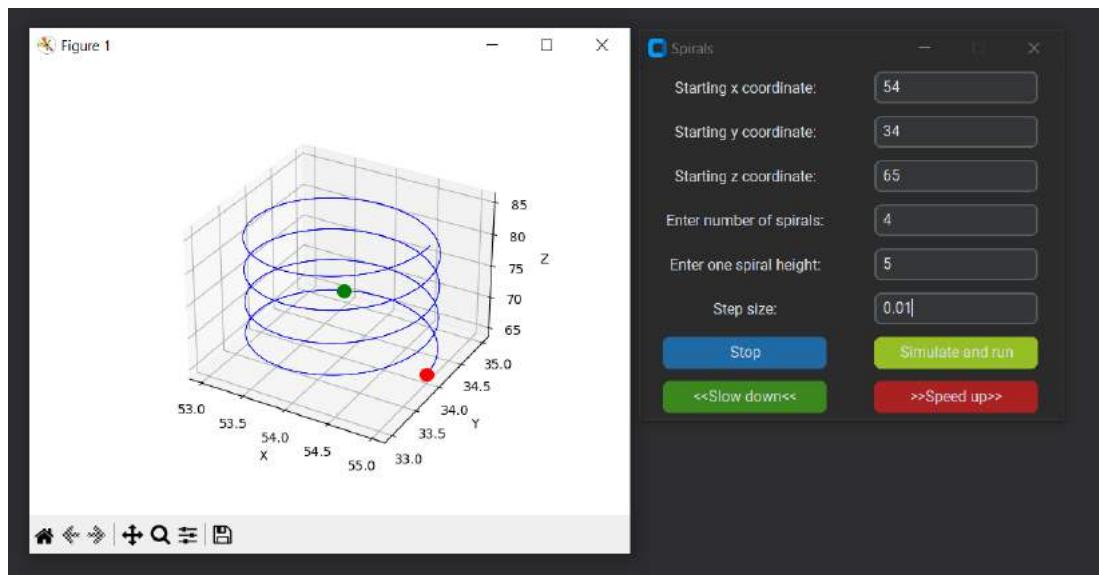


Rys. 100 Linearny tor ruchu
Źródło: zdjęcie własne



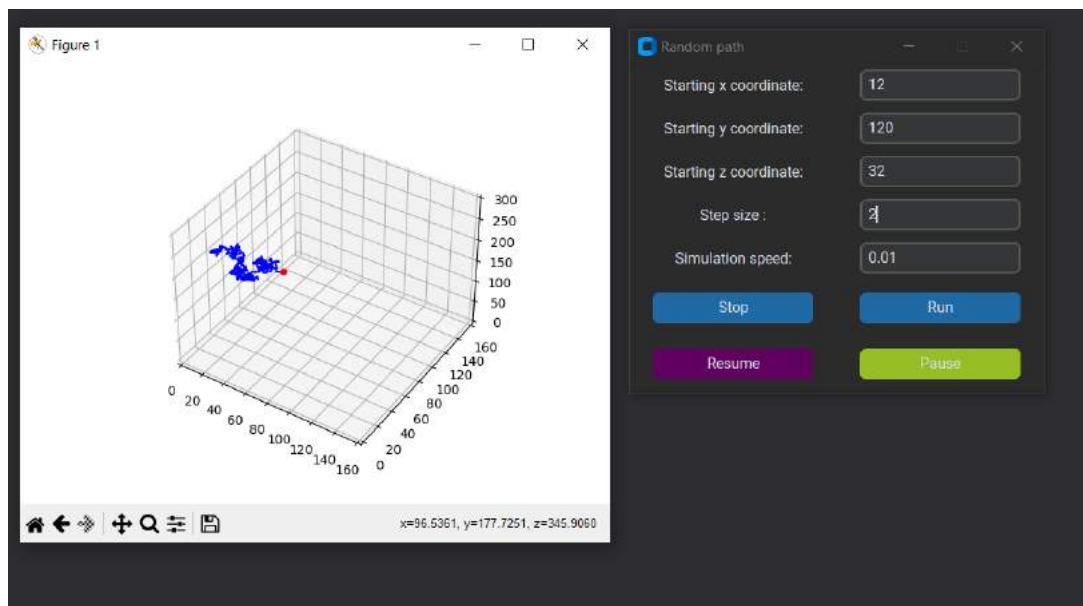
Rys. 101 Tor ruchu po prostokącie
Źródło: zdjęcie własne

Również jedną z zaproponowanych przez nas opcji jest spirals (Rys. 102) pozwalającą na ruch po spirali. Znajduje się tutaj standardowy wybór parametrów lewitacji, ale również możemy znaleźć takie opcje jak wybranie ilości tworzonych spirali w jednym ruchu, a także wysokość pojedynczej spirali.



Rys. 102 Ruch po spirali wraz z symulacją
Źródło: zdjęcie własne

Z kolei random path (Rys. 103) umożliwia ruch w losowym kierunku. W ruchu tym wystarczy określić współrzędne x, y, z, step size i następnie go uruchomić. Program korzysta z biblioteki **random** do określania losowego kierunku poruszania, oblicza przy użyciu wielkości kroku współrzędne nowego punktu, a następnie aktualizuje jego położenie.



Rys. 103 Ruch losowy wraz z symulacją
Źródło: zdjęcie własne

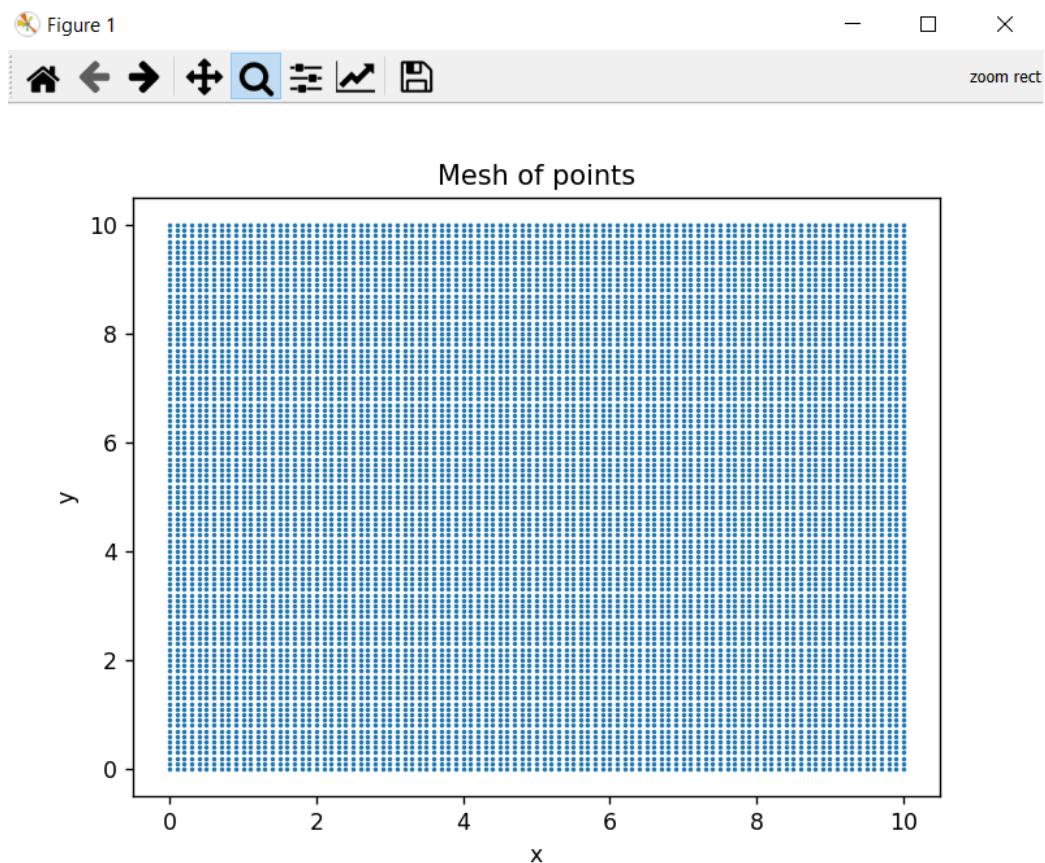
III.2.6.2 Sekcja rysowania

W trakcie pisania programu chcieliśmy, aby był on jak najbardziej użyteczny w procesie logistycznym, gdzie znalazłby zastosowanie w przyszłości. Aby więc przyszły

użytkownik mógł dowolnie ustalić trasę lewitującego obiektu, stworzyliśmy funkcjonalność, w której, oprócz wcześniej określonych możliwości poruszania czy to poprzez panel sterowniczy, czy też sekcje z danymi typami torów ruchu, użytkownik był w stanie **stworzyć swoją unikatową ścieżkę poprzez narysowanie jej**. Po wyszukaniu w odpowiednich źródłach informacji zdecydowaliśmy się zrobić to w **bibliotece matplotlib**. Biblioteka ta kojarzy się nam tylko i wyłącznie z tworzeniem prostych, a także i bardzo kompleksowych wykresów 2D lub 3D, lecz mało kto wie, że zawiera ona szereg udogodnień przeznaczonych do rysowania, w tym funkcję obsługi zdarzeń myszki, co pozwala na interaktywne rysowanie.

Samo rysowanie jednak nie zezwalało na użycie takiego programu od razu. Musieliśmy go odpowiednio zmodyfikować dodając szereg nowych funkcji. Pierwszym krokiem było stworzenie przez nas przy użyciu **biblioteki numpy**, funkcji **numpy.arange()**, siatki punktów, rozdzielonej od siebie o odległość równą **0,1 mm**. Wielkość 0,1 wynika z faktu, że linia narysowana przez użytkownika posiada nieskończenie wiele punktów, a nieskończona rozdzielcość nie miałaby racji bytu. Zastosowanie siatki eliminowało ten problem, gdyż określona na niej liczba punktów, zmusiłaby ścieżkę do dobrania sensownej rozdzielcości.

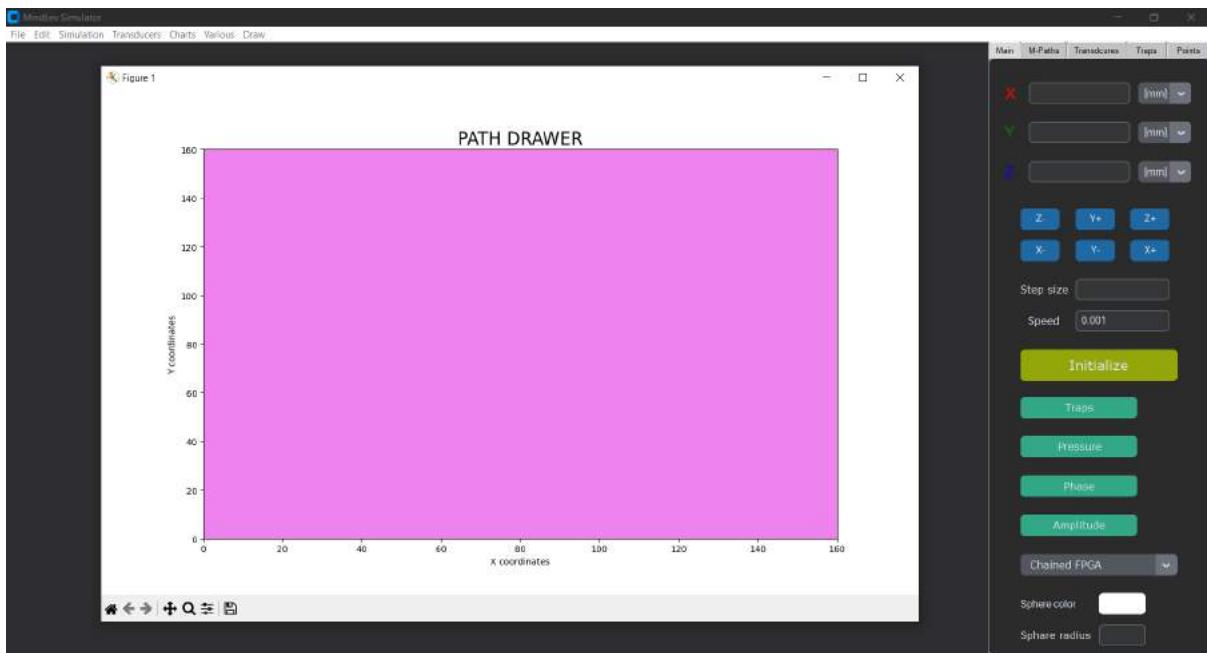
Siatka ta miała służyć jako warstwa odniesienia do warstwy rysowania. Aby ją utworzyć użyliśmy funkcji **meshgrid()** oraz **arange()** z biblioteki **numpy**. Funkcja **arange** pozwala na stworzenie równo oddalonych punktów, określając odległość między nimi, początkowe oraz końcowe położenie. Stosując tą funkcję wzdłuż osi x oraz y dostajemy szereg punktów. Jednak aby dostać siatkę punktów musimy wykonać wszystkie kombinacje między wartościami x i y, w tym właśnie celu posłużyła nam funkcja **meshgrid**. **Program po narysowaniu przez użytkownika ścieżki, miał wyszukiwać najbliższego punktu z siatki punktów, a następnie wyświetlać jego współrzędne w konsoli.**



Rys. 104 Siatka składająca się z punktów rozdzielonych o odległość równą 0,1 jednostek
Źródło: zdjęcie własne

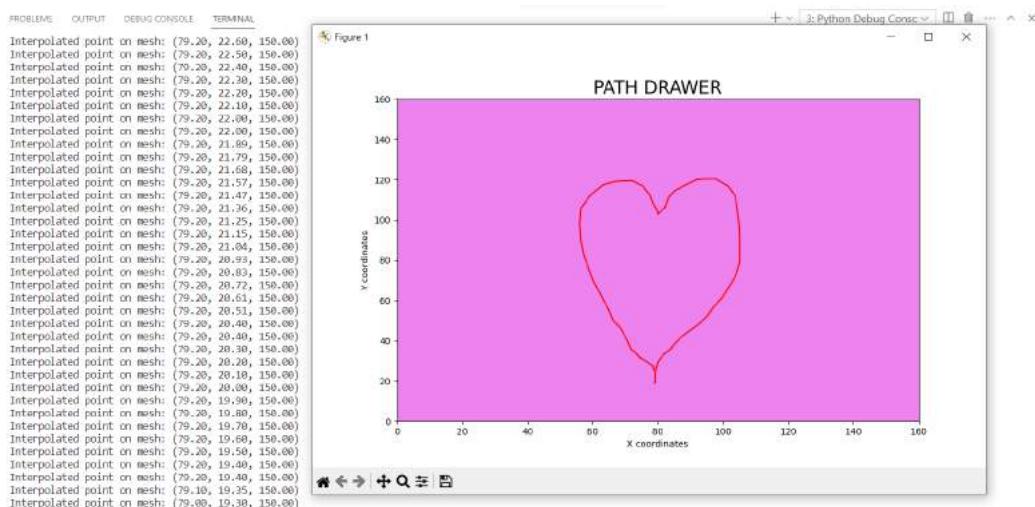
Następnie zaimplementowaliśmy do tego kodu **funkcje zezwalające na rysowanie**, tak aby można było **interaktywnie narysować ścieżkę na wykresie**, klikając i przeciągając myszą. Ścieżka zostanie wtedy poddana interpolacji między punktami siatki przy użyciu **interpolacji liniowej**, a interpolowane współrzędne punktów zostaną wyświetcone i pokazane jako czerwona linia na wykresie.

Gdy interpolacja działała i wyświetlała oczekiwane wartości x, y przeszliśmy do stworzenia docelowej siatki z położeniami odzwierciedlającymi obszar naszego lewitatora. Stosując 100 głośników na jeden układ mamy wzduż osi x oraz y, $10 * 16 \text{ mm} = 160 \text{ mm}$, gdzie 16 mm to promień głośnika, a 10 to numer głośników wzduż boku kwadratu układu. Oprócz tego zmieniliśmy **zakres skali** wykresu (`set_xlim= ([0,160]) , set_ylim= ([0,160])`) i uruchomiliśmy program.



Rys. 105 Sekcja rysowania
Źródło: zdjęcie własne

Po uruchomieniu program był bardzo nieresponsywny. Wiedzieliśmy co było problemem. Tworząc siatkę o wymiarach 160x160 nie przemyśleliśmy wysiłku obliczeniowego programu który musiał stworzyć 2 560 000 punktów i ciągle przeprowadzać obliczenia związane z interpolacją. W początkowym etapie aby uniknąć niepotrzebnych komplikacji zmieniliśmy ilość tworzonych punktów, rozstawiając je w odległości zamiast 0,1 na 0,2. W takim układzie program musiał stworzyć 640 000 punktów co znacznie go odciążyło. Po uruchomieniu program działał bezproblemowo.



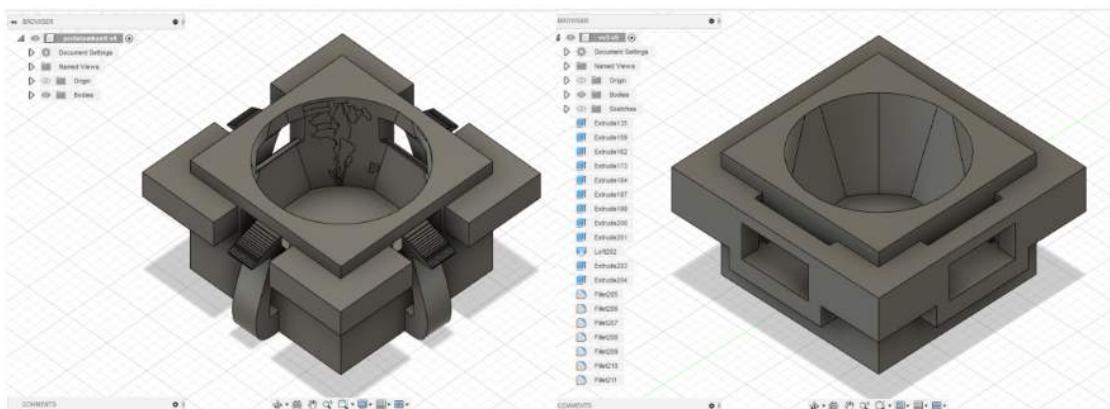
Rys. 106 Wykaz interpolowanych punktów
 Źródło: zdjęcie własne

II.2.7 Obudowa lewitatora

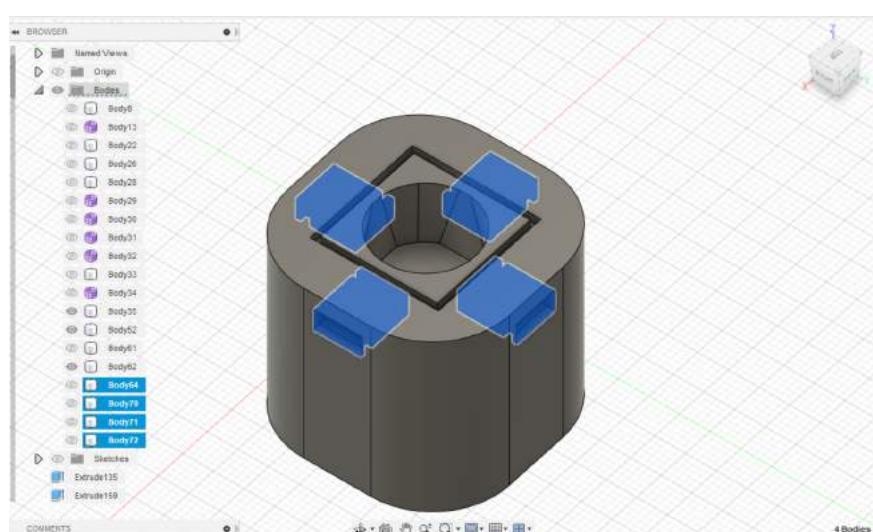
II.2.7.1. Projekt obudowy

W trakcie realizacji finalnego prototypu, przyszedł moment stworzenia obudowy. Wybraliśmy **stworzenie całej drukowanej obudowy od podstaw**.

Pracę rozpoczęliśmy od stworzenia **wzorcowych elementów**, które miały nam posłużyć do **testowania różnych wariantów** drukowanej podstawki pod głośniki. Stworzona przez nas podstawa miała być osobną częścią, która łączy się z obudową. Cała konstrukcja miała być przede wszystkim łatwa do demontażu i niezawodna. Pierwszym wariantem była podstawa, która wchodziłaby do obudowy na zatrzaski (patrz. Rys. 107, po lewej stronie). Jednak po wydrukowaniu wzorca zatrzaski niestety albo **łamały się**, albo **nie było możliwości ich poruszenia**.



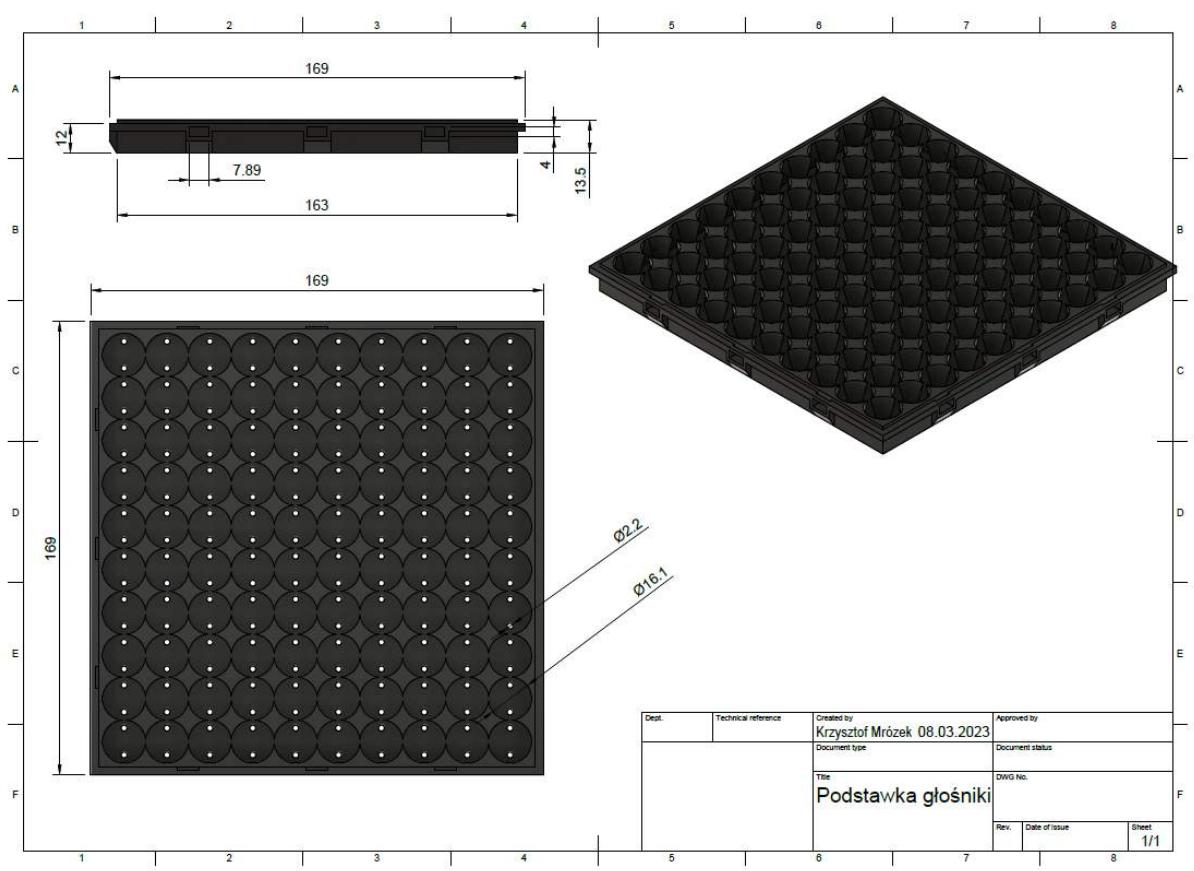
Rys. 107 Wzorcowe druki do testowania mechaniki podstawki z głośnikami
Źródło: zdjęcie własne



Rys. 108 Element wzorcowy
Źródło: zdjęcie własne

Ostatecznie zdecydowaliśmy się na wybór drugiej opcji zawierającej **połączenie klinowe**. Po wydrukowaniu podstawa z głośnikiem po unieruchomieniu jej klinami była bardzo dobrze dopasowana, bez żadnych luzów.

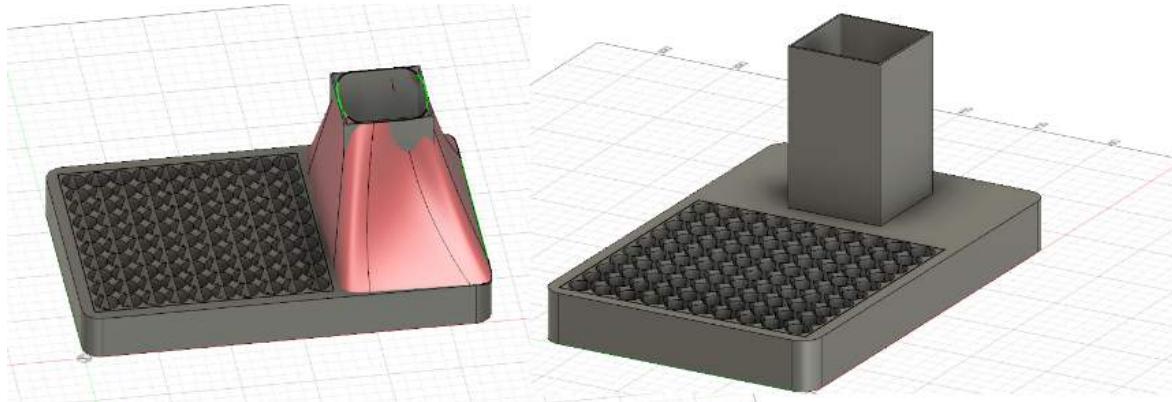
Kliny były zrobione do czoła obudowy, dlatego aby po zamocowaniu można było je również wyjąć. Posiadały one podłużne wycięcia, do których można się było dostać cienkim przedmiotem, dzięki włożeniu go do szczeliny między podstawką z głośnikami a obudową.



Rys. 109 Rysunek techniczny podstawki pod głośniki

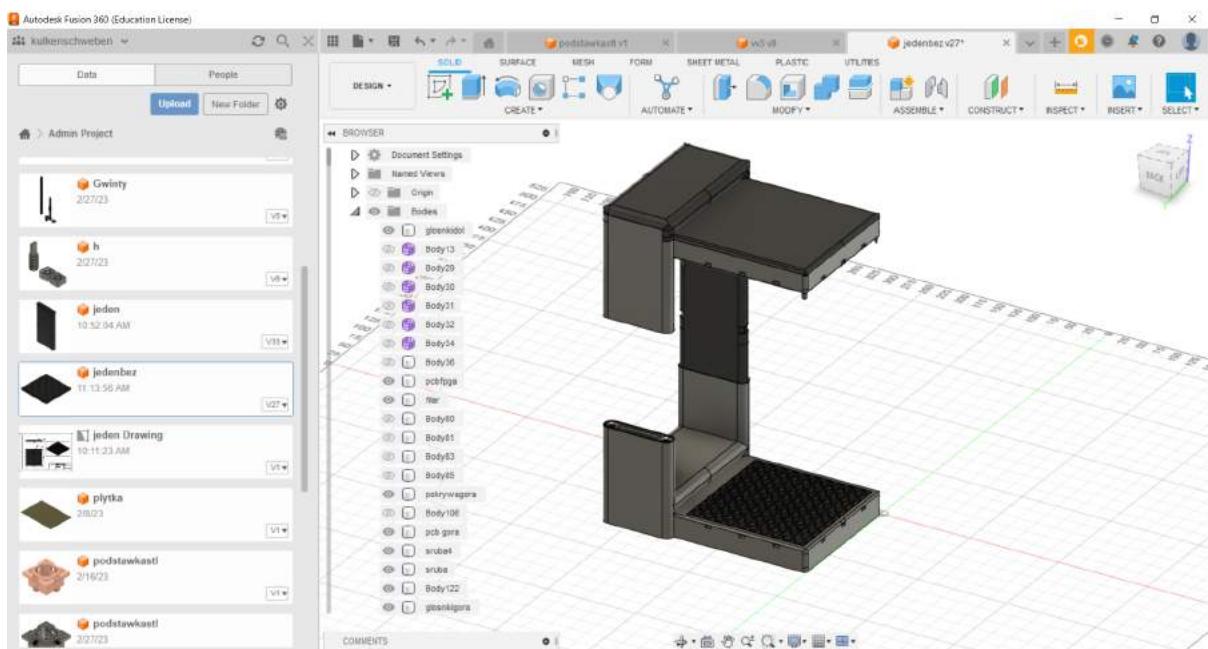
Źródło: zdjęcie własne

Po pozytywnym przetestowaniu połączeń klinowych **podstawkę przeskalowaliśmy tak, aby otrzymać miejsce na 100 głośników** (Rys. 109). Otwory pod kliny były umiejscowione po trzy bocznych stronach podstawki, po trzy na jedną stronę. Taka ilość zapewniała niezawodność konstrukcji oraz bardzo dobrą stabilizację płytka w obudowie.



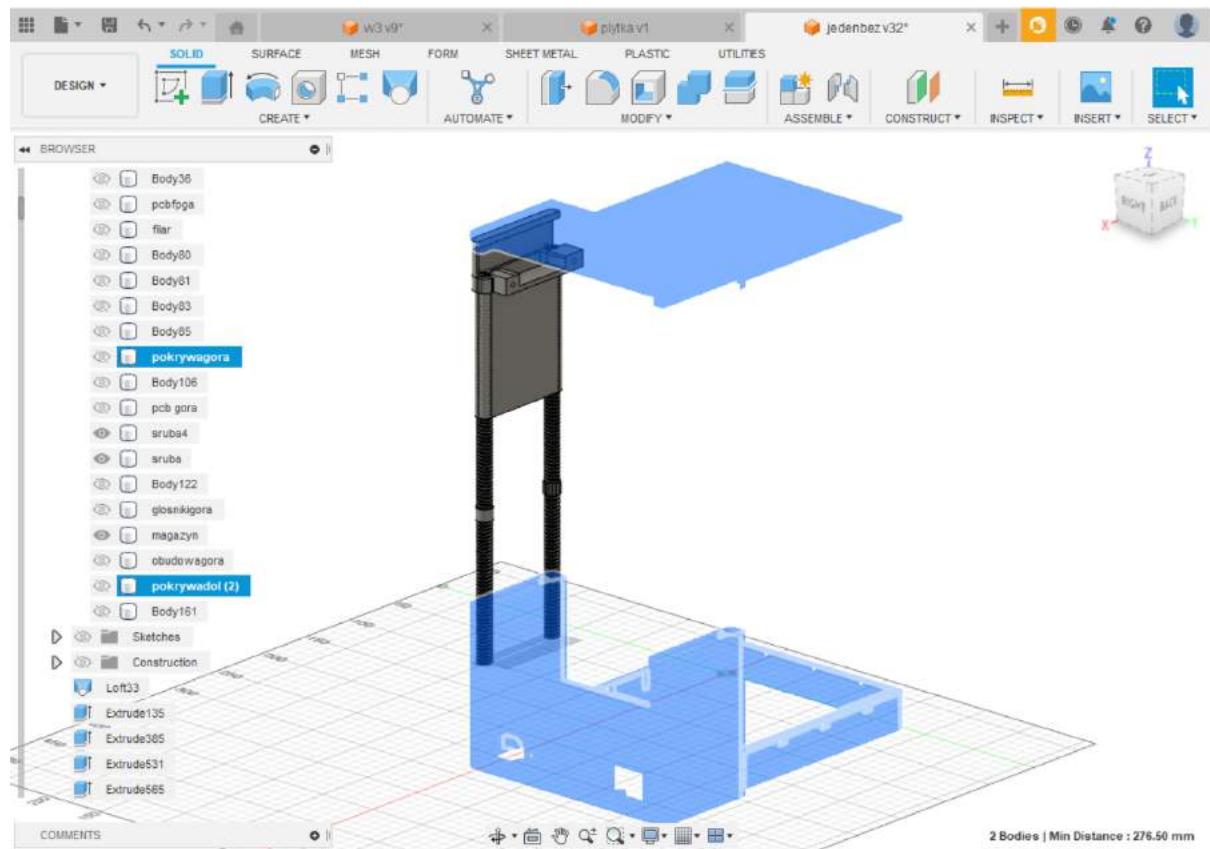
Rys. 110 Różne opcje budowy lewitatora
Źródło: zdjęcie własne

Obudowa miała za zadanie solidnie trzymać górną część lewitatora równolegle do jej dolnej części, w płaszczyźnie poziomej. Równocześnie ważne dla nas było, aby filar łączący dwie podstawy zajmował jak najmniej miejsca, tak aby lewitacja była widoczna od każdej strony.

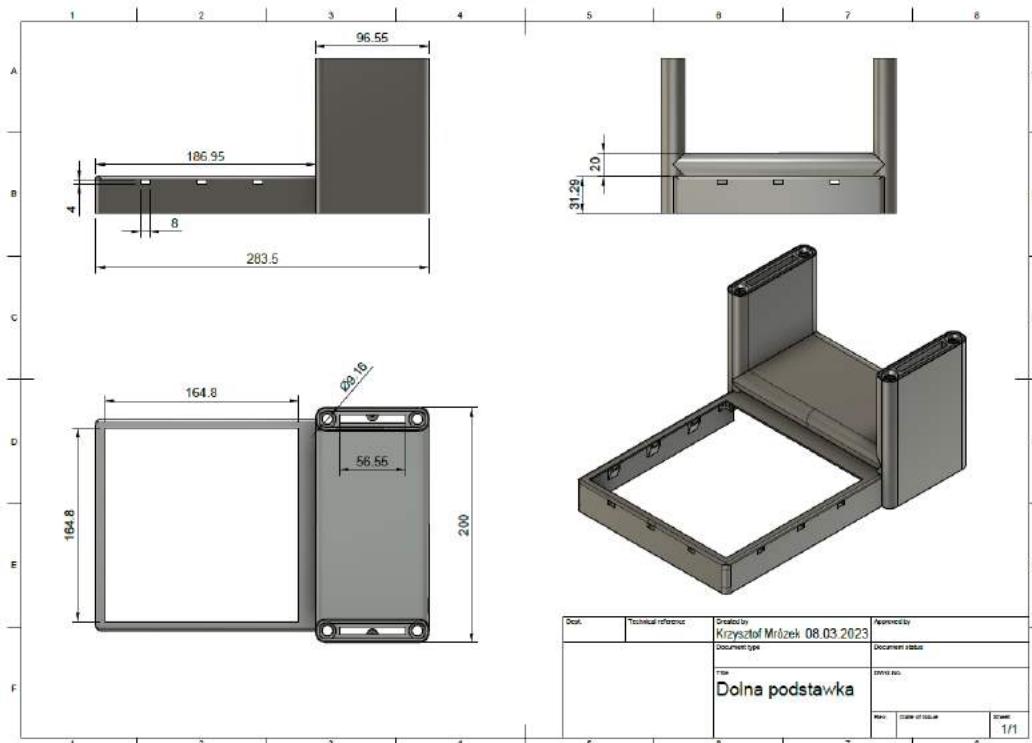


Rys. 111 Projekt obudowy
Źródło: zdjęcie własne

Ostatecznie zdecydowaliśmy się na dwa filary po bokach (Rys. 111). Zapewniały one optymalną stabilność dzięki dużemu rozstawowi śrub. Było ważne, żeby zapewniały również wytrzymałość na moment zginający, dlatego ustawiliśmy je tak, aby był on jak najmniejszy.

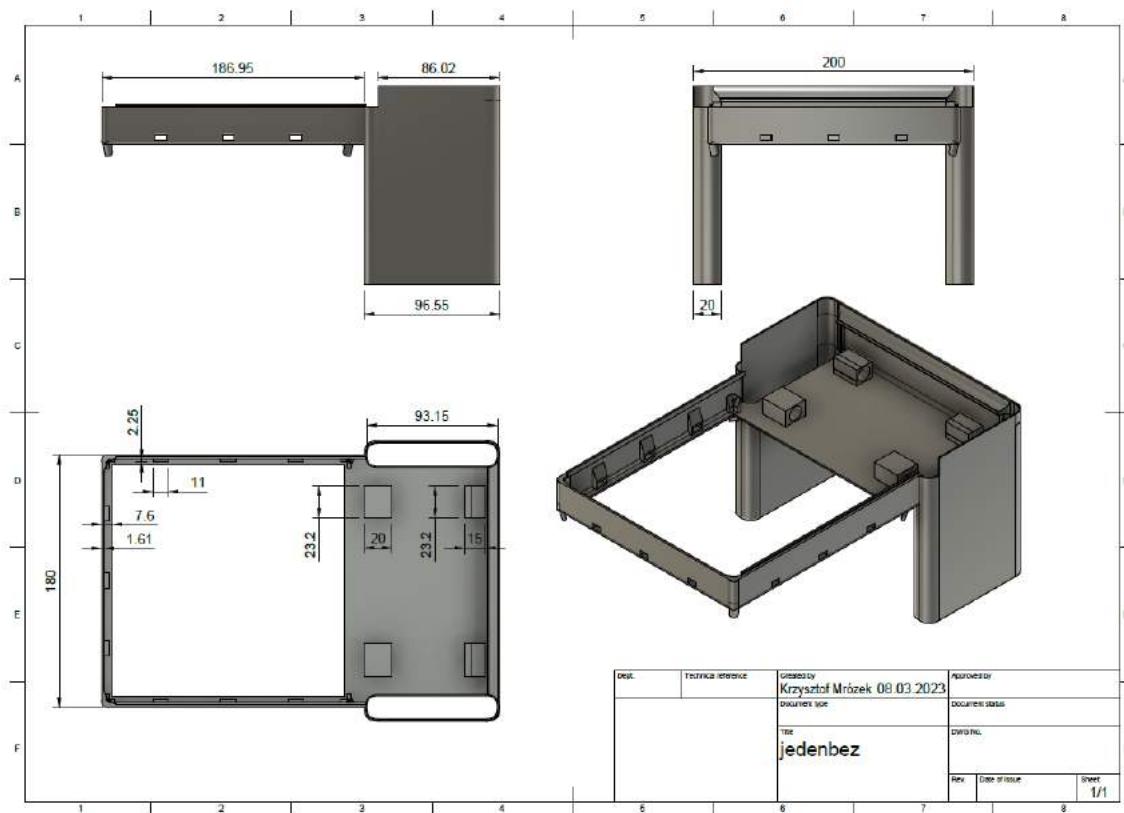


Rys. 112 Mechanizm regulacji odległości między płytami
Źródło: zdjęcie własne



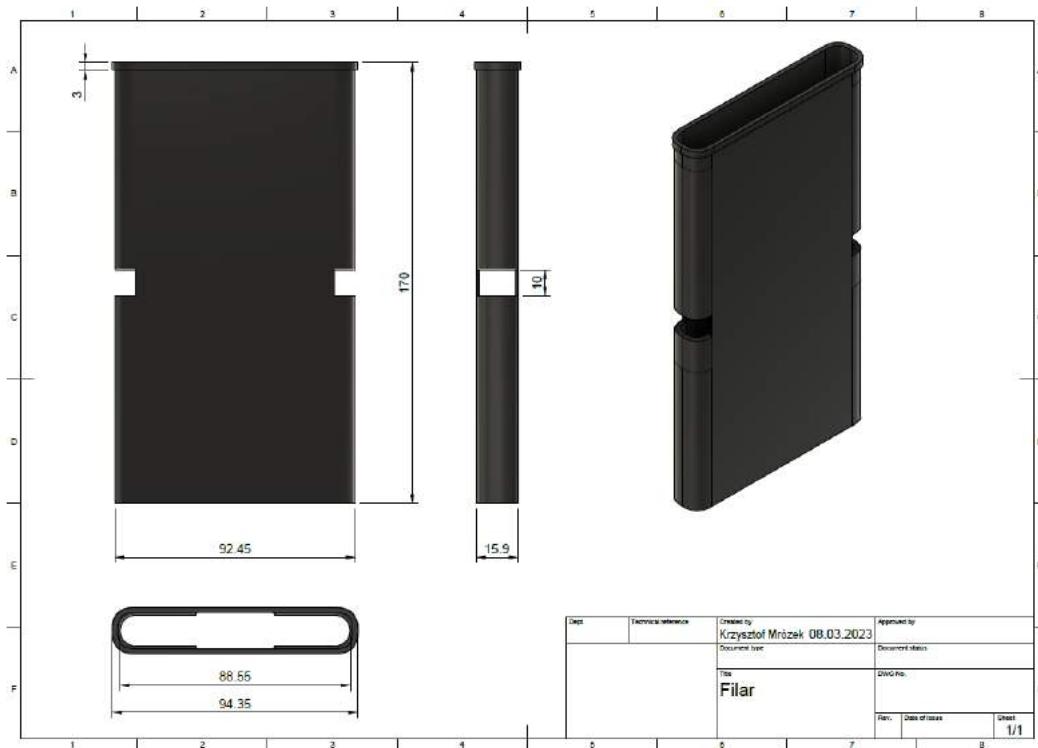
Rys. 113 Rysunek techniczny dolnej podstawki

Źródło: zdjęcie własne



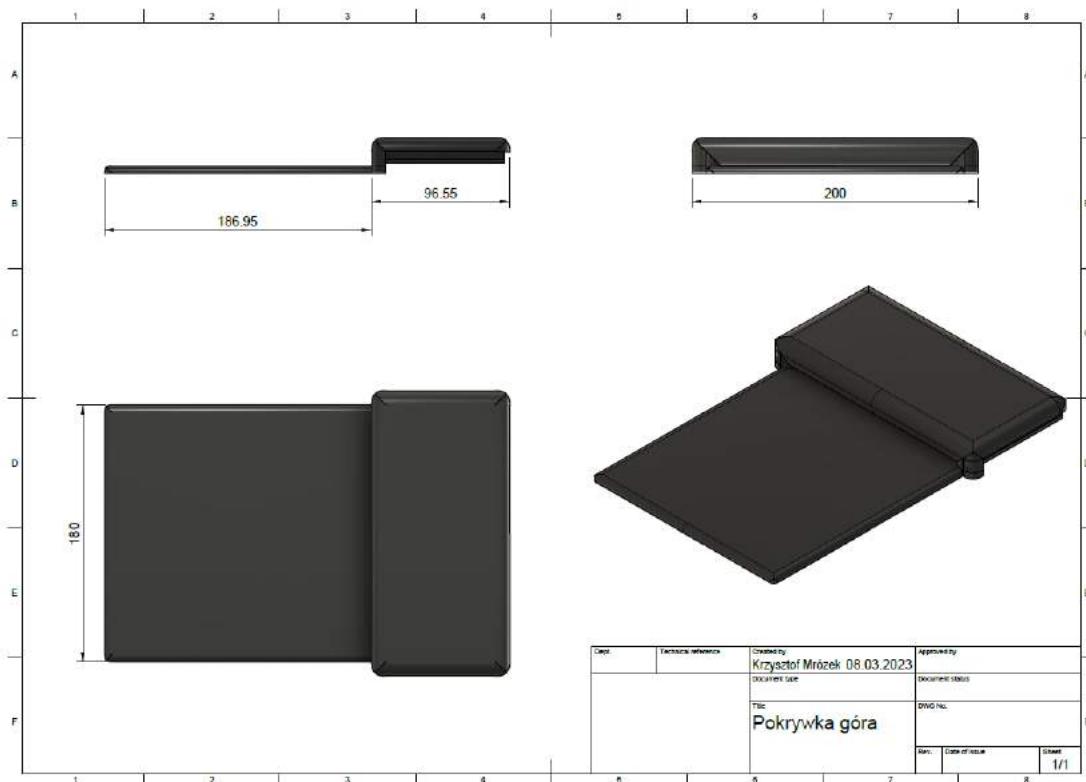
Rys. 114 Rysunek techniczny górnej podstawki

Źródło: zdjęcie własne

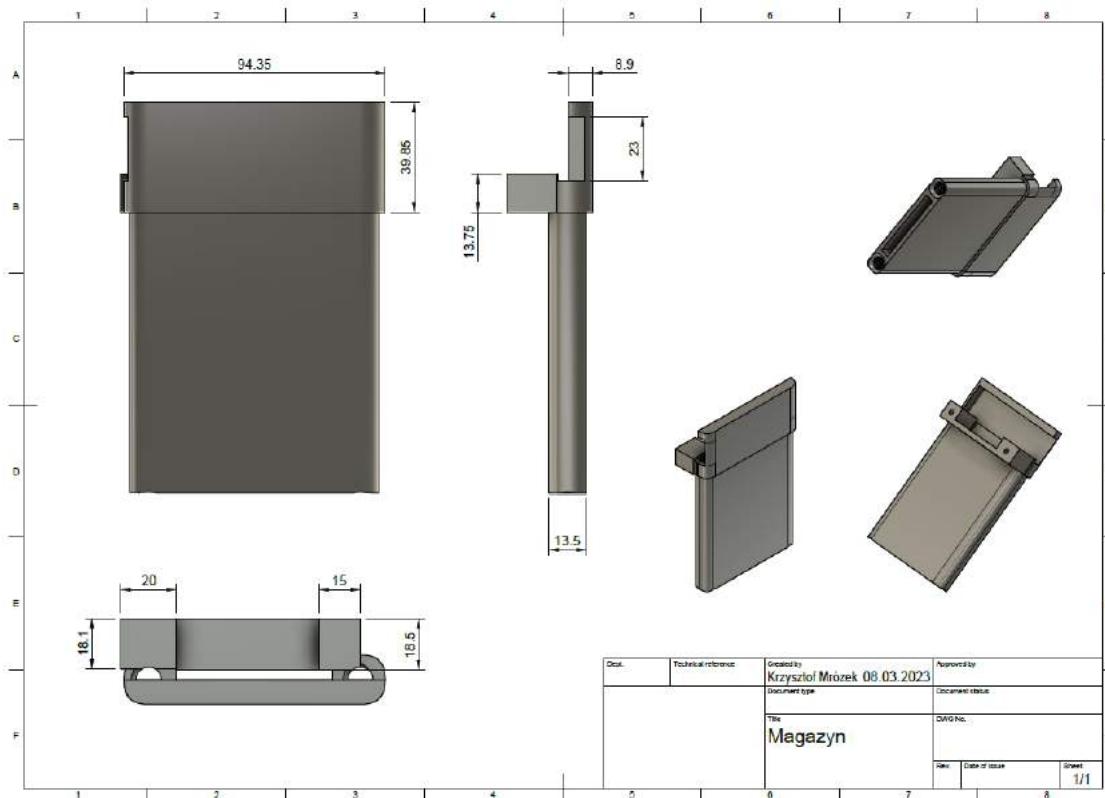


Rys. 115 Rysunek techniczny filarów

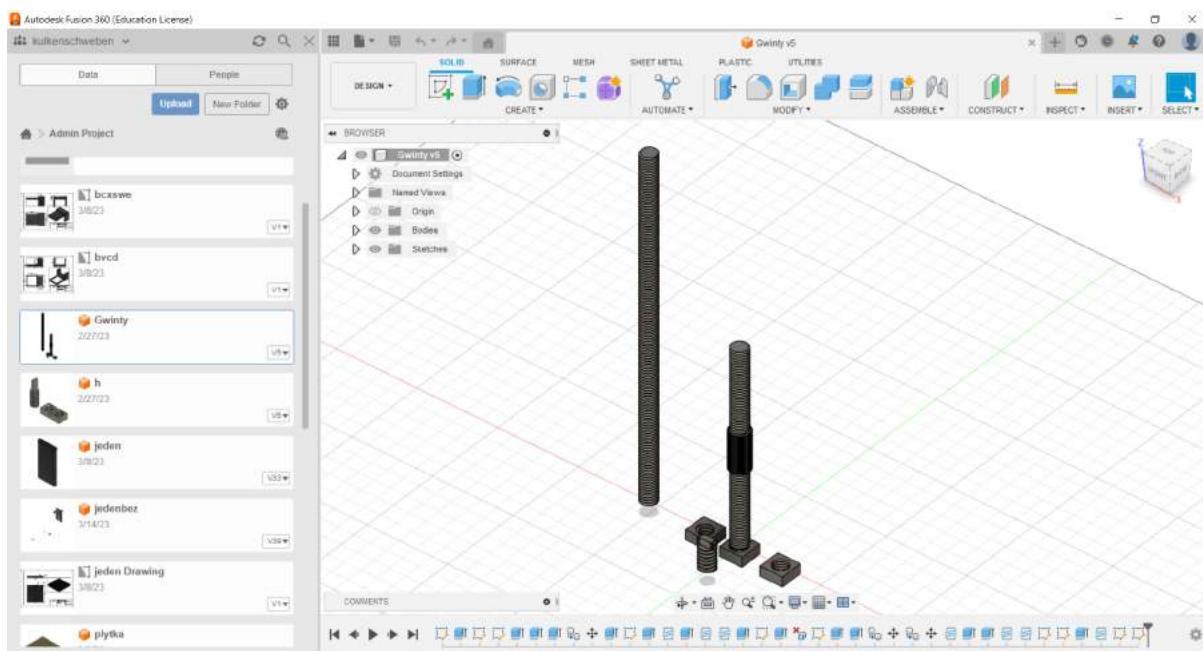
Źródło: zdjęcie własne



Rys. 116 Rysunek techniczny pokrywy górnej
Źródło: zdjęcie własne



Rys. 117 Rysunek techniczny elementu mocującego
Źródło: zdjęcie własne



Rys. 118 Gwinty trapezowe lewe i prawe
 Źródło: zdjęcie własne

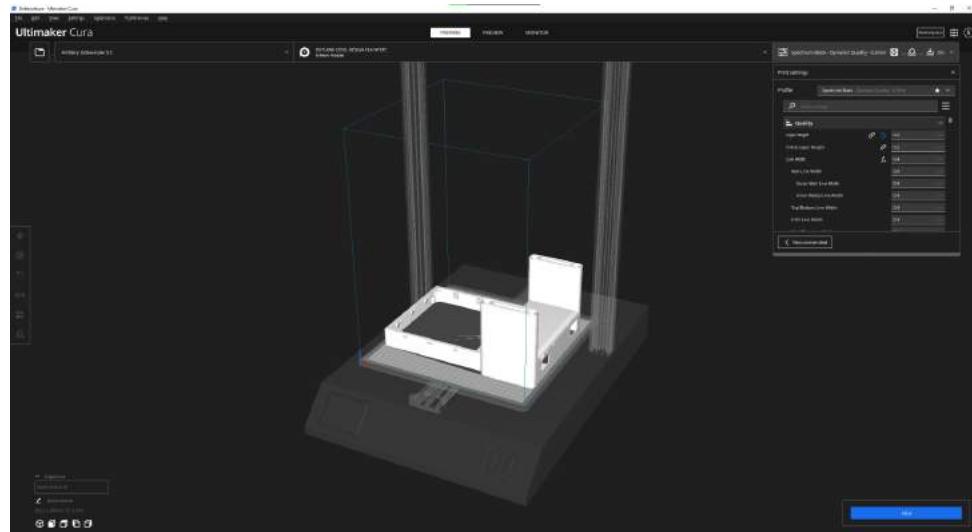
II.2.7.2. Drukowanie obudowy

Po wykonaniu wszystkich projektów technicznych 3D obudowy, przeszliśmy do jej drukowania. Do tego celu użyliśmy drukarki Artillery Sidewinder x2. Po wydrukowaniu kostki kalibracyjnej uzyskaliśmy pożądane parametry do druku. Temperatura drukowania podana od producenta różniła się od faktycznej, która musieliśmy podwyższyć o kilkanaście stopni. Finalne parametry drukowania przedstawia Rys. 119.

<i>Printing Temperature</i>	f_x	220.0	°C
<i>Printing Temperature Initial Layer</i>	f_x	220.0	°C
<i>Initial Printing Temperature</i>		220.0	°C
<i>Final Printing Temperature</i>		220.0	°C
<i>Build Plate Temperature</i>	\varnothing	60.0	°C
<i>Build Plate Temperature Initial Layer</i>	\varnothing	80.0	°C
<i>Flow</i>		100.0	%
<i>Wall Flow</i>		100.0	%
<i>Outer Wall Flow</i>		100.0	%

Rys. 119 Parametry drukowania
 Źródło: zdjęcie własne

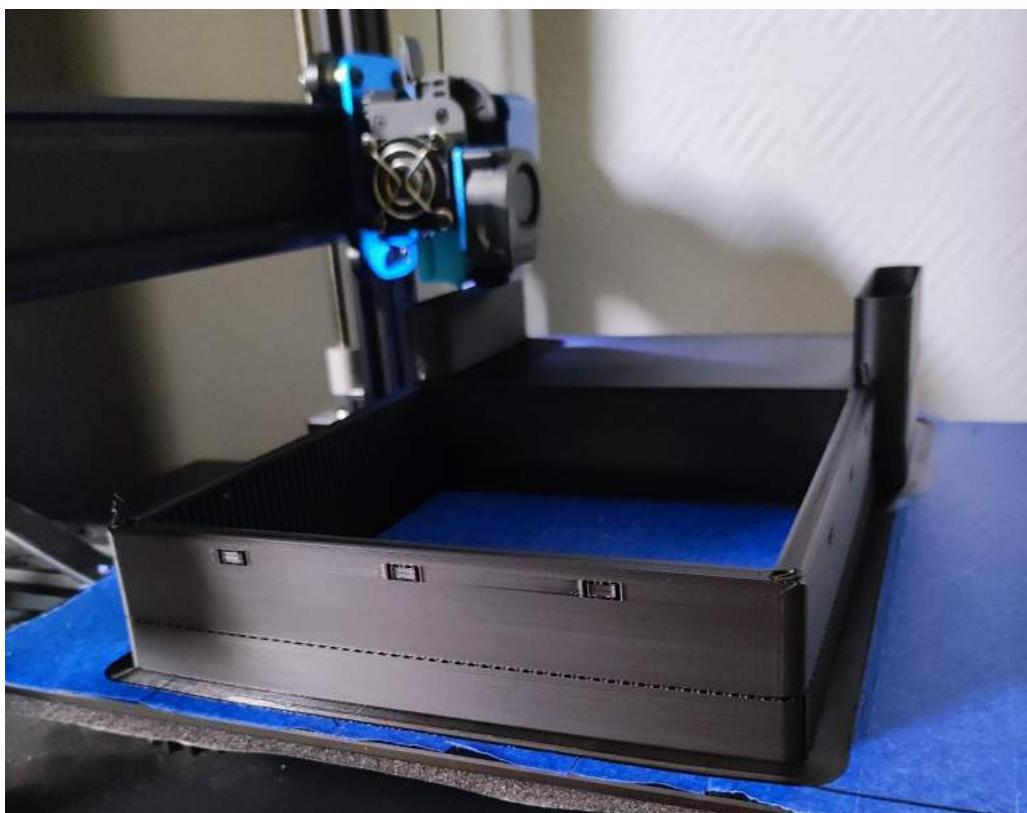
Pierwszym elementem przeznaczonym do druku była dolna część obudowy (Rys. 120).



Rys. 120 Ultimaker Cura - podgląd do wydruku

Źródło: zdjęcie własne

Stół drukarki w celu zachowania czystości i co ważniejsze - uzyskania lepszej przyczepności filamentu pokryliśmy taśmą malarską. Jest to sprawdzony sposób zapobiegania chwiejności i poruszania się drukowanego elementu po stole drukarki.



Rys. 121 Drukowanie dolnej części obudowy

Źródło: zdjęcie własne

Proces drukowania tego elementu zajął około 21 godzin. Szybkość drukowania ustanowiliśmy na 60 mm/s.

Po zakończonym procesie drukowania, oczyściliśmy druk, usuwając supporty. Część ta prezentowała się jak na Rys. 122.



Rys. 122 Wydrukowana dolna część obudowy
Źródło: zdjęcie własne

Element ten był bardzo stabilny przy grubość ściany jedynie 1,5 mm. Poddaliśmy go testom wytrzymałościowym, które nie wykazały żadnych znaczących wad.

Następnie przeszliśmy do drukowania kolejnych elementów. Czas ich wykonania wyniósł ok. 5 dni licząc wszystkie nieudane elementy oraz czas potrzebny na przygotowanie stołu pod następny druk.



Rys. 123 Pokrywka do dolnej części obudowy
Źródło: zdjęcie własne

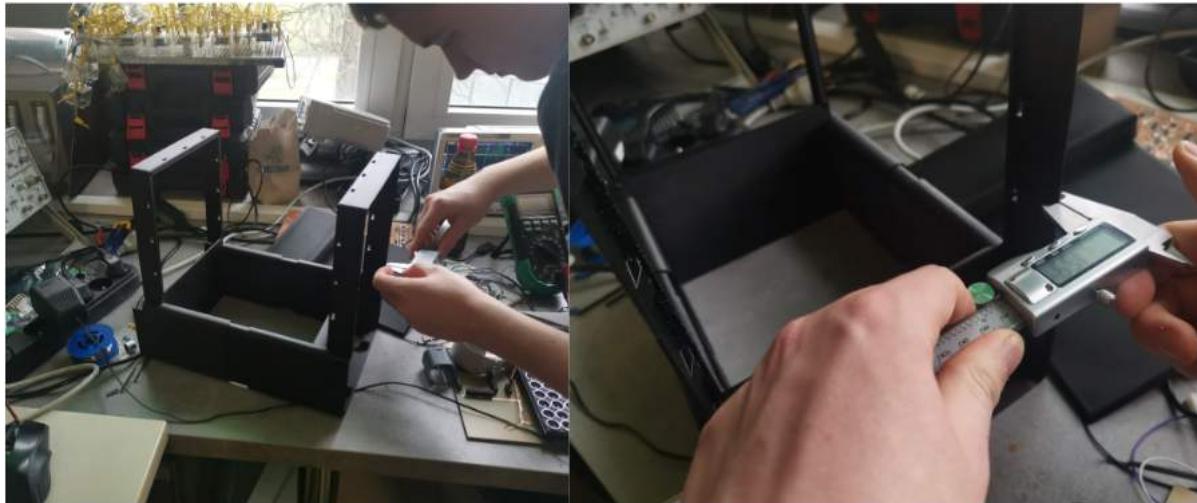


Rys. 124 Filary oraz górną pokrywa obudowy
Źródło: zdjęcie własne

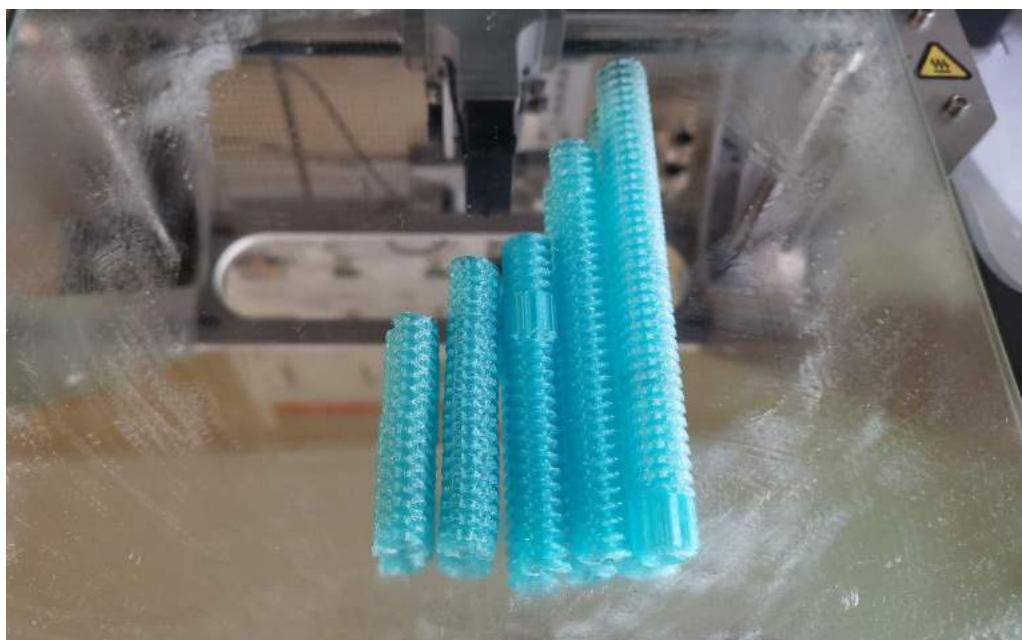


Rys. 125 Dolna i górnna część obudowy wraz z zamontowanymi filarami
Źródło: zdjęcie własne

W czasie drukowania elementów wykonywaliśmy również ich pomiar kontrolny m.in. czy aby na pewno każdy element posiadał odpowiednie wymiary. Wszystkie wydrukowane przez nas części mieściły się w granicach tolerancji ustalonej w trakcie ich projektowania.



Rys. 126 Pomiar grubości ścian podstawy obudowy
Źródło: zdjęcie własne



Rys. 127 Śruby trapezowe z PET-G
Źródło: zdjęcie własne

Całość obudowy miała być regulowana śrubami trapezowymi (patrz Rys. 128), umieszczonymi w filarach. Po przeprowadzonych **testach wytrzymałościowych** śruby uległy jednak złamaniu. Postanowiliśmy wydrukować takie same, lecz z minimalnym otworem w ich środku, gdyż śruba tak wykonana posiada większą odporność na moment zginający, przez co jest bardziej wytrzymała.

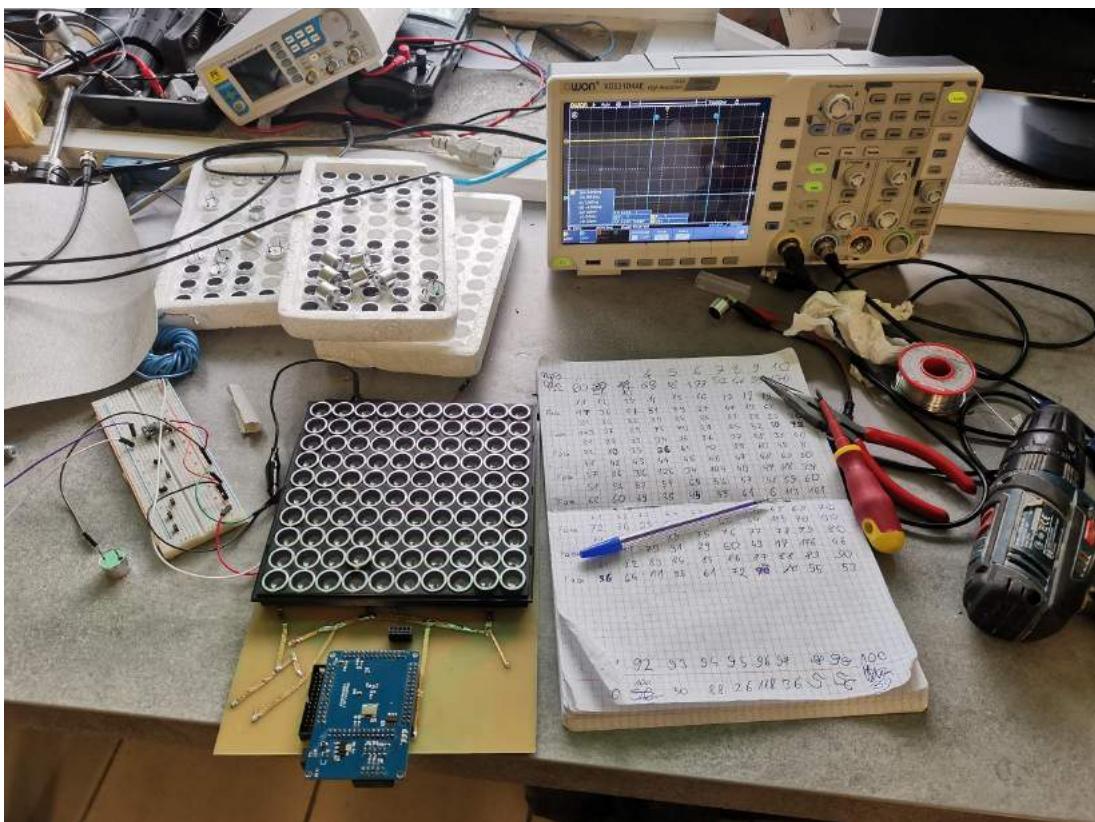
II.2.8. Złożenie finalnego prototypu

Po przygotowaniu wszystkich części nadszedł czas złożenia końcowego prototypu urządzenia. Pierwszym zadaniem było oznaczenie polaryzacji głośników. Przy użyciu oscyloskopu cyfrowego sprawdzaliśmy po kolej i każdy z głośników (Rys. 129) oraz zadając sygnał 40 kHz na mikrokontroler mierzyliśmy fabryczne przesunięcie fazowe głośnika. Kompensacja fazy w przypadku wysłania na głośniki przesunięć fazowych jest istotna, gdyż może ona mieć znaczący wpływ na właściwą interferencję fal. Oscyloskop pokazywał przesunięcie między sygnałami w stopniach, które zapisywaliśmy na obudowie głośników.



Rys. 128 Sprawdzanie polaryzacji oraz wewnętrznych przesunięć fazowych głośników
Źródło: zdjęcie własne

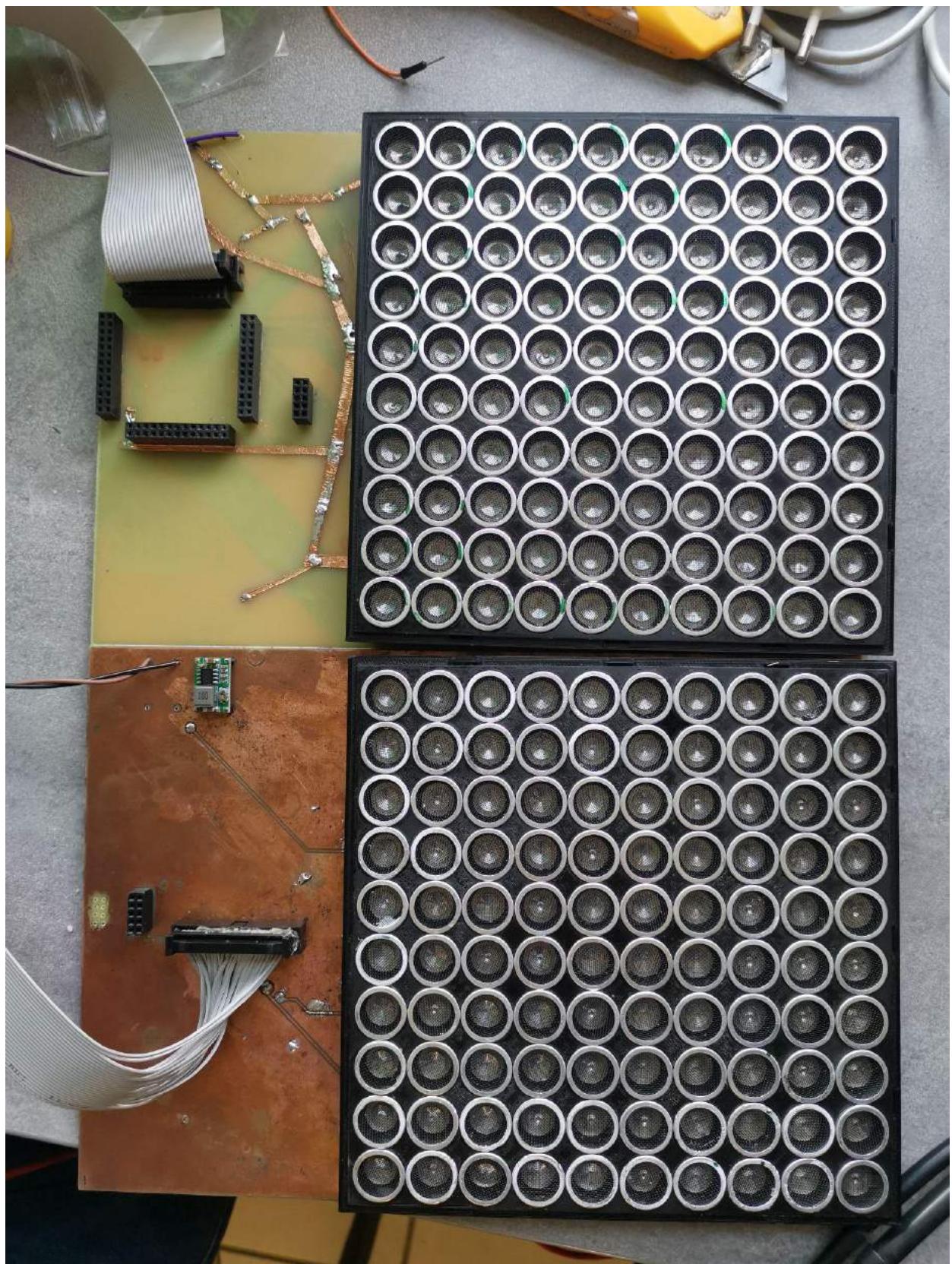
Po oznaczeniu nóżek głośników zaczęliśmy je wkładać stroną z oznaczoną nóżką. Po montażu głośniki zlutowaliśmy. Proces powtórzyliśmy dla drugiej płytki.



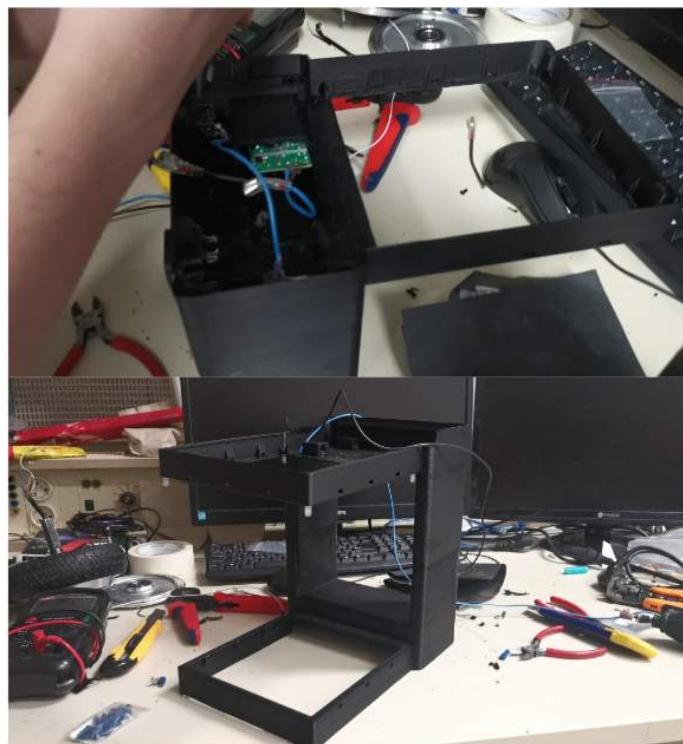
Rys. 129 Głośniki zamontowane w płytce PCB wraz z podłączonym mikrokontrolerem FPGA
Źródło: zdjęcie własne

Do FPGA (AS-SERIAL) wgraliśmy uprzednio przygotowany program i zaczęliśmy testować sygnały z niego odbierane. Prawie wszystkie głośniki emitowały poprawny sygnał o wielkości 40 kHz. Te które nie wysyłały nic skontrolowaliśmy. W części przypadku była to wina spalonych układów tc4427, a w części były one źle przypisane programowo w Quartusie. Parę z nich miało źle połączenie z masą, przez co sygnał pojawiał się na dwóch nóżkach. Po analizie usterek i ich naprawie wszystkie głośniki były w pełni sprawne.

Przygotowaliśmy program napisany w Pythonie do wysyłania przesunięć fazowych przez USB-UART do FPGA i rozpoczęliśmy bardziej dokładną kompensację faz. Po kalibracji otrzymaliśmy płytkę z wszystkimi głośnikami nadającymi sygnał w tej samej fazie. Proces powtórzyliśmy dla drugiej płytki, a następnie przeszliśmy do składania obudowy (Rys. 131).

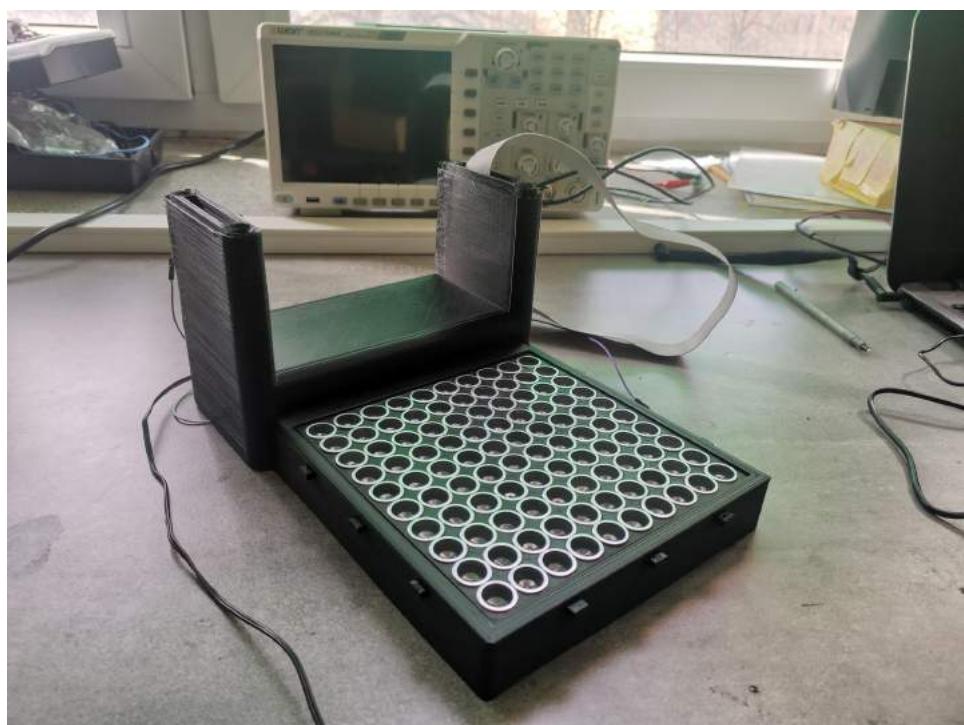


Rys. 130 Płytki z zamontowanymi głośnikami
Źródło: zdjęcie własne

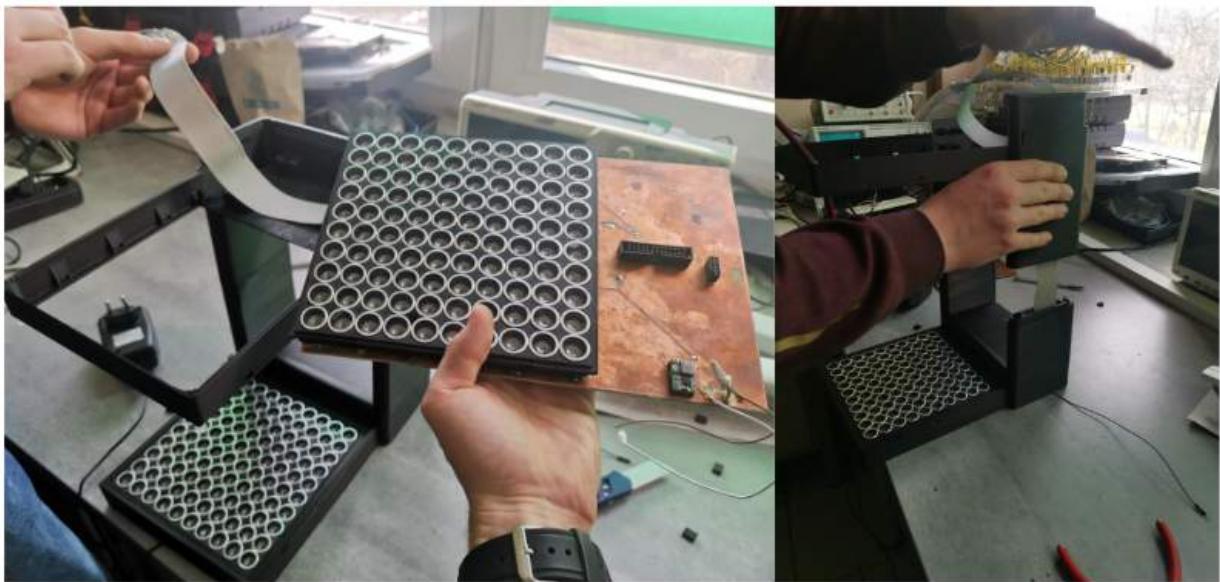


Rys. 131 Montaż zasilania 230/12 V AC-DC
Źródło: zdjęcie własne

Nie wszystkie elementy mieściły się zgodnie z projektem. Rozwiążanie jednak okazało się proste, gdyż wystarczyło problematyczne miejsca delikatnie przeszlifować.



Rys. 132 Montaż płytka z głośnikami do dolnej części obudowy
Źródło: zdjęcie własne



Rys. 133 Połączenie obudowy
Źródło: zdjęcie własne



Rys. 134 Obudowa po umieszczeniu wszystkich elementów
Źródło: zdjęcie własne

II.2.9. Uruchomienie urządzenia

Po złożeniu całego prototypu w całość mogliśmy zacząć testować jego możliwości. Uruchomiliśmy naszą aplikację sterowania lewitacji i zaczęliśmy sprawdzać poprawność kodu obliczającego przesunięcia.



Rys. 135 Sprawdzanie poprawności współrzędnych obiektu lewitującego
Źródło: zdjęcie własne

Położenie obiektu w przestrzeni było bardzo zbliżone do tego w symulatorze (patrz: Rys. 135). Obiekt unosił się w punkcie wcześniej ustalonym w programie.



Rys. 136 Uruchomienie lewitatora
Źródło: zdjęcie własne

Lewitator zgodnie z zamysłem był w stanie **dowolnie zmieniać położenie obiektu**. Lewitujący przedmiot jest w stanie poruszać się z prędkością ograniczoną przez szybkość komunikacji i odległość pomiędzy dwoma następującymi położeniami. Przesył danych pomiędzy Pythonem a FPGA wynosi **921600 bauds = 155200 bajtów/s**. Aby raz zmienić opóźnienie fazowe na wszystkich głośnikach trzeba wysłać 400 liczb - 200 numerów głośników i 200 przypisanych do nich opóźnień, co daje 400 bajtów. Możemy więc zmienić stan wszystkich głośników $155200/400 = 388$ razy na sekundę. Przy kroku wielkości 1 mm daje to prawie **39 cm/s**.

II.2.10. Udoskonalony wzór na obliczanie przesunięć fazowych

W trakcie pierwszych testów nowszej wersji prototypu, mieliśmy pewne **zarzuty do kodu obliczającego przesunięcia fazowe**. Towarzyszył on nam od chwili stworzenia pierwszego prototypu, lecz podczas jego użytkowania punkty skupienia lewitacji były bardzo słabe.

Zaczęliśmy testować metodą prób i błędów różne warianty naszego kodu realizującego obliczanie przesunięć. Niektóre w ogóle nie umożliwiały lewitacji, z kolei inne działały jeszcze gorzej od obecnego kodu. Podczas jednej sesji testowania uświadomiliśmy sobie popełniony błąd. Kod faktycznie obliczał przesunięcie, lecz nie uwzględniał on wymogu względnej różnicy 180 stopni między emitowanymi falami. Sytuacja taka sprawia, że fala emitowana z głośników z jednego z układów ma większą drogę do pokonania aby spotkać się w przeciwfazie z drugą falą.

```

class SimPoints:
    def __init__(self):
        self.transducers = []

    def addTransducer(self, pt, isOnTop):
        self.transducers.append(Transducer(pt, isOnTop))

    def calculatePhase(self, obj_pos):
        global PIEZO_FREQ
        WAVE_C = 343000
        PIEZO_FREQ = 40000
        WAVE_K = 2*math.pi*PIEZO_FREQ/WAVE_C

        for t in range(len(self.transducers)):
            delta = obj_pos - self.transducers[t].pt
            r = delta.length()
            s = -r*WAVE_K
            if self.transducers[t].isOnTop:
                s += math.pi

            self.transducers[t].focusedPhase = math.fmod(s, 2*math.pi) + 2*math.pi

```

Rys. 137 główny kod realizujący ulepszone obliczanie przesunięć fazowych

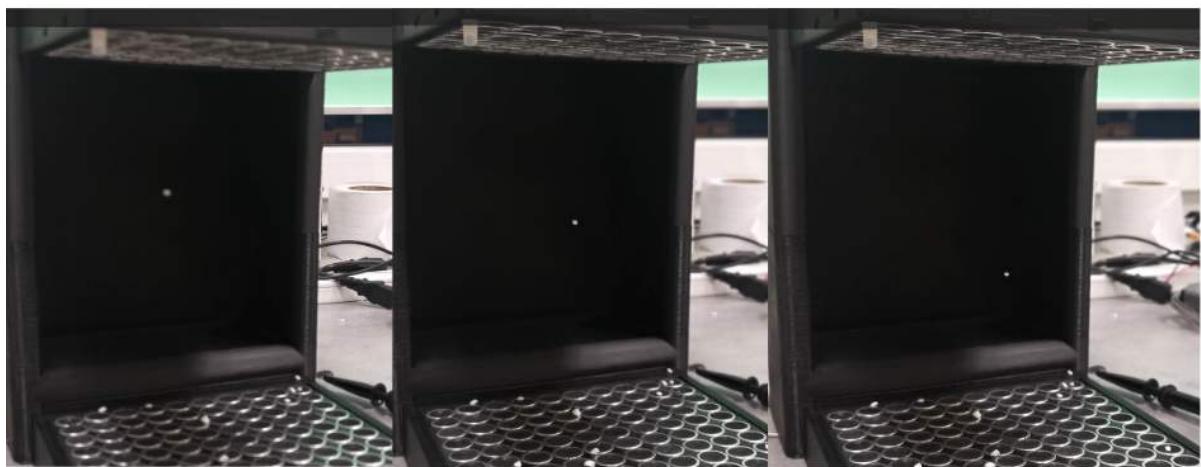
Źródło: opracowanie własne

Widzimy **modyfikację uprzedniego kodu**. Funkcja calculatePhases posiada stałe zmienne związane z prędkością dźwięku w powietrzu (343 m/s) oraz częstotliwością rezonansową głośników 40 kHz. Poniżej widzimy pętlę for, która oblicza różnicę odległości między głośnikiem a punktem skupienia i następnie zamienia ją na wynik w mikrosekundach. Na koniec funkcja sprawdza parametr isOnTop. Jeżeli jego wartość jest prawdą dodaje ona 180 stopni do wyniku, jeżeli jego wartością jest fałszem zostawia ona obliczoną wartość bez zmian.

Po zaimplementowaniu odpowiednich zmian, właściwości lewitacji znaczaco wzrosły. Dało to później podstawy do realizacji bardziej dynamicznych ruchów.

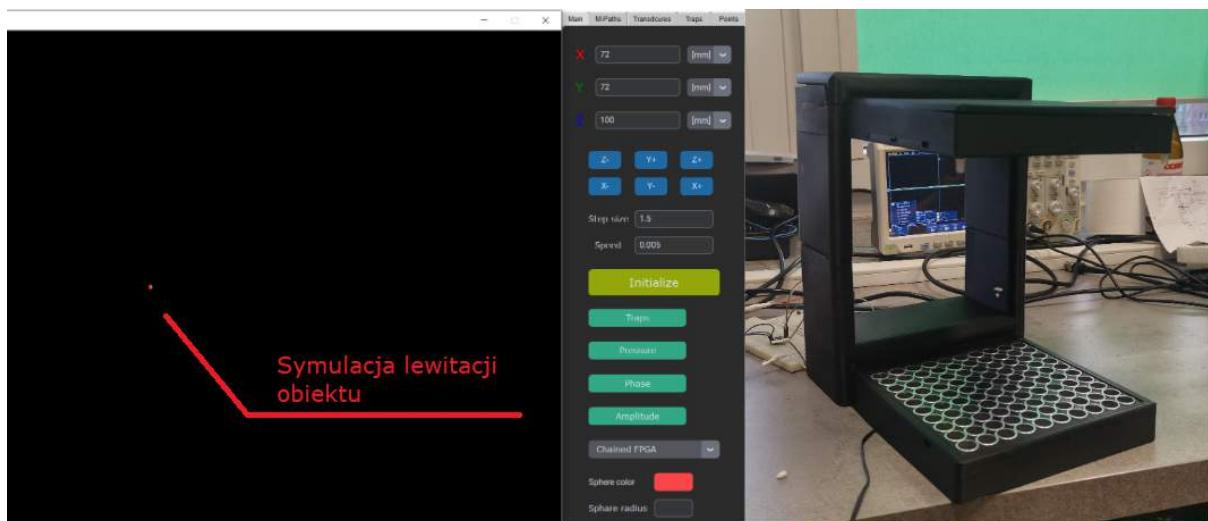
II.2.11. Testowanie działania

Po uruchomieniu aplikacji oraz podłączeniu urządzenia FuturFlow do zasilania, rozpoczęliśmy proces **testowania jego możliwości**. Pierwszy zadany ruch przedmiotu wykonywał poruszanie po trajektorii liniowej ukośnej, ze współrzędnymi początkowymi (72,72,100) a końcowymi (120,120,150) (patrz Rys. 138).



Rys. 138 Zmiana położenia obiektu w czasie podczas ruchu po linii ukośnej
Źródło: opracowanie własne

Następnie przeszliśmy do **testowania sterowania manualnego**. Najpierw uruchomiliśmy naszą aplikację, potem zainicjalizowaliśmy ruch początkowy, włączycyliśmy symulację lewitacji, naciskaliśmy odpowiednie przyciski w panelu sterowania 3D i obserwowaliśmy zarówno zmianę położenia w urządzeniu jak i na symulacji programowej.

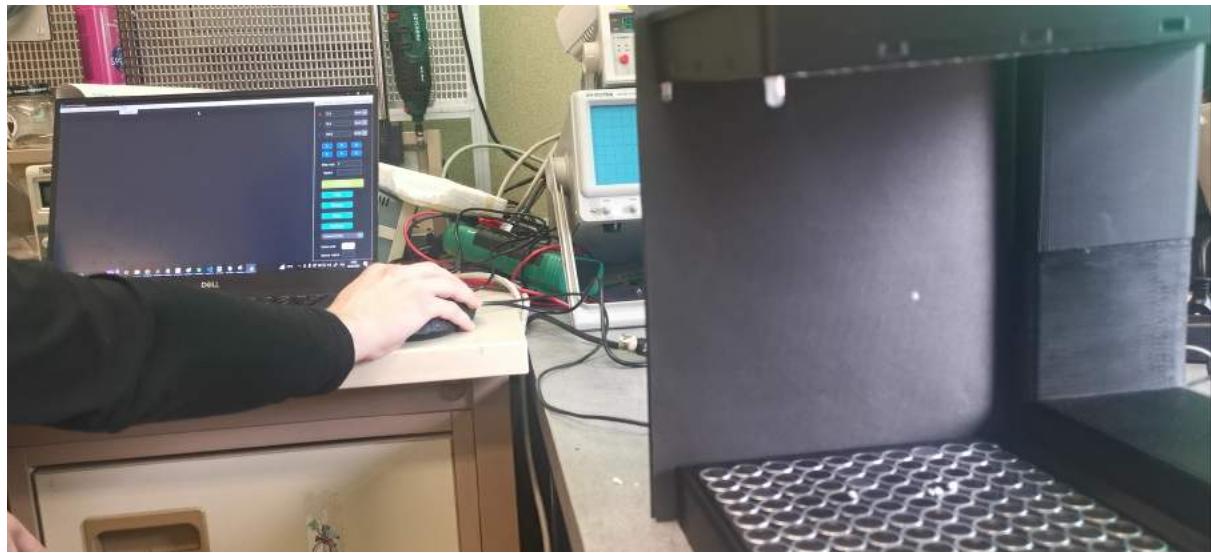


Rys. 139 Zmiana położenia symulowanego obiektu na żywo
Źródło: opracowanie własne

Uruchamiając po kolej ka d g   cie k  ruchu, obserwowali my tworzone  cie ki. Zaobserwowa my,  e w sytuacji gdy wielko  kroku by a za wielka obiekt wypada  z trajektorii.

Po przetestowaniu ka dej z mo liwych  cie ek, postanowili my sprawdzić dzia anie panelu r cznego sterowania lewitacj . Po uruchomieniu w miejsca oznaczone x, y, z, wpisywana zosta a warto  aktualnej pozycji punktu skupienia. Obiekt umieszczony

w tym miejscu z łatwością można było sterować w osiach 3D, również po zwiększeniu parametru step size aż do nawet 3 mm na krok.



Rys. 140 Manualne sterowanie położeniem obiektu w 3D

Źródło: zdjęcie własne

Wykonując manualny ruch postanowiliśmy sprawdzić możliwy zakres lewitacji obiektów. Po eksperimentalnych badaniach wynikało, że lewitacja wciąż zachodziła dla punktów odległych o 1 cm od najdalej wysuniętego układu z głośnikami (Rys. 141).

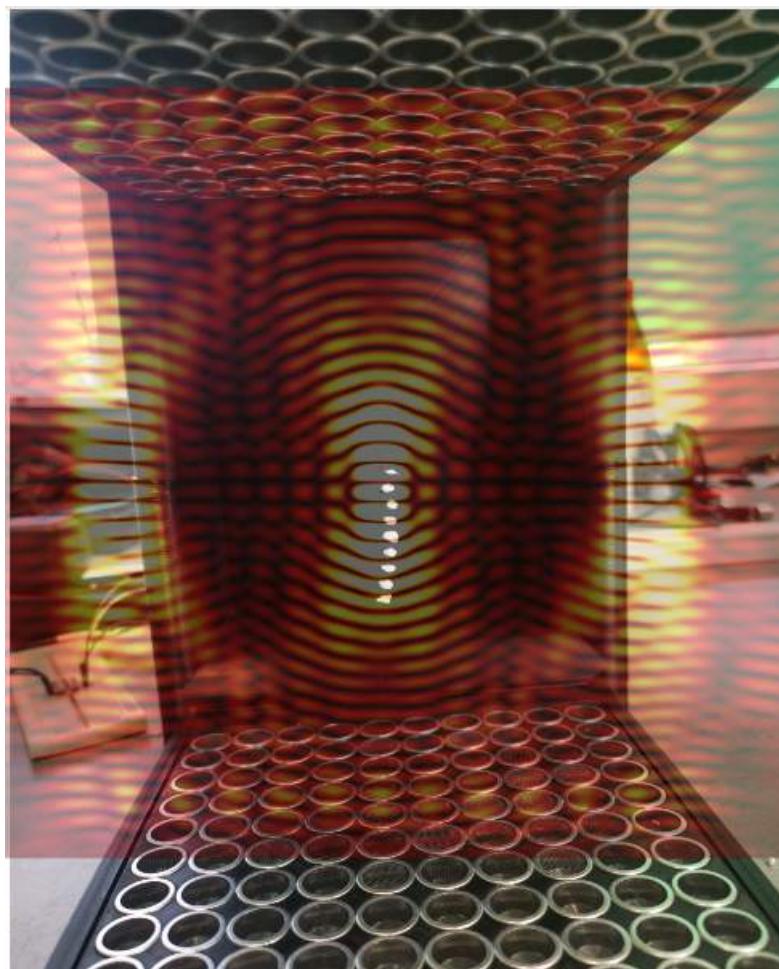


Rys. 141 Krańcowe położenie unoszonego obiektu

Źródło: zdjęcie własne

W opcjach dostępnych w programie dostępna jest jeszcze jedna ścieżka, która wymagała sprawdzenie poprawności działania. Było to najważniejsza opcja pozwalająca użytkownikowi na określenie własnej, unikatowej trajektorii ruchu. Byliśmy zaskoczeni efektem jaki uzyskaliśmy po jego uruchomieniu. **Obiekt faktycznie poruszał się z dokładnością do paru milimetrów po wyznaczonej trajektorii.**

Aby lepiej przedstawić zasadę tworzenia fali akustycznej stworzyliśmy specjalną grafikę obrazującą działanie pola akustycznego w lewitatorze.



Rys. 142 Rozkład natężenia pola akustycznego w urządzeniu dla współrzędnych środka
źródła: opracowanie własne

II.2.12. Instrukcja obsługi prototypu

1. Po podłączeniu urządzenia do zasilania podłączyć interfejs USB-UART do portu szeregowego komputera.
2. Następnie uruchomić aplikację **FuturFlow** prosto z kodu źródłowego w Pythonie.
3. Aby umożliwić jakikolwiek ruch, należy wpisać wartość prędkości ruchu inicjalizującego i nacisnąć zielony przycisk o nazwie “Initialize”. Punkt skupienia powinien znajdować się pośrodku lewitatora tj. we współrzędnych (80, 80, 100).
4. Za pomocą przyrządu podawczego nakładamy uprzednio przygotowany obiekt do lewitacji. Świetnie do tego sprawdzają się kulki styropianowe o niewielkiej średnicy i możliwie sferoidalnym kształcie.
5. Przechodząc w zakładkę M-Paths, do dyspozycji jest kilka trajektorii ruchu. Po wyborze jednej z nich, wpisać odpowiednie parametry ruchu i obserwować efekty na urządzeniu. Istnieje możliwość zmiany prędkości lewitacji poprzez zmianę jej domyślnej wartości w oknie wyboru.
6. W górnej zakładce oznaczonej “Draw”, dostępna jest sekcja rysowania trajektorii ruchu przez użytkownika. Ruch ten tworzyć się będzie o stałej współrzędnej “z” wynoszącej 100 mm oraz będzie się zmieniać zgodnie z narysowaną ścieżką w przestrzeni 2D.
7. Aby zatrzymać ruch w dowolnym momencie należy nacisnąć przycisk “Stop” w oknie wyboru danej trajektorii ruchu
8. Po skończonym ruchu obiekt powraca na położenie inicjalizacji, a więc nie ma potrzeby ponownego uruchamiania inicjalizacji początkowej.
9. W przypadku utraty stabilności obiektu, ponownie uruchamiamy inicjalizację poprzez wcisnięcie przycisku “Initialize”. Powoduje to powrót do współrzędnych wyjściowych (80, 80, 100).
10. Lewitacja kilku obiektów na raz jest możliwa. Aby to zrobić należy wyszukać miejsca skupienia lewitacji, a następnie zawiesić kolejny obiekt nad lub pod tym miejscem w odległości ok 4.3 mm.

III. Wnioski powykonawcze

Tworząc projekt, zastosowaliśmy nowatorski pomysł transportowania obiektów w procesie produkcyjnym. W naszej pracy skupiając się na dźwięku, który jest podłożem do zjawiska akustycznej lewitacji, chcieliśmy pokazać jak mały pomysł może przerodzić się w przyszłość w nowatorską technikę sterowania obiektyami znacznie cięższymi niż te użyte przez nas. FuturFlow to urządzenie przyszłości, które może być stosowane kiedyś na szeroką skalę w bezinwazyjnym sterowaniu obiekty w 3D.

Nasze rozwiązanie to kompleksowe urządzenie, przed którym stoi jeszcze długa droga udoskonalania. Prace, które przewidujemy obejmują udoskonalenie programu poprzez dodanie znaczącej ilości funkcjonalności w tym wykonywanie bardziej dynamicznych i kompleksowych ruchów, zwiększenie mocy obliczeniowej programu, oraz dodanie zaawansowanych funkcji obliczania i programowej wizualizacji pola akustycznej lewitacji.

W ramach struktury projektu pragniemy zwiększyć ilość docelowych głośników, oraz dodać więcej układów z głośnikami aby zwiększyć moc lewitacji. Zależy nam, aby przetestować również inne opcje układów z głośnikami, szczególnie ułożenie heksagonalne. Dalszym krokiem byłoby również stworzenie połączonych urządzeń lewitacyjnych, zwiększając tym samym całkowitą długość (w porównaniu z szerokością), gdyż to właśnie ten parametr byłby najbardziej znaczący w implementacji do już określonego procesu produkcyjnego.

Odnośnie materiałów służących do lewitacji, duży potencjał widzimy w lewitacji plynów oraz przedmiotów o odmiennym materiale budowy.

Potencjalnie przygotowany przez nas levitator mógłby stanowić moduł, który w większej ilości można łączyć ze sobą dowolnie, tworząc ścieżkę transportową dostosowaną do określonych potrzeb.

IV. Kosztorys

1. FPGA altera cyclone II EP2C5144C8N 2 szt po 155,42 zł /310,84 zł
2. Sterownik MOSFET TC4427ACOA SOP-8 120 szt po 0,97 zł /116,4 zł
3. Płytnica PCB drukowana uniwersalna 1 szt /7,90 zł
4. Gwinty M8 z odzysku /0 zł
5. Nakrętki M8 z odzysku /0 zł
6. Kondensatory 0805 0,1 µF, 50 V 100 szt, 0,12 zł /12 zł
7. Przetwornik ultradźwiękowy TCT40-16T 40 kHz, 16 mm, 300 szt po 1 zł /300 zł
8. Przewody połączeniowe Męsko-żeńskie 40 szt /8,94 zł
9. Kalafonia /0 zł
10. Spoiwo lutownicze /0 zł
11. Filament PLA 2 kg /200 zł
12. Analizator stanów logicznych 1 szt /30 zł
13. Programator USB-Blaster (ALTERA CPLD/FPGA) 1 szt /19,50 zł
14. Zwierciadło sferyczne, ogniskowa 750 mm, 1 szt /200 zł
15. Jednostronny Laminat PCB 2 szt /10 zł
16. Przetwornice Step-Down Mini 360 12V, 2 szt po 3,85 zł /7,7 zł
18. Przetwornica 230 V/12V AC/DC 2A, 1 szt /12 zł
19. Konwerter poziomów napięć MSX 4-kanałowy, 2 szt 5,6 zł 11,2 zł
20. Rejestr przesuwny 74HC595 100 szt, 0,86 zł /86 zł
21. Lakier do paznokci 2 szt, 10 zł /20 zł
22. Diody RGB 4 szt, 0,7 zł /2,8 zł
23. Przełącznik kołyskowy 16 A/250 VAC czerwony 1 szt /9,15 zł
24. Gniazdo zasilające IEC C14 10A, 250V AC, 1 szt /5,2 zł
25. Adapter gniazdo-wtyk USB A , micro USB 2 szt po 5,85 zł /11,7 zł
26. Zasilacz 12V /0 zł
27. Zasilacz 5V /0 zł

Całkowita kwota: **1 381,33 zł.**

V. Bibliografia

Dźwięk i jego percepja: aspekty fizyczne i psychoakustyczne, wyd. PWN, Edward Ozimek, (2023)

Podstawy Mechatroniki, wyd. WSiP, Mariusz Olszewski, (2018)

Podstawy Elektroniki Cyfrowej, Wydawnictwa Komunikacji i Łączności WKŁ, Józef Kalisz, (2002)

Master Handbook of Acoustics, Seventh Edition, F.Alton, Ken Pohlmann, (2021)

Marzo, A. , & Drinkwater, B. W. (2019). Holographic acoustic tweezers. Proceedings of the National Academy of Sciences of the United States of America, 116(1), 84–89. <https://doi.org/10.1073/pnas.1813047115>

Foresti, D., Nabavi, M., Klingauf, M., Ferrari, A. & Poulikakos, D. Acoustophoretic contactless transport and handling of matter in air. Proc. Natl Acad. Sci. USA 110, 12549–12554 (2013).

Courtney, C. R. et al. Dexterous manipulation of microparticles using Bessel-function acoustic pressure fields. Appl. Phys. Lett. 102, 123508 (2013).

Wei, B. Acoustic levitation with self-adaptive flexible reflectors. Rev. Sci. Instrum. Hong, Z. Y., Xie, W. J. & 82, 074904 (2011).

Ding, X. et al. On-chip manipulation of single microparticles, cells, and organisms using surface acoustic waves. Proc. Natl Acad. Sci. USA 109, 11105–11109 (2012).

Ochiai, Y., Hoshi, T. & Rekimoto, J. Pixie dust: graphics generated by levitated and animated objects in computational acoustic-potential field. ACM Trans. Graph. 33, 85 (2014).

Lee, J. et al. Single beam acoustic trapping. Appl. Phys. Lett. 95, 073701 (2009).

Lam, K. H. et al. Ultrahigh frequency lensless ultrasonic transducers for acoustic tweezers application. *Biotechnol. Bioeng.* 110, 881–886 (2013).

Zhang, P. et al. Generation of acoustic self-bending and bottle beams by phase engineering. *Nat. Commun.* 5, 4316 (2014).

Baresch, D., Thomas, J. L. & Marchiano, R. Observation of a single-beam gradient force acoustical trap for elastic particles: acoustical tweezers. (2014).

Silva, G. T. & Baggio, A. L. Designing single-beam multitrapping acoustical tweezers. *Ultrasonics* 56, 449–455 (2015).

Andrade, M. A., Buiochi, F., and Adamowski, J. C., “Finite element analysis and optimization of a single-axis acoustic levitator,” *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* 57(2), 469–479 (2010).

Benmore, C. J. and Weber, J. K. R., “Amorphization of molecular liquids of pharmaceutical drugs by acoustic levitation,” *Phys. Rev. X* 1(1), 011004 (2011).

Brandt, E. H., “Acoustic physics: Suspended by sound,” *Nature* 413(6855), 474–475 (2001).

Bruus, H., “Acoustofluidics 7: The acoustic radiation force on small particles,” *Lab Chip* 12(6), 1014–1021 (2012).