VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

THEREMIN: BEZKONTAKTNÍ HUDEBNÍ NÁSTROJ

THEREMIN: TOUCHLESS MUSICAL INSTRUMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ BAŠTÝŘ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2022

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Baštýř Lukáš**
Program: Informační technologie
Název: **Theremin: bezkontaktní hudební nástroj**
Theremin: Touchless Musical Instrument
Kategorie: Vestavěné systémy

Zadání:

1. Vytvořte přehled, zdokumentujte a diskutujte vlastnosti prostředků (např. ultrazvukových či jiných čidel) použitelných pro bezkontaktní detekci polohy a rychlosti pohybu částí lidského těla; použitelnost vybraných čidel pro tento účel experimentálně ověřte a vyhodnoťte.
2. Navrhněte princip činnosti a ovládání hudebního nástroje ("Theremin") ovládaného bezkontaktně předem danými částmi lidského těla pomocí vhodně zvolené kombinace prostředků z bodu 1; zvolte platformu pro vytvoření prototypu nástroje, vytvořte blokové schéma systému založeného na této platformě a navrhněte mechanismus jeho řízení.
3. Prototyp nástroje z bodu 2 realizujte; principy a problémy související s konečnou realizací podrobně zdokumentujte.
4. Funkčnost vytvořeného prototypu vhodně demonstrujte, ověřte a shrňte.
5. Diskutujte vlastnosti zvolené realizace, její využitelnost a navrhněte její možné návaznosti či rozšíření.

Literatura:

- P. Nikitin, "Leon Theremin (Lev Termen)," in *IEEE Antennas and Propagation Magazine*, vol. 54, no. 5, pp. 252-257, Oct. 2012, doi: 10.1109/MAP.2012.6348173.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 29. října 2021

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a implementací digitálního bezkontaktního hudebního nástroje theremin. V první části je nejprve zdokumentován theremin, jeho vlastnosti a dosavadní řešení. Druhá část je zaměřena na dokumentaci a testování softwaru a hardwaru, který je možné použít k řešení této práce. Předposlední část je zaměřena na samotnou implementaci nástroje a vytvoření funkčního prototypu. Finální kapitola testuje funkcionalitu finálního prototypu nástroje.

KLÍČOVÁ SLOVA

Theremin, hudební nástroj, bezkontaktní ovládání, mikrokontrolér, senzory, ToF, ultrazvuk

ABSTRACT

This bachelor thesis deals with the design and implementation of a digital touchless musical instrument theremin. In the first part, the theremin, its features and existing solutions are first documented. The second part focuses on the documentation and testing of the software and hardware that can be used to solve this thesis. The penultimate part focuses on the actual implementation of the instrument and the creation of a working prototype. The final section tests functionality of the final prototype instrument.

KEYWORDS

Theremin, musical instrument, touchless control, microcontroller, sensors, ToF, ultrasound

BAŠTÝŘ, Lukáš. *Theremin: bezkontaktní hudební nástroj*. Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačových systém, 2022, 50 s. Bakalářská práce. Vedoucí práce: Ing. Josef Strnadel, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Lukáš Baštýř
VUT ID autora:	221825
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Theremin: bezkontaktní hudební nástroj

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Josefovi Strnadlovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Bakalářská práce se věnuje implementaci bezdotykového hudebního nástroje, přesněji nástroje theremin. Vychází ze semestrální práce, jejímž tématem bylo zdokumentování možných senzorů pro detekci pohybů částí lidského těla a ověření jejich použití v této práci. Druhým tématem semestrální práce bylo navrhnout princip ovládání výsledného zařízení a zvolení, jaká část lidského těla bude použita pro ovládání zařízení (v tomto případě byla zvolena dlaň ruky) a výběr vhodné platformy pro implementaci.

Theremin je hudební nástroj, který v roce 1919 vynalezl tehdy sovětský fyzik Lev Sergejevič Těrmen. Jedná se o velice specifický hudební nástroj, který stále i v dnešním světe nemá konkurenci, co se týče způsobu hraní na tento nástroj. Oproti takřka každému jinému hudebnímu nástroji tehdejší, ale i dnešní doby je theremin ovládán bez dotyků samotného nástroje. Stačí pouze pohybovat rukou od a ku anténě tohoto nástroje, čímž se změní frekvence nebo hlasitost tónu, který nástroj vydává. Fakt, že je přistroj ovládán bezdotykově, přináší při digitální implementaci řadu výzev, které je nutné překonat. Asi nejpodstatnější z těchto výzev je samotná detekce pohybu dlaně a následná interpretace dat o tomto pohybu pro generování výstupního tónu.

První kapitola práce se zabývá úvodem do problematiky, vysvětlením základního principu, na kterém theremin funguje, a prozkoumáním dosavadních řešení.

Druhá kapitola práce se zabývá realizačními možnostmi. Zde jsou jednotlivé senzory, mikrokontrolery a programovací jazyky zdokumentovány a otestovány. Nakonec se vybere nejlepší kombinace těchto hardwarových a softwarových možností pro práci.

Třetí kapitola práce je poté zaměřena na implementaci nástroje. Nejprve je zde navržení blokového schématu nástroje a jeho sestavení. Následně je zde popsána softwarová implementace získávání dat ze senzorů, generování tónu a ovládání konfiguračního menu.

Poslední kratší kapitola se zabývá otestováním finálního sestaveného nástroje.

1 Rešerše k tématu bakalářské práce

1.1 Úvod do problematiky bezdotykových hudebních nástrojů

V dnešní době existuje pět základních skupin hudebních nástrojů [1]:

- chordofony – strunné nástroje (např. housle, klavír)
- aerofony – vzduchové nástroje (např saxofon, varhany, atd.)
- membranofony – blanozvučné nástroje (např bubny a podobné)
- idiofony – samozvučné nástroje (např. činely nebo xylofon)
- elektrofony – elektronické či eletrifikované nástroje (např. elektrická kytara nebo klávesy)

Nás budou nadále zajímat pouze **elektrofony**, čili elektronické hudební nástroje. Ty byly v roce 1940 přidány mezi *Sachsova–Hornbostelovu klasifikaci hudebních nástrojů* a vyznačují se hlavně tím, že pro tvorbu jejich výsledného zvuku se nějak využívá elektřina. Avšak výstupní zvuk těchto nástrojů nemusí být nutně generován pomoci zvukové syntézy. Tato skupina nástrojů se totiž dále dělí na 3 podskupiny podle toho, jakým způsobem je zvuk v nástroji vytvářen [2].

První z těchto skupin je tvořena nástroji, které místo mechanické akce, kterou člověk musel provádět manuálně, používají elektrickou. Sem se dají zařadit například elektrické varhany. Druhou skupinou jsou nástroje, které mají pouze elektricky zesílený výstup. Tyto nástroje tedy v základu pracují na stejném principu jako jejich ne-elektrické protějšky, ale jejich výsledný zvuk je následně zesílený a převeden na elektrický signál. Sem patří například elektrická kytara. Poslední skupinou a tou, která nás nejvíce zajímá, jsou takzvané **radioelektrické** nástroje, které generují svůj výstupní tón/signál pomocí elektronického obvodu a nějaké syntézy zvuku. Právě mezi tyto hudební nástroje se dá zařadit nástroj, kterým se zabývá tato práce a to **theremin**.

Lev Sergeyevich Termen, mimo Rusko více znám jako **Léon Theremin**, narozen roku 1896, byl tehdejší sovětský vědec, který v roce 1920 pracoval na státem sponzorovaném projektu na výzkum senzorů detekce blízkosti objektů. Během výzkumu zde narazil na zajímavý jev a to ten, že kapacitance lidského těla může ovlivňovat frekvenci elektronického oscilátoru. Theremin následně využil tohoto zjištění pro sestavení prvního prototypu thereminu, kterému dal tehdy název **etherphone** [3]. Toto původní zařízení mělo oproti běžně známým verzím thereminu pouze jednu anténu. Ta se používala pro změnu frekvence tónu, které nástroj vydával. Čím blíže byla ruka hráče k anténě (kondenzátor, který se skládal z antény a lidského těla, měl nejmenší kapacitanci), tím vyšší byla frekvence vydávaného tónu. Hlasitost

byla ovládána pomocí pedálu, který se nacházel na zemi, a podle sešlápnutí udával hlasitost. Ještě v tomtéž roce udělal novější verzi tohoto nástroje, kde byl pedál pro změnu amplitudy nahrazen horizontální zakulacenou anténou, kde stejně jako u vertikální antény vzdálenost ruky udávala amplitudu. V roce 1927 se vydal na turné po Evropě, kde představoval svůj nový hudební nástroj a demonstroval, jak funguje. Koncem roku 1927 se přestěhoval do Spojených států amerických, kde dále předváděl svůj nástroj a dále na něm pracoval a kde tento nástroj také dostal svoje konečné jméno, theremin.

Obr. 1.1: Léon Theremin hrající na theremin[1]

Co se týče bezkontaktních hudebních nástrojů, těch ani v dnešní době moc neexistuje, jelikož pro vytvoření nějakého tónu je u všech typů nástrojů zapotřebí nějaký vstupní podnět. A téměř ve všech případech je tento podnět nějaký fyzický dotyk části nástroje samotného. U chordofonů to je rozkmitání struny. U membranofonů zase rozkmitání membrány. Idiofony vytváří zvuk rozkmitáním vlastního těla. Za jedinou výjimku by se zde daly označit pouze některé aerofony, jako je například panova flétna, u kterých stačí, aby do nich hráč pouze foukal a není technicky potřeba se jich nijak dotýkat. Elektrofony ve velké části případů jsou také ovládány fyzickými dotyky. Například elektrické klávesy, které se dají použít na syntézu všemožných hudebních nástrojů, se stále musí mačkat pomocí prstů. Avšak elektrofony mají oproti všem ostatním typům hudebních nástrojů velikou výhodu. Jejich vstupem nemusí být nutně fyzický podnět. Nástroj může být klidně předprogramovaný, aby hrál sám nějaké tóny. A nebo, jako v případě již dříve zmíněného thereminu,

[1]Autor: Bettmann, Corbis – http://tygodnik.onet.pl/35,0,57107,sowiecki_faust,artykul.html (polish), Volné dílo, Wikimedia

je vstupem nějaký výstup elektronického obvodu, který se dá bezkontaktně ovlivňovat. V případě thereminu to je vysokofrekvenční oscilační obvod, jehož podnětem je změna kapacitance kondenzátoru, jehož negativní elektrodu tvoří samotné tělo hráče a kladnou elektrodu tvoří anténa nástroje.

1.2 Princip a ovládání thereminu

Jak již bylo řečeno výše, theremin se od téměř všech jiných hudebních nástrojů odlišuje tím, že je hrán bez přímých dotyků nástroje. Po zapnutí (nebo v dnešní době digitálních thereminů po jeho nastavení a zapnutí) stojí nástroj před hráčem, který ho bezdotykově ovládá pomocí pohybů rukou v blízkosti dvou antén nástroje. Kulatá horizontální anténa je na ovládání amplitudy (hlasitosti) tónu, vertikální rovná anténa zase na ovládání frekvence (výšky) tónu. Ve většině případů je použita pravá ruka na ovládání frekvence a levá ruka na ovládání amplitudy. Nic hráči samozřejmě nebrání nástroj otočit a prohodit tak, která ruka ovládá co. V originálním prototypu, čili etherphonu, byla tato druhá anténa pro změnu amplitudy nahrazena pedálem, který obsahoval potenciometr, a hráč ho ovládal pomocí nohy [3].

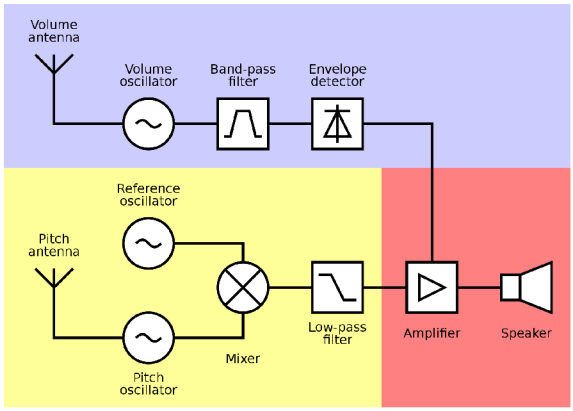
Obr. 1.2: Antény thereminu[2]

Obvod pro změnu frekvence (obr. 1.3, žlutá část) je tvořen dvěma rezonančními obvody a dolní propustí (lineární filtr). Výsledný tón vzniká na základě heterodynu, což je signál, který vznikne smíšením různých signálů. Jeden z těchto dvou rezonančních obvodů má fixní frekvenci (v hodnotách stovek kilohertz). Druhý je variabilní a má velice podobnou (může mít i stejnou) volně běžící frekvenci jako fixní. Avšak kondenzátor tohoto oscilačního obvodu má variabilní kapacitanci. Jeho pozitivní deska je tvořena anténou, jeho uzemněná deska je hráčova ruka. Tudíž změnou vzdálenosti ruky od antény se změní i kapacitance kondenzátoru, což následně vede

[2]Převzato z webu muziker.cz, následně upraveno popisem antén

ke změně frekvence variabilního oscilačního obvodu. Čím je ruka dále od antény, tím se frekvence obvodu více přibližuje frekvenci volně běžícího obvodu a změna kapacitance tedy má největší vliv na celkovou frekvenci obvodu. Toto vede k faktu, že změna frekvence není lineární a nástroj je tedy citlivější, čím dále je hráčova ruka od antény. U původní verze thereminu byla celková změna frekvence vcelku malá, mezi 65Hz až 3kHz [3], což jsou zhruba 3 oktávy. Dnešní verze již mají větší rozsah, a to kolem 5ti oktáv. Většinou je ale tento rozsah také nastavitelný. Tento frekvenční rozsah variabilního oscilátoru tedy určuje výsledný frekvenční rozsah, kterého může nástroj dosáhnout. To ale není výstupní frekvence nástroje. Výstup tohoto variabilního oscilátoru se poté odečte od frekvence fixního oscilátoru a tím se dostane výsledná frekvence (heterodyn). Tento výsledný signál po menších úpravách poté projde přes dolní propust a pokračuje dál do zesilovače a následně do reproduktoru (obr. 1.3, červená část). [4]

Obvod pro změnu amplitudy (obr. 1.3, modrá část) se skládá z jednoho variabilního oscilačního obvodu, pásmové propusti a detektoru obálky. Oscilační obvod zde funguje stejně jako variabilní obvod pro změnu frekvence. Má variabilní kondenzátor, který je tvořen anténou a hráčovou rukou. Vzdálenost ruky od antény mění kapacitanci kondenzátoru, což zase mění celkovou frekvenci oscilátoru. Výstup tohoto oscilačního obvodu následně projde pásmovou propustí a detektorem obálky, které z této výstupní frekvence udělají analogový výstup, kterým se dá ovládat již dříve zmíněný zesilovač. Levnější verze thereminu dnes používají místo antény jednoduchý potenciometr na zařízení samotném, který je ovládán jednou rukou. [4]



Obr. 1.3: Blokové schéma thereminu[3]

[3]Autor: Dlrohrer2003 - Vlastní práce, CC BY-SA 3.0, Wikimedia

1.3 Existující digitální řešení

Theremin se bohužel nikdy nestal moc populárním nástrojem. Jeho hlavní využití bylo v hororových či futuristických filmech. Ani v dnešní době se jeho popularita příliš nezměnila. Lidé ho sice znají, ale pravděpodobně ho v realitě nikdy neslyšeli hrát. Kvůli této nízké popularitě se nikdy nevyráběl ve velké míře a tak tomu je i nyní. Jedná se o specializovaný produkt pro poměrně malé množství lidí a v dnešní digitální době pro ještě menší využití. To ale neznamená, že neexistují moderní řešení digitálního thereminu, právě naopak. Povedlo se mi najít mnoho menších (a jednodušších) projektů, ale i pár větších a komerčních řešení.

Mezi tyto menší projekty bych zařadil ty, které se dají jednoduše dohledat na internetu. Jedná se většinou o nadšence, kteří si pomocí vývojové desky s mikrokontrolérem (například Arduino Uno) a různých senzorů (například ultrazvukových senzorů[4], nebo jenom pomocí posuvných rezistorů) sestaví a naprogramují digitální „theremin“. Tyto projekty se do jisté míry podobají zaměření praktické části této práce. Avšak snahou této práce je udělat celistvé finální a komplexnější řešení, které oproti téměř všem řešením (ne jenom těmto projektům) nepoužívá vyhledávací tabulku na generování tónu, ale generuje tón za běhu.

Jediný větší projekt, který se mi povedlo najít je **OpenTheremin**[5]. Jedná se o *open source* projekt[6], který je podobně jako většina dříve zmíněných menších projektů založený na platformě Arduino Uno. Oproti nim se ale jedná o značně komplexnější projekt, který až do verze 4 vyžadoval rozšiřující desku (verze 4 je již samostatný nástroj). Podobně jako běžný theremin obsahuje analogovou část pro snímání pozice hráčovy ruky. Kvůli této analogové části bych ho ale nenazval čistě digitálním, ale analogovo-digitální. Oproti běžnému thereminu má dále ještě digitální část pro generování (syntetizování) výsledného tónu. Díky této digitální části je možné například změnit vlnu generovanou tónu nástroje. Tyto vlny jsou uloženy ve vyhledávacích tabulkách, kde se podle dané pozice v čase vyhledá potřebná hodnota generované vlny.

[4] arduino.cc – Theremino a theremin made with Arduino

[5] gaudi.ch – OpenTheremin

[6] Open source – otevřený projekt, který člověk může volně upravovat a podle licence i dále tyto úpravy šířit

Obr. 1.4: OpenTheremin V4[7]

Jakožto komerční řešení se mi podařilo najít **Moog Theremini**[8]. Jedná se také o digitální theremin od společnosti Moog Music.inc. Tato firma, založena roku 1953, se zabývá syntetizéry a jeden z jejích moderních produktů je právě tento digitální theremin. Obdobně jako OpenTheremin ale obsahuje analogovou část pro snímání pozice ruky hráče a tudíž bych ho také nazval jako analogovo-digitálním. V základě jsou si OpenTheremin a Moog Theremini velice podobné. Jelikož je Moog Theremini komerční produkt, má mnohem více funkcí a jedná se o kompletní zařízení, které obsahuje všechny potřebné součásti pro okamžité zprovoznění.

Obr. 1.5: Moog Theremini[9]

[7]Převzato z webu guaidshop.ch
[8]MoogMusic.com – Theremini
[9]Převzato z webu thomann.de

1.4 Porovnání analogové a digitální verze thereminu

Původní theremin bylo plně analogové zařízení. Dnešní levnější verze jsou také často plně analogové, ale už se objevují i digitální (analogovo-digitální) verze, jako již dříve zmíněný Moog Theremini a OpenTheremin. Když porovnáme analogový theremin s těmito digitálními/analogovo-digitálními verzemi, je největší rozdíl právě v jejich digitální části. Díky ní je možné prakticky jakkoliv upravit výstup nástroje. Mezi nejčastější modifikace, které jsou tímto umožněny, patří změna tvaru výstupní vlny. Vlna generovaného tónu analogového thereminu je do jisté míry podobná skloněné sinusovce, ale téměř nikdy to není čistá sinusovka. Jelikož digitální verze používají pro generování tónů výhradně vyhledávací tabulky, ve kterých jsou jednotlivé vlny již před–vygenerované, je možné generovat prakticky jakoukoliv vlnu (dokud je uložena v paměti). Moog Theremini také například hráči umožňuje nastavit délku prodlevy mezi pohybem ruky a změnou výstupního tónu, nebo například přidání dozvuku a dalších efektů [5]. Tabulka 1.1 níže popisuje nejpodstatnější rozdíly.

Název:	Theremin	Moog Theremini	OpenTheremin
Typ nástroje:	Analogový	Digitální	Digitální
Frekvenční rozsah:	Fixní	Variabilní	Fixní
Typ vlny:	Skloněná sinusovka	Různé	Různé
Způsob generace:	Modulace[10]	Syntéza	Syntéza

Tab. 1.1: Základní rozdíly

1.4.1 Technické parametry a požadované vlastnosti

Jelikož byl původní theremin analogové zařízení, je potřeba si při jeho převodu na digitální verzi definovat požadované vlastnosti pro vstupy a výstupy zařízení. Analogová zařízení mají kontinuální spojitou změnu, zatímco digitální zařízení potřebují vzorkování o nějaké frekvenci. Čím větší je daná frekvence, tím více se výsledek podobá realitě. S rychlejší frekvencí vzorkování ale také přichází větší nároky na hardware. Je tedy také potřeba si vyjasnit, kdy je vhodné nahradit přesnost za rychlost a naopak.

Co se týče vstupu (snímání pozice hráčovy ruky), tak citlivost i rychlost snímání jsou u analogové verze teoreticky neomezené. Digitální implementace se s tímto musí nějak vyrovnat. Rychlost snímání zde bude nahrazena za vzorkovací frekvenci. Čím vyšší je tato frekvence, tím plynulejší je přechod mezi výstupními frekvencemi a výsledný tón je tak značně příjemnější na poslech. Zde by se tedy hodila co nejvyšší

[10] Wikipedia – modulace

vzorkovací frekvence. Je zde ale také možnost použít *interpolaci*[11] pro generování hodnot mezi dvěma vzorky. Tím se dá docílit relativně plynulejšího přechodu mezi výstupními frekvencemi, přičemž vstupní vzorkovací frekvence může zůstat podstatně nižší. Citlivost thereminu byla teoreticky také neomezena. U digitální implementace je toto omezeno. Při použití analogových čidel je citlivost omezena rozlišením AD (analogově digitálního) převodníku. Při použití digitálních čidel (čidla komunikující přes nějaký protokol), je citlivost omezena citlivostí použitých senzorů (čili je omezena převodníkem na daném čidlu). Oproti vzorkovací frekvenci se s tímto nedá moc dělat a je potřeba vybrat co nejlepší hardware.

Výstupní frekvence analogového thereminu je závislá na frekvenčním rozmezí variabilního oscilačního obvodu. Maximální frekvence byla velice blízká (někdy i shodná) s frekvencí fixního oscilačního obvodu a v průměru byla zhruba 65Hz pod frekvencí fixního oscilátoru [3]. Minimální frekvence poté byla zhruba o 3kHz nižší než frekvence fixního oscilátoru. Rozmezí výstupní frekvence tedy bylo zhruba 65 – 3000Hz, přibližně 3 oktávy. Moderní verze thereminu často mají větší rozsah kolem asi 5ti oktáv. V digitální verzi se toto rozmezí dá teoreticky libovolně nastavovat a je i možné měnit jeho šířku (na úkor možné přesnosti).

Kvůli tomu, jak kapacitance kondenzátoru ovlivňuje frekvenci oscilátoru, byla změna frekvence tónu u původního thereminu nelineární. Důvodem pro to byl fakt, že když je hráčova ruka nejdále od antény a frekvence oscilátoru byla nejblíže volně běžící frekvenci, malé změny v kapacitanci antény měly veliký dopad na změnu frekvence oscilátoru. Naopak čím blíže byla hráčova ruka od antény, malé změny kapacitanci vůči celkové velké kapacitanci se nedokázaly tolik projevit. To ale nebylo žádoucí a později se pomocí ladění dala výstupní frekvence linearizovat. V práci tedy bude snaha o implementaci lineární změny frekvence.

Typ řešení:	Analogové	Digitální
Rozlišení snímání polohy:	Teoreticky neomezené	Limitováno rozlišením AD převodníku
Rychlost snímání polohy:	Teoreticky neomezené	Limitováno vzorkovací frekvencí
Frekvenční rozsah:	Limitován (3 – 5 oktáv)	Variabilní
Odezva:	Okamžitá	Limitovaná rychlostí zpracování

Tab. 1.2: Rozdíly mezi implementacemi

[11] Wikipedia – interpolace

1.5 Problematika měření spojená s digitalizací

Podstatnou součástí této práce, jakožto bezkontaktního hudebního nástroje, je opakované měření pro snímání vzdálenosti hráčovy ruky od nástroje. Žádné měření, měřicí přístroj nebo měřicí postup ale nejsou dokonale přesné a do výsledku se nám vždy promítne nějaká chyba. Snahou je tedy odstranit co nejvíce možných chyb, aby výsledné měření co nejpřesněji odpovídalo reálným hodnotám.

Rozdíl mezi skutečnou měřenou hodnotou a výslednou hodnotou měření se nazývá *chyba měření*. Je veliké množství zdrojů těchto chyb a jejich základní dělení je buďto podle jejich působení na měření (náhodné, systematické a hrubé), nebo podle jejich zdroje (přístroje, metody, pozorování a vyhodnocení) [6].

Náhodné chyby ovlivňují výsledky měření zcela nahodile a je téměř nemožné je přesně předpovídat. Dají se poznat během opakovaného měření, kdy naměřená výsledná hodnota bude pokaždé odlišná, ale bude se pohybovat okolo reálné hodnoty. Pro určení jejich hodnoty se používají statistické modely pro pravděpodobnostní rozdělení s požadovanou náhodnou chybou. V praxi se poté nejčastěji používá normální-Gaussovo rozdělení [6].

Hrubé chyby jsou zcela nepředvídatelné chyby, které znehodnotí dané měření. Jedná se o naměřené hodnoty, které se značně odchylují od reálných hodnot. Tyto chyby mohou být částečně (někdy i zcela) eliminovány dodržováním příslušných postupů, podmínek měření a školením personálu. Jedním z těchto postupů je například filtrování hodnot, které by reálně nebylo možné v daném okamžiku získat.

Systematické chyby jsou chyby, které mají při stálých podmínkách stálou hodnotu. Dopad těchto chyb se dá často z velké části odstranit pomocí korekcí a kompenzací. Zbylé části těchto chyb, které se nedají odstranit, se nazývají nevylučitelné systematické chyby.

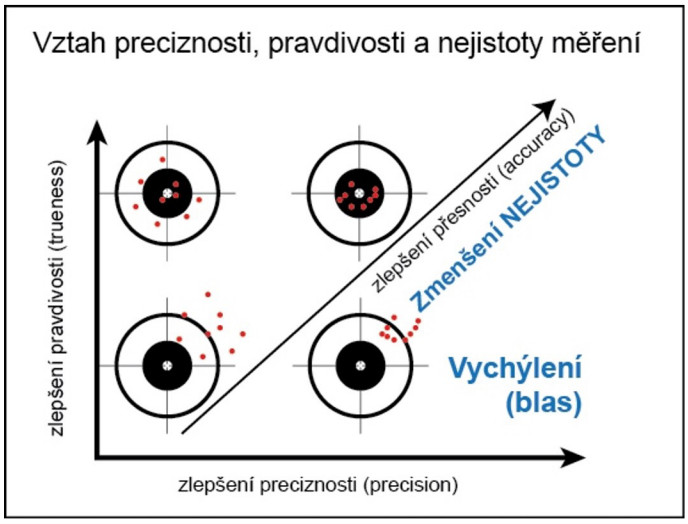
Chyby přístroje jsou způsobeny hlavně nedokonalostí při výrobě, opotřebením a stářím přístroje. Také sem patří například použití přístroje ve špatných podmínkách nebo ve špatné orientaci/nesprávném uchycení. Chyby metody vznikají při zjednodušení měřicí metody. Chyby pozorování a vyhodnocení jsou způsobeny pozorovatelem. U pozorování se jedná hlavně o chyby vzniklé kvůli nedokonalosti smyslů. U vyhodnocení se může jednat třeba o zjednodušení vztahů nebo linearizaci a zaokrouhlení výsledků.

Je tedy zřejmé, že z výsledné hodnoty měření není možné přesně určit reálnou hodnotu a samotnou chybu. Je ale možné určit rozsah hodnot, kterých může výsledná hodnota nabýt. K tomu slouží *nejistota měření*. Tento nový koncept nahradil koncem dvacátého století již dříve zmiňovaný koncept chyb měření. Zdroje nejistot (značí se jako u) jsou stejné jako zdroje chyb a jedná se tedy o všechny možné proměnné během měření (zaokrouhlování, nepřesnost měřicích přístrojů atd.). Výsledná

kombinovaná nejistota se skládá ze dvou základních typů nejistot - **nejistoty typu A** a z **nejistoty typu B** [6]. Nejistota typu A (značena u_A) vychází ze statistické analýzy výsledků opakovaného měření. Nejistota typu B (značena u_B) vychází z dílčích nejistot všech zdrojů nejistot (informace od výrobce, poznatky z dřívějších měření, informace z kalibrace atd.) a musí se jednat o jinou metodu než je statistická analýza. Vzorec pro výpočet kombinované nejistoty je poté:

$$u_C = \sqrt{{u_A}^2 + {u_B}^2} \tag{1.1}$$

Souhrn všech možných chyb/nejistot měření nám udává výslednou **přesnost měření**, která se skládá ze dvou složek. Jednou z nich je **pravdivost**, která určuje vzdálenost měření od reálné hodnoty. Druhou je **preciznost**. Ta udává vzdálenost mezi jednotlivými měřeními hromadného měření. Společně tyto dvě složky tedy vytváří výslednou přesnost měření. Obrázek 1.6 popisuje vztah mezi oběma složkami vzhledem k přesnosti.



Obr. 1.6: Přesnost - pravdivost a preciznost[12]

V práci je tedy snaha eliminovat co nejvíce možných chyb během měření a to pomocí testování jednotlivých komponent a zjištění jejich vlastností (viz podsekce testování senzorů 2.3.3). Testování samotné se snaží zjistit přesnost a preciznost jednotlivých senzorů a jejich limitující faktory. V implementaci je následně snaha aplikovat získané znalosti z této kapitoly a testování pro získaní co nejlepších možných výsledků (pokud je to potřeba).

[12] Autor: Václav Senft, převzato z webu slideplayer.cz

2 Rozbor realizačních možností

Tato kapitola se zabývá dokumentací, porovnáním a testováním možných hardwarových a softwarových prostředků (mikrokontrolérů, senzorů a programovacích jazyků).

2.1 Mikrokontroléry

Vzhledem k zaměření této práce je potřeba vybrat adekvátní mikrokontrolér, který bude zpracovávat všechny potřebné vstupy a generovat potřebné výstupy v reálném čase. Níže jsou stručně popsány jednotlivé vývojové desky s příslušnými mikrokontroléry, ke kterým jsem měl přístup a mezi kterými jsem se rozhodoval. Největší rozdíl mezi jednotlivými vybranými mikrokontroléry je architektura. Jeden je **8bitový**, zbylé dva jsou **32bitové**. Hlavní výhodou 8bitových mikrokontrolérů byla donedávna cena a spotřeba. To už ale v dnešní době moc neplatí. V porovnání s 32bitovými mikrokontroléry jsou již mnohdy stejně drahé, nebo i dražší za jednotku, a spotřebou je moderní 32bitové mikrokontroléry začínají dohánět. Velikou nevýhodou je pak tedy samotný výkon. 32bitové mikrokontroléry jsou podstatně výkonnější, obzvláště při práci s velkými a desetinnými čísly.

2.1.1 Arduino UNO (ATmega328)

Arduino Uno je asi jedna z nejznámější vývojových desek na světe. Deska má velikost zhruba jako kreditní karta a je osazena mikrokontrolérem **ATmega328**[1]. Mikrokontrolér je z rodiny AVR od společnosti Microchip (dříve Atmel). Jedná se o 8bitový mikrokontrolér o maximální rychlosti 16MHz. V porovnání s ostatními mikrokontroléry má poměrně málo paměti, ale pro tuto práci by to nemusel být problém. Pro potřeby této práce by mikrokontrolér měl podporovat všechny možné komunikační protokoly (I2C, SPI, UART) a typy vstupů/výstupů (analogové a digitální).

[1] Microchip – ATmega328

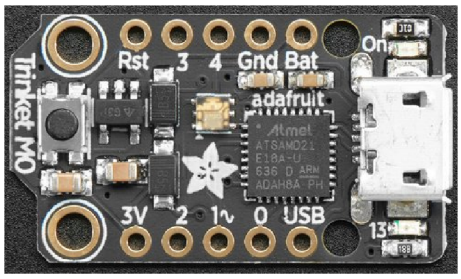
Tab. 2.1: Arduino Uno[2]

ATmega328	
Architektura	8bit AVR
Rychlost	16MHz
Paměť	32KB ROM, 2KB RAM, 1KB EEPROM
Vstupy	23 GPIO
Komunikační protokoly	USART, I2C, SPI
AD převodník	10bit

Tab. 2.2: Základní vlastnosti [7]

2.1.2 Trinket M0 (ATSAMD21E18)

Další vývojová deska, ke které jsem měl přístup, byla Trinket M0 od společnosti Adafruit. Ta je osazena mikrokontrolérem **ATSAMD21E18**[3], který je také od společnosti Microchip. Je postaven na 32bitovém jádru ARM Cortex M0+[4] a je taktován na 48MHz. V porovnání s předchozí deskou je zde výhoda 32 bitové platformy. Nevýhodou je zase rozhraní. Mikrokontrolér má sice dostatek vstupů/výstupů, většina z nich ale nemá na desce vývody. Pro tuto práci je ale vstupů/výstupů dostatek.



Tab. 2.3: Trinket M0[5]

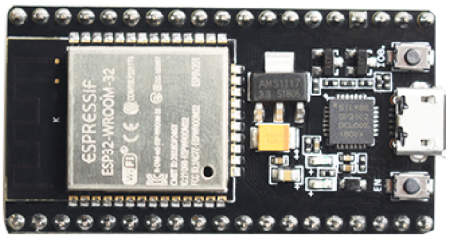
Trinket M0	
Architektura	32bit Arm Cortex-M0+
Rychlost	48MHz
Paměť	256KB ROM, 32KB RAM
Vstupy	26 GPIO pinů
Komunikační protokoly	USB, USART, I2C, SPI, I2S
AD převodník	12bit
DA převodník	10bit

Tab. 2.4: Zakladní vlastnosti [8]

[2] Převzato z webu Arduino.cc
[3] Microchip.com – ATSAMD21E18
[4] Arm.com – Cortex-M0+
[5] Převzato z webu Adafruit.com

2.1.3 NodeMCU-32S (ESP32-S3)

NodeMCU-32S byla poslední deska, kterou jsem měl k dispozici. Je od společnosti Ai-Thinker[6] a stejně jako předchozí Trinket M0 obsahuje 32bitový mikrokontrolér. Oproti Trinket M0 se ale nejedná o jádro ARM. Mikrokonroler **ESP32-S3**[7] totiž obsahuje dvě jádra Xtensa LX7[8], která jsou taktována na maximální rychlost 240MHz. Největším rozdílem mezi předešlými miktrokontrolery je ten, že tento obsahuje dvě jádra. Je tedy možné pracovat na dvou úlohách současně, což bude pro tuto práci velice užitečné. Samotná deska má také veliké množství vstupů/výstupů a mikrokontrolér podporuje všechny možné komunikační protokoly. Mikrokontroléry ESP32 jsou hlavně známé tím, že obsahují vestavěný WiFi modul. Tento specifický mikrokontrolér obsahuje i Bluetooth modul. Ani jeden z těchto modulů ale není pro práci podstatný.



Tab. 2.5: NodeMCU-32S[9]

ESP32-S3	
Architektura	32bit Xtensa LX7
Rychlost	240MHz(x2)
Paměť	384 KB ROM, 512 KB SRAM
Vstupy	45 GPIO pinů
Komunikační protokoly	UART(x3), I2C(x2),SPI(x4), I2S(x2)
AD převodník	12bit(x2)

Tab. 2.6: Základní vlastnosti [9]

2.1.4 Testování mikrokontrolérů

Všechny výše popsané mikrokontroléry obsahují všechny potřebné zdroje, které by mohly být potřebné pro tuto práci. Výsledné testování bylo tedy zaměřeno hlavně na celkovou rychlost mikrokontrolérů. Ta byla otestována pomocí několika jednoduchých funkcí. Čtyři na sčítání celých čísel různě velkých typů, jedna na sčítání necelých čísel a jedna na dělení necelých čísel. Tímto se dá jednotlivě otestovat jak rychlost *ALU* (aritmeticko-logická jednotka)[10], tak rychlost *FPU* (matematický koprocesor)[11]. Každá funkce měla sto tisíc iterací, ve kterých se prováděla daná akce

[6] Ai-Thinker.com
[7] espressif.com – ESP32-S3
[8] Popis Xtensa LX7 jader
[9] Převzato z webu Ai-Thinker.com
[10] Wikipedia – Aritmeticko-logická jednotka
[11] Wikipedia – Matematický koprocesor

na proměnné daného typu. Při sčítání celých čísel se vždy přičítala pouze 1. Při sčítání necelých čísel se přičítala hodnota 0.001 do proměnné `double` (velikost tohoto typu záleží na platformě). Při dělení necelých čísel se dělila maximální hodnota typu `double` pro danou platformu hodnotou 1.00001. Velikost jednotlivých typů pro jednotlivé platformy popisuje tabulka 2.7.

MCU	uint8_t	uint32_t	uint64_t	int	double
Atmega	8bit	32bit	64bit	16bit	32bit
ATSAMD	8bit	32bit	64bit	32bit	64bit
ESP	8bit	32bit	64bit	32bit	64bit

Tab. 2.7: Velikost typů

Tabulka 2.8 níže zobrazuje čas v příslušných jednotkách, který byl potřeba pro výpočet jednotlivých funkcí. Výsledné časy byly získány ze sta běhů, které byly následně zprůměrovány.

Znaménka v hlavičce tabulky značí akci, která se prováděla pro daný typ funkce. Mikrokontrolér ESP32-S3 byl jako jediný otestován vícekrát vždy s jiným nastavením (závorky za označením mikrokontroléru). Jelikož tento mikrokontrolér jako jediný dovoluje vcelku jednoduše měnit rychlost jádra, byly testy provedeny i na různých rychlostech pro přesnější porovnání se zbylými mikrokontroléry. První značka tak označuje rychlost mikrokontroléru v MHz. Druhá značka označuje počet použitých jader. **D** značí použití obou jader, **S** naopak značí použití pouze jednoho jádra.

MCU	uint8(+)	uint32(+)	uint64(+)	int(+)	double(+)	double(/)
Atmega	3.24μs	3.32μs	2.76μs	3.68μs	918.585ms	3.087s
ATSAMD	1.86μs	1.79μs	1.79μs	1.79μs	427.352ms	2.045s
ESP (240, S)	0.71μs	0.75μs	0.78μs	0.74μs	25.013ms	254.758ms
ESP (240, D)	0.92μs	0.88μs	0.25μs	0.99μs	24.940ms	254.750ms
ESP (10, S)	19.82μs	18.81μs	19.03μs	20.83μs	731.423ms	7.462s
ESP (20, S)	8.84μs	10.06μs	10.05μs	8.98μs	327.365ms	3.337s
ESP (40, S)	3.95μs	4.52μs	3.97μs	4.47μs	155.923ms	1.589s

Tab. 2.8: Doba výpočtů

Z tabulky je patrné, že při výchozích maximálních rychlostech mikrokontrolér ATmega328 je nejpomalejší ze všech dostupných mikrokontrolérů, ať se jedná o celá nebo necelá čísla. Jako druhý se umístil mikrokontrolér ATSAMD21E18 a jako

značně nejrychlejší je ESP32-S3. Tento výsledek se dal předpokládat, jelikož ATmega328 je jako jediný 8bitový mikrokontrolér a je taktován na nejnižší rychlost. ESP32-S3 je zase naopak taktován nejvýše.

Ve výsledcích je také dobře vidět, že mít dvě jádra ještě neznamená dvakrát rychlejší výpočet. Z jednotlivých funkcí se zde vytvoří úlohy. Tyto úlohy se poté dají paralelizovat, takže jedno jádro vždy pracuje pouze na jedné úloze. Proto jednotlivé časy úloh jsou mezi jedno-jádrovým a dvou-jádrovým režimem velice podobné (ale nejsou stejné). U počítání s celými čísly je vidět, že téměř všechny funkce trvaly celkově delší dobu. Můj odhad je, že je toto způsobeno přepínáním mezi jednotlivými úlohami, což přidává časovou náročnost. Jedinou výjimkou zde je počítání se 64bitovým integerem bez znaménka. Kvůli této anomálii jsem testy spouštěl několikrát a pokaždé to dopadlo velice podobně. Z nějakého mně neznámého důvodu bylo počítání s 64bitovým integerem bez znaménka při použití obou jader značně rychlejší, než jakýkoliv jiný výpočet. Z výsledků je také možné vydedukovat, že pokud člověk vytvoří jako první úlohu nejdéle trvající funkci (dělení typu `double`), na zbylém jádru se během výpočtu této úlohy stihnou prostřídat všechny ostatní vytvořené úlohy a celková doba výpočtu ve dvou-jádrovém režimu tak trvá stejně dlouho jako dělení typu `double`.

V tabulce lze také vidět, že ESP32-S3 taktované na 40MHz je podstatně rychlejší při počítání s desetinnými čísly v porovnání ATSAMD21E18, které je taktováno na 48MHz. Na druhou stranu při práci s celými čísly je ESP32-S3 na tomto taktu značně pomalejší. Při dělení je dokonce pomalejší než Atmega328 (taktován na 16MHz). Pokud ESP32-S3 nataktujeme na 20HMz, je Atmega328 dokonce rychlejší ve všem, kromě sčítání necelých čísel. To mě v celku překvapilo, jelikož Atmega328 je 8bitový procesor, který je zároveň taktován na nižší rychlost (16MHz). ESP32-S3 na taktu 10MHz je ve všem značně pomalejší než jakýkoliv jiný mikrokontrolér. Jedinou výjimkou zde je sčítání necelých čísel, ve kterém je Atmega328 pomalejší.

2.2 Programovací jazyky

Programovací jazyky, které se daly v této práci použít, byly limitované faktem, že se v práci používájí mikrokontroléry. Kvůli této limitaci jsem se nakonec rozhodoval mezi vcelku novým jazykem **MicroPython/CircuitPython** a pro mikrokontrolory běžným jazykem **C/C++**.

MicroPython, jak již název napovídá, je speciální verze programovacího jazyka Python 3[12], který je optimalizovaný, aby mohl běžet na různých mikrokontrolérech.

[12]Python.org

MicroPython vytvořil v roce 2013 teoretický fyzik Damien George jako *crowdfundingový* (skupinově financovaný)[13] projekt na platformě Kikcstarter[14] společně s vývojovou deskou, na které mohl MicroPython běžet. Tento jazyk podporuje veliké množství moderních mikrokontrolérů, ale je limitován pouze na 32bitovou architekturu.

Veliká výhoda MicroPythonu (a Pythonu obecně) je rychlost, ve které se dá prototypovat. Oproti C/C++ je prototypování značně jednodušší a rychlejší, jelikož se člověk nemusí zajímat o různé maličkosti jako manuální správa paměti a podobné. Jeho obrovskou nevýhodou, kvůli které je často nepraktické ho používat ve finální verzi nějakého projektu, je jeho rychlost. Jelikož se jedná o interpretovaný jazyk, mikrokontrolér, na kterém tento jazyk běží, musí běžet interpreter, který zpracovává napsaný kód v reálném čase. To je ovšem značně náročnější než přímo běžet předkompilovaný kód jako v C/C++. Další nevýhodou je využití paměti. Člověk se sice nemusí o paměť starat, MicroPython jí ale potřebuje značně více pro běh stejného programu jako v C/C++. Tyto paměťové nároky jsou ještě vyšší, pokud člověk potřebuje pracovat s nějakými senzory. Pro tuto práci a vybraný mikrokontrolér by jediným problémem mohla být rychlost. Minimální požadavky na paměť mikrokontroléru jsou také podstatně větší než pro C/C++ a kvůli nim není možné tento jazyk použít na mikrokntroleru ATmega328 (Arduino Uno).

C/C++ je v dnešní době již standardní jazyk pro programování mikrokontrolérů (vestavěných zařízení). Tento jazyk se dá přeložit pro praktický každý mikrokontrolér, na který člověk narazí. Díky tomu, že se jedná o překládaný jazyk a ne o interpretovaný, je značně rychlejší než dříve zmíněný MicroPython. To je obrovská výhoda pro mikrokontroléry, které často nemají příliš mnoho výkonu, který by mohly postrádat na interpretaci. Další velikou výhodou je zpětná kompatibilita mezi verzemi jazyka a samotné stáří jazyka. Díky tomu se dá dohledat veliké množství užitečných knihoven, které je možné v práci použít.

2.2.1 Testování rychlostí jazyků

Oba jazyky byly otestovány na rychlost stejným způsobem jako jednotlivé mikrokontroléry. Pro testování byl použit mikrokontrolér ESP32-S3 s taktem 240MHz a pouze s jedním aktivním jádrem. Jelikož MicroPython nemá specifické velikosti typů, bylo otestováno pouze sčítání celého (`int`) a necelého čísla (`double`) a násobení necelého čísla.

[13] Wikipedia – Crowdfunding

[14] Kickstarter – Micro Python – Python for microcontrollers

Jazyk	int(+)	double(+)	double(/)
C/C++	0.71μs	25.013ms	254.758ms
MicroPython	282.003ms	756.932ms	777.815ms

Tab. 2.9: Rozdíl rychlostí jazyků

Z tabulky výsledků je zřejmé, že MicroPython je opravdu značně pomalejší než C/C++. Velice zajímavé je, že práce s necelými čísly trvala podobně dlouho bez ohledu na to, o jakou akci šlo. Počítání s celými čísly bylo více než 350000x pomalejší v porovnání s C/C++. MicroPython tak není prakticky možné použít pro tuto práci.

2.3 Senzory

V práci byly pro snímání pozice hráčovy ruky od nástroje použity různé senzory, které bylo nutné nejdříve zdokumentovat a následně otestovat jejich přesnost a vlastnosti.

2.3.1 Pasivní senzory

Pasivní senzory v tomto kontextu jsou senzory, se kterými pro získání hodnot nemusíme aktivně komunikovat (např. pomocí nějakého protokolu). Stačí pouze číst napětí, které senzor neustále vysílá na výstupní pin. Přesnost těchto čidel je do jisté míry ovlivněna také AD převodníkem, který se nachází uvnitř použitého mikrokontroléru.

Fotorezistor

Fotorezistor je speciální typ rezistoru, jehož celkový odpor se mění na základě intenzity dopadajícího světla na jeho citlivý povrch. Čím více světla na něj dopadá, tím menší je jeho odpor. Jedná se o čistě analogovou komponentu, které má zaměnitelný vstup i výstup (nezáleží na orientaci zapojení). Jedná se o polovodič, ale oproti dalším fotosenzitivním komponentám jako je například fotodioda nebo fototranzistor neobsahuje PN přechod. Fotorezistivita každého fotorezistoru je závislá na okolní teplotě a může se zásadně a nepředvídatelně měnit. Nedají se tudíž spolehlivě použít v měnícím se prostředí bez předem známého chování v dané teplotě. Jednou ze zajímavých vlastností fotorezistorů je také určitá odezva při změně okolního světla. Tomuto jevu se říká *rychlost obnovy odporu* (anglicky resistance recovery rate). Při přechodu ze tmy na světlo tak trvá zhruba 10ms, než dojde ke snížení odporu na

ustálenou hodnotu. Zatímco při přechodu ze světla do tmy může návrat na maximální hodnotu odporu trvat až vteřinu [10]. Tato vlastnost by do jisté míry mohla být výhodnou, jelikož by se touto odezvou dal eliminovat vstup, kdyby se hráči například třásla ruka.

Nehledě na závislost na okolní teplotě, veliký problém, který takovýto senzor přináší, je nekonzistentnost. Finální nástroj se může pokaždé nacházet na jiném místě, kde jsou jiné světelné podmínky. Fotorezistor by tak měl vždy jinou počáteční hodnotu. Toto by se dalo vyřešit pomocí počáteční kalibrace, kdy se vezme v potaz, kolik světla je v okolí a podle toho by se dopočítal výsledný vstup, aby byl vždy stejný. Opravdový problém ale nastává během hraní. Jelikož senzor neumí snímat hladinu světla pouze přímo nad sebou, ale v celém okolí, je možné senzor velice jednoduše nechtěně ovlivnit. Stačí se nad senzor třeba jenom více nahnout, nebo jakkoliv během hraní způsobit snížení celkové hladiny světla a celkový odpor již nebude odpovídat poloze hráčovy ruky nad rezistorem.

Senzor, který byl použit, je **NSL-06S53** od výrobce Silonex.

Tab. 2.10: NSL-06S53[15]

Teplotní limit	-60°C - +75°C
Energetická ztráta	50mW
Maximální napětí	320V
Odpor	20kΩ - 100kΩ

Tab. 2.11: Základní parametry [11]

Hallova sonda

Hallova sonda je senzor, které využívá tzv. *Hallův jev*[16] pro detekci přítomnosti a síly magnetického pole. Senzor většinou umí reagovat jak na pozitivní, tak i negativní magnetické pole. V klidovém stavu, kdy není naměřeno žádné magnetické pole, je výstupní hodnota ve středu maximální a minimální výstupní hodnoty. Čím silnější je poté magnetické pole v nějakém směru (negativní, pozitivní), tím nižší nebo vyšší je výstupní napětí. Tyto senzory mají velice dobrou rychlost odezvy, jelikož mají vzorkovací frekvenci kolem 10kHz. To dává zhruba 10us na vzorek. Oproti výše zmíněnému fotorezistoru není hallova sonda tak náchylná na změnu okolní teploty a je tak možné provádět přesné opakované měření.

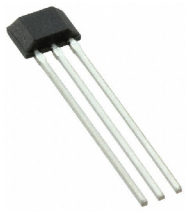
Tento senzor ale také doprovází řada problému, které ale mají možné řešení. Celkem zásadní problém je, že lidské tělo nevytváří magnetické pole. Možné řešení by bylo vytvořit třeba nějakou rukavici, ve které by byly magnety. Nebo by mohlo

[15] Převzato z webu Digikey.cz
[16] Wikipedia – Hallův jev

stačit mít neodymiový prstýnek na ruce. Je ale potřeba nějaká externí pomůcka pro detekci pozice ruky. Další z problémů je dosah. Tyto senzory mají obecně malý dosah (v řádu centimetrů), tudíž by použití jednoho senzoru určitě nestačilo. Jedním z řešení by mohlo být dát mnoho těchto senzorů do řady a následně by se pomocí interpretace napětí ze všech těchto senzorů dalo určit, u kterých senzorů se magnet (ruka s magnetem) právě nachází.

Využití toho typu senzorů je často jako bezkontaktní spínač, bezkontaktní otáčkoměr a nebo jako senzor pro měření proudu

Byl použit senzor **AH49Hz3-G1** od společnosti Diodes INC.



Tab. 2.12: AH49Hz3-G1[17]

Základní parametry	
Teplotní limit	-40°C - +105°C
Spotřeba	2mA (při 2V)
Maximální napětí	10V
Výstupní napětí	0.85V - 2.6V

Tab. 2.13: Základní parametry [12]

Infračervený senzor vzdálenosti

Tento senzor používá pro detekci vzdálenosti nějakého objektu infračervené světlo. Senzor obsahuje tři důležité součásti: infračervenou diodu, infračervenou fotodiodu a logický obvod pro vyhodnocení výstupního napětí. Dioda v nějaké frekvenci osvětluje prostor, který se před ní nachází. Vlastnosti této diody jsou předem známé a senzor tak ví, jakou intenzitu má očekávat jako maximální hodnotu. Fotodioda snímá intenzitu odraženého světla od cíle. Podle množství světla, které se odrazí na fotodiodu, fotodioda limituje proud, který skrz ní prochází. Čím více světla na ni dopadá, tím větší proud prochází. Vestavěný logický obvod následně pomocí této informace vypočítá napětí, které pošle na výstupní pin.

Největším problémem těchto senzorů je, že jsou závislé na intenzitě odraženého světla. Intenzita odraženého světla je zas dále závislá na vlastnostech objektu, od kterého se světlo odráží. Především je to tedy barva objektu a jeho tvar. Co se týče barvy, zcela bílý povrch má velice dobrou reflexivitu a jeho měření by tedy mělo být přesnější a konstantnější. Naopak černý povrch vstřebává světlo a měření tak může být značně nepřesné. Problémy ale dělají i vícebarevné objekty, obzvláště pokud se přechod mezi barvami nachází ve středu měřené oblasti. Jako řešení toho problému by bylo možné použít bílou rukavici, kterou by si hráč dal na ruku, aby se zajistila konzistence a nejlepší reflektivita. Lidská ruka ale má ale poměrně konzistentní

[17]Zdroj:

barvu a případný problém s odstínem by se dal vyřešit pomocí kalibrace. Co se týče tvaru, tak to je menší problém. Nerovné objekty mají tendenci většinu světla odrážet směrem od zdroje, takže by senzor měl celkově číst nižší hladinu odraženého světla. Lidská ruka je ale pro tento senzor poměrné rovný objekt, takže by to nemusel být problém. Problém ale může nastat, když je měřený objekt příliš malý a neodráží tak dostatek světla. U ruky by toto mohl být problém.

Senzor, který byl použit, je **GP2Y0A21YK0F** od výrobce SHARP.

Tab. 2.14: GP2Y0A21YK0F[18]

Teplotní limit	-10°C - +60°C
Spotřeba	50mA
Maximální napětí	7V
Výstupní napětí	-0.3V - +0.3V
Dosah	100cm - 550cm

Tab. 2.15: Základní parametry [13]

2.3.2 Aktivní senzory

Ultrazvukový senzor

Ultrazvukový senzor funguje podobně jako již dříve zmíněný infračervený senzor vzdálenosti. Podobně jako dříve zmíněný infračervený senzor vzdálenosti (podsekce 2.3.1) obsahuje tři základní součásti. Ultrazvukový vysílač (reproduktor), ultrazvukový přijímač (mikrofon) a vestavěný logický obvod pro interpretaci dat. Oproti všem prozatím zmíněným senzorům se z tohoto senzoru nedají přímo číst data. Nemá totiž analogový výstup, ale jeden digitální vstup (trig pin) a jeden digitální výstup (echo pin). Pro získání informací ze senzoru je potřeba na vstupní pin poslat signál log. 1 o délce 10μs. Senzor následně vyšle z vysílače osm 40kHz vln v řadě a čeká na jejich zachycení v přijímači. Po zachycení těchto osmi vln na přijímači vyšle na výstupní pin log. 1. Délka této log. 1 udává dobu v μs mezi vysláním vln a jejich zachycením. Senzor nám tím pádem nevrací vzdálenost od senzoru, ale dobu letu těchto ultrazvukových vln od senzoru k objektu a zpátky. Požadovaná vzdálenost v milimetrech se dá dopočítat pomocí následující rovnice:

$$2d = \frac{t_f}{v} \tag{2.1}$$

Kde t_f je doba letu ultrazvukových vln, v je přibližná rychlost zvuku ve vzduchu při 20°C v milimetrech za mikrosekundu. Následně se získaná vzdálenost vydělí dvěma, aby se dostala pouze vzdálenost k objektu.

[18]Převzato z webu Farnell.com

Jelikož se nejedná o optický senzor, není ovlivněn optickými vlastnostmi objektu, jehož vzdálenost měříme. To je veliká výhoda oproti všem ostatním senzorům, které byly v této práci použity. Jelikož se zde jedná o zvukové vlny, měření senzoru může být ovlivněno povrchovou úpravou nebo složením měřeného objektu. Proto je doporučeno, aby byl měřený objekt co nejrovnější a co nejméně vstřebával nebo propouštěl ultrazvukové vlny. To lidská ruka splňuje.

Nevýhodou senzoru je ale jeho dosah při snímání menších objektů. Jelikož se zvuk šíří na všechny strany od zdroje, je zde možnost například místo ruky hráče zaznamenat jeho hlavu, pokud je nad senzor nakloněný. Další nevýhodou je doporučená doba pro jednotlivý měřicí cyklus. Výrobce Sparkfun doporučuje 60ms mezi jednotlivými měřeními[19], což vychází na 16 měření za sekundu.

Použitý senzor je **HC-SR04** od výrobce MCM.

Tab. 2.16: HC-SR04[20]

Teplotní limit	-10°C - +60°C
Spotřeba	15mA
Maximální napětí	5.5V
Rychlost měření	60ms
Dosah	2cm - 400cm

Tab. 2.17: Základní parametry [14]

Senzor doby letu světla

Senzor doby letu světla, anglicky *time of flight sensor*, kombinuje do jisté míry vlastnosti ultrazvukového senzoru a infračerveného senzoru vzdálenosti. Stejně jako v ultrazvukovém senzoru se zde měří doba mezi vypuštěním nějakého signálu a jeho návratem. Ale stejně jako u infračerveného senzoru vzdálenosti se zde místo zvuku používá infračervené světlo a místo intenzity se zde měří pouze čas, kdy byla zaznamenána nějaká změna v intenzitě. Jedná se o jediný senzor, který používá pro komunikaci komunikační protokol **I2C** [15] (propojené integrované obvody; anglicky interconnected integrated circuits). Jedná se o synchronní komunikační protokol, který se používá pro komunikaci mezi jedním kontrolním zařízením (v našem případě mikrokontrolérem) a více cílovými zařízeními na stejné komunikační lince. Pro výběr cílových zařízení se používá adresování a každé zařízení tak musí mít vlastní specifickou adresu. V případě že má více cílových zařízení stejnou cílovou adresu, je nutné jejich adresu změnit, nebo zařídit, aby cílové zařízení komunikovalo pouze,

[19]Sprakfun – HCSR04

[20]Převzato z webu Sparkfun.com

kdy potřebujeme (například pomocí povolovacího pinu). Pro komunikaci se používají dvě obousměrné linky *SDA*[21] a *SCL*[22]. Na lince SCL vysílá kontrolní zařízení hodinový signál, který určuje rychlost komunikace. Na lince SDA se poté posílají data. Kontrolní zařízení si vyžádá nějaká data z cílového zařízené a to je při příštím hodinovém signálu pošle po stejné lince zpátky.

Velikou výhodou tohoto senzoru je, že by neměl být ovlivněn reflexivitou měřeného objektu, ani jeho tvarem a ani žádnými jinými vlastnostmi prostoru, ve kterém se senzor nachází. Senzor může pracovat v temné místnosti, jelikož produkuje vlastní zdroj světla a dokud povrch měřeného objektu nevstřebá téměř všechno světlo, měl by být senzor schopen určit jeho vzdálenost. Další výhodou je, že senzor při inicializaci provádí kalibraci a při měření sám do jisté míry zpracovává měřená data. Částečně také vyhodnocuje kvalitu měření a měl by tak být schopen určit, kdy se jedná o validní/nevalidní měření (i bez pomoci mikrokontroléru). [16]

Tento senzor také podporuje několik různých módů měření, které se odlišují podle rychlosti měření a následné přesnosti. V práci se používá nejrychlejší mód, při kterém trvá jedno měření 20ms, ale celková přesnost je oproti jiným módům nejhorší. [16]

V práci byl použit modul **GY-VL53L0X**, který obsahuje senzor **VL53L0X** od výrobce STMicroelectronics a všechny potřebné pasivní komponenty.

Tab. 2.18: VL53L0X[23]

Teplotní limit	-20°C - +70°C
Spotřeba	19mA
Maximální napětí	3.6V
Rychlost měření	20ms
Rychlost komunikace	400kHz
Dosah	15cm - 200cm

Tab. 2.19: Základní parametry [16]

Video kamera

Video kamera je v porovnání se vším ostatním velice komplikované zařízení a přináší mnoho problémů, které mají celkem složité řešení. První problém je samotné měření. Kamera samotná totiž nevrací žádné užitečné informace ohledně vzdálenosti nějakého objektu od snímače kamery. Vrací pouze data, která reprezentují, co se před kamerou právě nachází. Takže by bylo nutné použít buď nějaký kalibrovací objekt,

[21] Serial DAta line – linka na posílání dat
[22] Serial CLock line – linka na posílání hodinového signálu
[23] Převzato z webu LaskaKit.cz

u kterého je známa velikost (například *QR kód*[24]), a poté podle velikosti objektu se dá určit jeho vzdálenost od senzoru kamery. Nebo by bylo nutné použít 2 kamery pro stereoskopické snímání, ze kterého se pomocí složitých algoritmů dá získat přímo vzdálenost objektu od kamery. Ani jedno ale nepřipadá v úvahu, jelikož dnešní mikrokontroléry stále nemají dostatečný výkon na zpracování a počítání takovýchto dat v reálném čase (30 snímku za sekundu). Z těchto důvodů jsem se tedy rozhodl tento typ senzoru úplně vyloučit.

2.3.3 Testování senzorů

Všechny výše uvedené senzory, kromě video kamery, byly otestovány, aby se dal určit obecně nejlepší senzor, který se následně v práci použije. Všechny testy probíhaly stejným způsobem, ale podle typu senzoru se testy částečně měnily. Celkově se pro každý senzor provedlo dvanáct testů, každý v jiné výšce. Jedinou výjimkou byla hallova sonda, u které se provedly pouze dva testy, jelikož je její dosah značně limitován (pouze několik desítek milimetrů). Testovala se zde jak pravdivost, tak preciznost nekalibrovaných senzorů. Testování probíhalo následovně. Vzal se jeden senzor a ten se připojil k mikrokontroléru (v případě potřeby se použil externí obvod). Následně byl senzor položen pod sledovaný objekt. Následně se provedlo sto měření pod každým typem objektu. Pro optické senzory se jednalo o **bílý**, **černý** a **malý bílý** objekt. Pro ultrazvuk se jednalo o **plochý**, **látkový** a **malý plochý** objekt. Pro hallovu sondu se jednalo pouze o neodymiový magnet neznámé síly.

Fotorezistor

Fotorezistor byl otestován jako první a jeho výsledky ani nemá cenu zobrazovat. Ať byl použit jakýkoliv typ objektu, odpor fotorezistoru se prakticky vůbec neměnil (ať se výška objektu měnila jakkoliv). A je zde vcelku jednoduché vysvětlení, proč tomu tak je. Jak již bylo řečeno v podsekci 2.3.1, fotorezistor pouze čte okolní hladinu světla. Pokud se tedy zdroj světla nenachází přímo nad senzorem, objekt množství světla, které na senzor dopadá, příliš neovlivní. Jeho odpor tak zůstane prakticky stejný po celou dobu testovaní. Pokud ale zdroj světla přesuneme nad fotorezistor, jeho odpor se již měnit bude. Ale pouze pokud se nad fotorezistorem nachází nějaký objekt. Jeho velikost, barva ani vzdálenost ale nehrála žádnou roli. Stejně jako předtím byl odpor fotorezistoru stále stejný, dokud se nad ním nějaký objekt nacházel. A opět je zde jednoduché vysvětlení. Bez ohledu na typ objektu, objekt byl pokaždé dostatečně veliký, aby dokázal zastínit téměř všechno světlo, které dopadalo ze zdroje na fotorezistor. Z těchto důvodů je fotorezistor pro tuto práci naprosto nepoužitelný.

[24] Wikipedia – QR kód

Hallova sonda

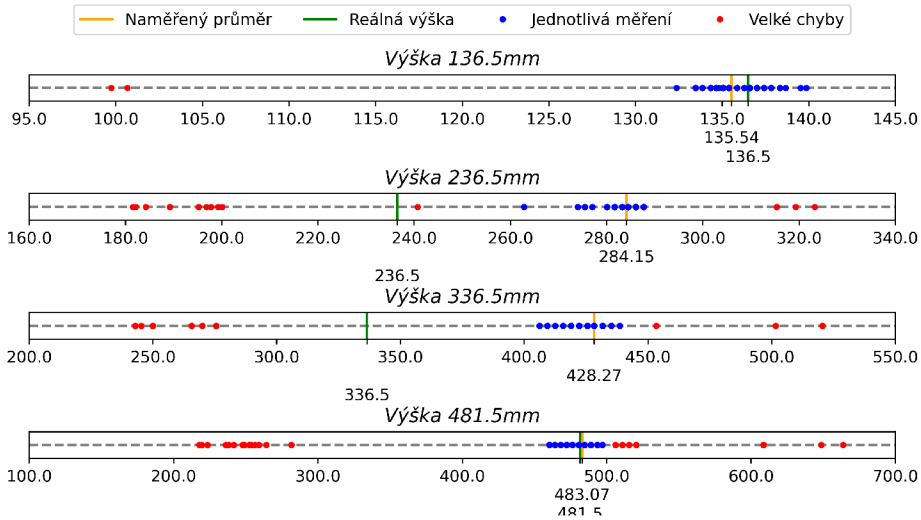
Hned po prvním zapojení a otestování funkčnosti jsem se rozhodl hallovu sondu dále ani pořádně netestovat. Jelikož je její dosah pouze několik desítek milimetrů a tento dosah (a přesnost) závisí na síle použitého magnetu, je tento senzor velice nepraktický pro použití v této práci.

Infračervený senzor vzdálenosti

Infračervený senzor vzdálenosti vrací jako jediný optický senzor napětí. Je tedy nutné z tohoto napětí získat přibližnou vzdálenost. Výstupní napětí senzoru ale není přímo úměrné vzdálenosti objektu od senzoru. Čím větší je vzdálenost objektu od senzoru, tím pomaleji se snižuje výstupní napětí na senzoru [13]. Je tedy nutné použít funkci, která tyto hodnoty linearizuje. Pro jednodušší práci se senzorem byla použita knihovna `SharpIR`[25], která implementuje rovnici 2.2 pro linearizaci hodnot ze senzoru.

$$l = 29.988 * V_{out}^{-1.173} \tag{2.2}$$

Proměnná l je vzdálenost a V_{out} je výstupní napětí senzoru. Mezi jednotlivými měřeními byla prodleva 40ms, jak doporučuje výrobce [13].

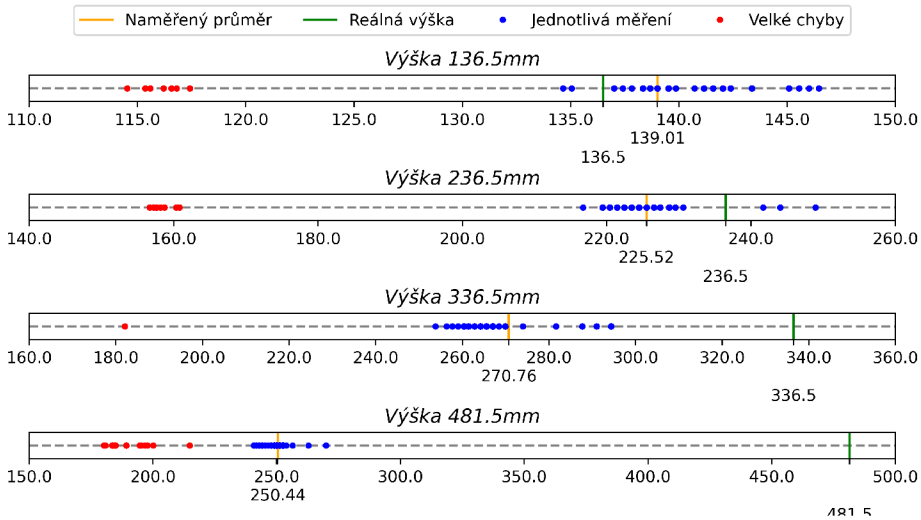


Obr. 2.1: Bílý objekt (350x500mm)

Z testování na bílém objektu je možné vidět, při poměrně krátké vzdálenosti je senzor celkem přesný. Hodnoty zde kolísají v přijatelném rozsahu. Průměrná hodnota těchto sta měření (hrubé chyby se do tohoto průměru nepočítají) je i poměrně blízko reálné hodnotě. To samé se ale bohužel nedá říct o dalších vzdálenostech. Při větších vzdálenost četnost hrubých chyb značně stoupla. Pravdivost senzoru zase naopak

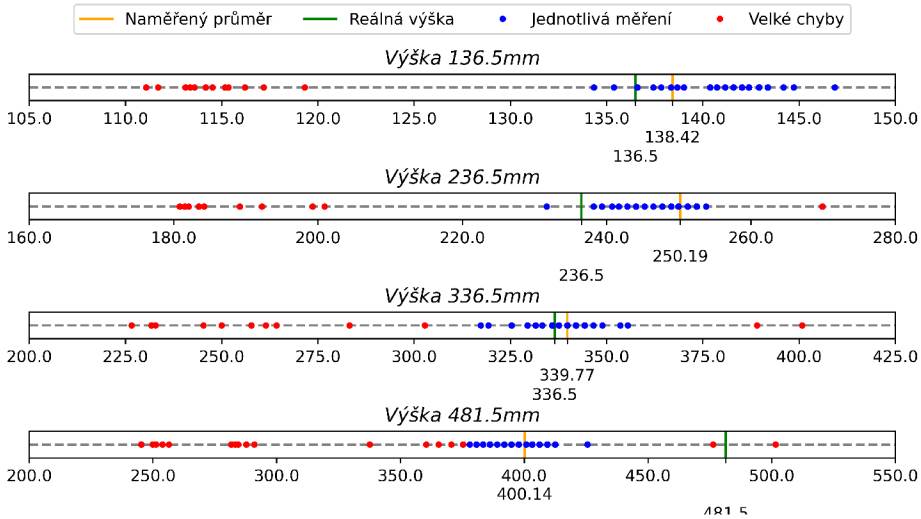
[25] GitHub – SharpIR

značně klesla. Jedinou výjimkou zde bylo měření největší vzdálenosti, kdy pravdivost senzoru byla opět v celku dobrá. Nicméně se zde stále vyskytovalo velké množství hrubých chyb. I po opakovaném testování se výsledky téměř nezměnily. Osobně si tak nedokážu vysvětlit drastický propad pravdivosti u měření 240mm a 350mm vzdáleností.



Obr. 2.2: Černý objekt (350x500mm)

Jak je z výsledků měření na černé objektu vidět, senzor je poměrně dost náchylný na barvu objektu, jehož vzdálenost měří. Oproti bílému objektu se zde již při nejkratší vzdálenosti objevilo podstatně více hrubých chyb. Přesnost senzoru byla při nejkratší vzdáleností také horší. Při vzdálenostech do 250mm se ale celková přesnost zlepšila. Největší problém ale nastal ve vzdálenosti nad 300mm. Při těchto vzdálenostech je měření vzdálenosti černého objektu prakticky nemožné. Hodnota se nikdy nepřiblíží skutečné vzdálenosti. Spíše naopak je naměřená vzdálenost stále menší, čím dále se objekt nachází. U černých objektů tak není možné tento senzor použít pro měření větších vzdáleností.



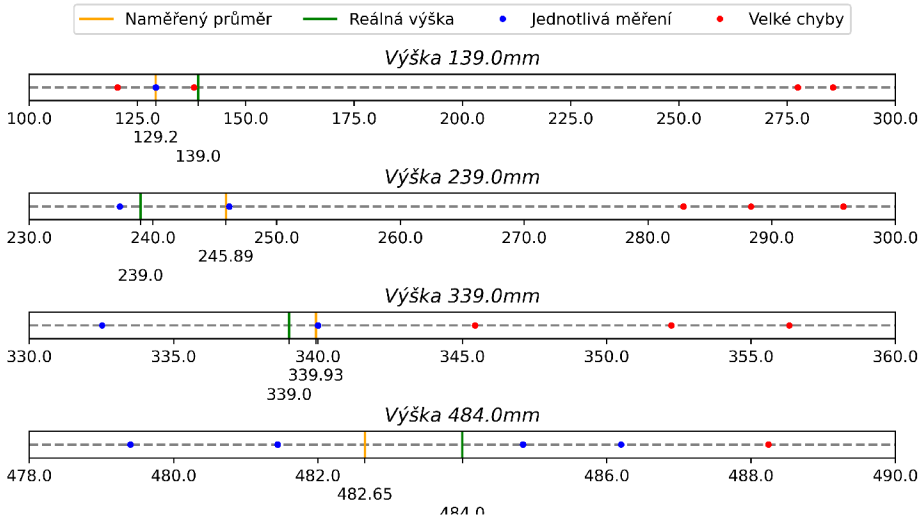
Obr. 2.3: Malý bílý objekt (90x180mm)

Z testů na malém objektu je zřejmé, že podobně jako u černého objektu má senzor problém při měření větších vzdáleností. Kromě největší vzdálenosti je přesnost senzoru u malého objektu celkově nejlepší v porovnání s ostatními objekty. Oproti bílému objektu se zde, stejně jako u černého objektu, objevilo podstatně více hrubých chyb při měření nejkratší vzdálenosti.

Ze všech testů na senzoru je vidět, že do vzdálenosti zhruba 250mm je senzor možné použít na měření různobarevných objektů i objektů různých velikostí (s případnou kalibrací). Nad tuto vzdálenost již je patrné, že od černého nebo malého objektu neodráží dostatek světla, aby senzor správně určil vzdálenost objektů.

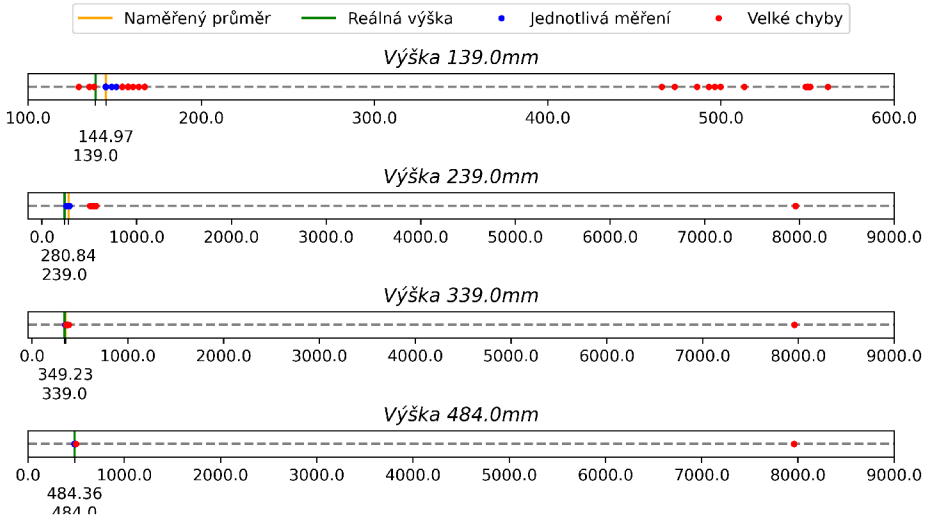
Ultrazvukový senzor

Jelikož ultrazvukový senzor je jako jediný akustický senzor, byly pro jeho testování místo různobarevných objektů použity objekty s různou povrchovou úpravou. Vyobrazené výsledky jsou zde pouze pro plochý a malý objekt. Důvodem je, že při testování látkou pokrytého objektu senzor nevracel naprosto žádná data. Senzor se tak nedá použít na měření vzdáleností od objektů, které jsou pokryté nějakou hustší látkou (jako například rukavice na hráčově ruce), která vstřebá téměř všechny ultrazvukové vlny. Mezi jednotlivými měřeními byla prodleva 60ms, jak doporučuje výrobce [14]. Jedná se tak o senzor s nejmenší vzorkovací frekvencí. Výsledná měřená vzdálenost byla vypočítána pomocí rovnice 2.1.



Obr. 2.4: Plochý objekt (350x500mm)

Výsledky měřené na plochém objektu ukazují, že ultrazvukový senzor je v celku přesný. Počet hrubých chyb byl poměrně malý. Je zde ale vidět problém, že průměrná naměřená výška byla v jedné polovině testů větší a ve druhé polovině menší než reálná výška. Řádná kalibrace by tak mohla být celkem složitá, jelikož se nedá určit „směr“ chyby.

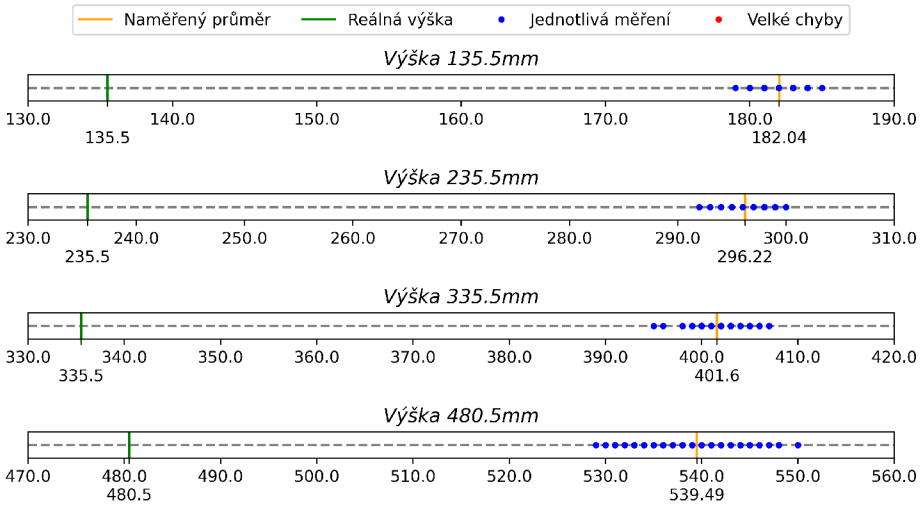


Obr. 2.5: Malý objekt (90x180mm)

Z výsledků je možné vidět, že přesnost senzoru se při změně velikosti objektu příliš nezměnila. Co je ale podstatně horší je četnost hrubých chyb. Všechny chyby kolem hodnoty 8000mm jsou hrubé chyby způsobeny tím, že senzor vůbec nezaznamenal vracející se vlny. Z grafů to sice není možné vidět, ale těchto hrubých chyb bylo celkově téměř třikrát více než použitelných hodnot. Získání použitelné hodnoty tak trvalo v průměru 180ms mezi jednotlivými měřeními.

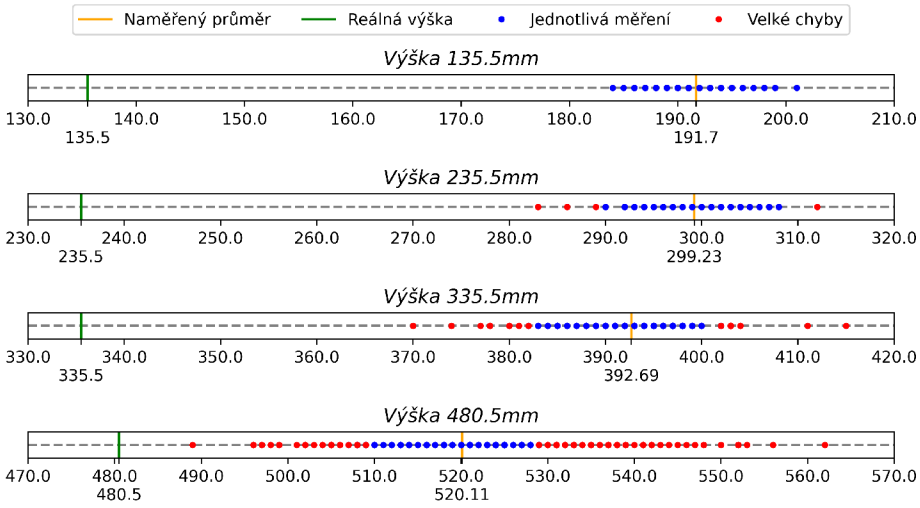
Senzor doby letu světla

Senzor doby letu světla jako jediný používá pro komunikaci I2C protokol. Pro jednodušší práci se senzorem byla použita knihovna `vl53l0x-arduino`[26]. Díky ní je možné jednoduše nastavit nejrychlejší (kontinuální) měřicí režim na senzoru se vzorkovací frekvencí 20ms.



Obr. 2.6: Bílý objekt (350x500mm)

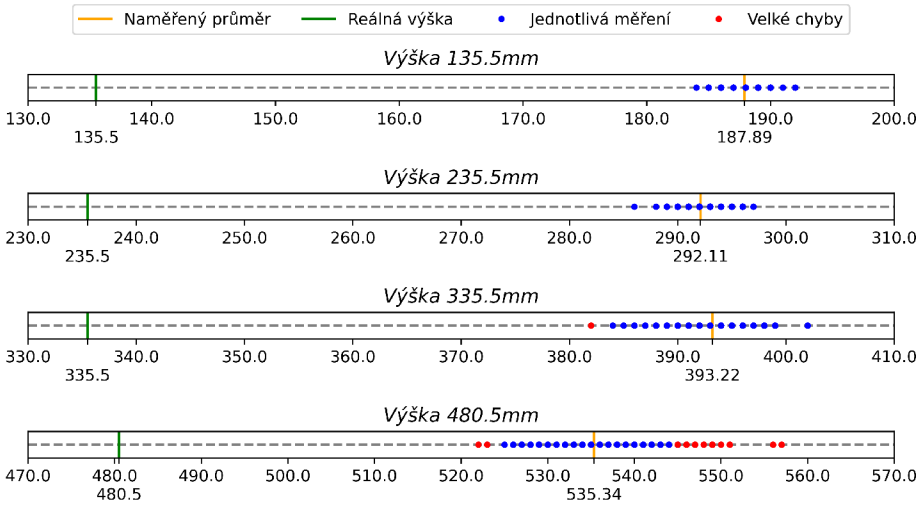
Z výsledků měření na bílém objektu je vidět, že se při měření neobjevily žádné hrubé chyby a to ani při měření největší vzdálenosti. Čím větší ale byla reálná vzdálenost, tím větší byl rozptyl vrácených hodnot. V průměru byla naměřená vzdálenost o 58mm větší než reálná vzdálenost.



Obr. 2.7: Černý objekt (350x500mm)

[26] GitHub – vl53l0x-arduino

Při testování na černém objektu se konečně objevily nějaké hrubé chyby. Obzvláště při největší testované vzdálenosti se zde objevilo opravdu veliké množství spíše velkých, než hrubých chyb. Celkově byla naměřená hodnota o 54mm větší než reálná. Je tak tedy vidět, že barva objektu ovlivňuje hlavně preciznost senzoru. Pravdivost senzoru se příliš nezměnila.



Obr. 2.8: Malý bílý objekt (90x180mm)

Z výsledků na malém objektu je vidět, že u prvních dvou testů na nejkratších vzdálenostech jsou výsledky velice podobné jako u velkého bílého objektu. Zbylé dva se spíše přibližují testům na černém objektu. Je zde menší množství velkých/hrubých chyb než u černého objektu, i tak se zde ale stále nachází.

Ze všech provedených testů je zřejmé, že naměřená vzdálenost senzorem doby letu světla je podstatně více konzistentní v porovnání s ostatními senzory. Hodnoty, které senzor vracel, byly v průměru 54-58mm nad reálnou vzdáleností. V průměru tak senzor

2.4 Dodatečný hardware

Kromě senzorů a mikrokontroléru byl pro finální nástroj použit ještě externí hardware pro nastavení nástroje a zvukový výstup.

Pro zobrazení a ovládání základního nastavení nástroje byl vytvořen menu systém. Ten byl následně zobrazen na OLED displeji a pro jeho ovládání byl zvolen rotační enkodér. Displej komunikuje s mikrokontrolérem pomocí protokolu I2C. Rotační enkodér potřebuje pouze 3 digitální piny pro připojení.

Pro finální zvukový výstup byl použit modul **GY-PCM5102** s čipem PCM5102 od výrobce Texas Instruments. Jedná se o modul, který komunikuje pomocí proto-

kolu I2S[27]. To je speciální protokol přímo vytvořený pro posílání zvukového signálu mezi zařízeními. Modul tak od mikrokontroléru dostane I2S signál, který zpracuje a rovnou vyšle do připojeného reproduktoru.

2.5 Výsledky testování

Jako finální vývojová deska byla zvolena deska **NodeMCU-32S**, jelikož mikrokontrolér, který obsahuje (ESP32-S3), je značně nejvýkonnější ze všech dostupných a testovaných mikrokontrolérů (viz podsekce 2.1.4). Jako jazyk bylo zvoleno **C/C++** a to také z důvodu rychlosti. Z testů v podsekci 2.2.1 je vidět, že MicroPython je znatelně pomalejší, což by v této práci mohl být problém. Ze senzorů byl vybrán senzor doby letu světla **VL53L0X**. Ze všech testovaných senzorů byly jeho měřené hodnoty nejvíce konzistentní a barva objektu, ani jeho velikost ho příliš neovlivnila (viz podsekce 2.3.3). Také se u něho objevilo malé množství hrubých chyb. Jednalo se tak o nejspolehlivější dostupný senzor.

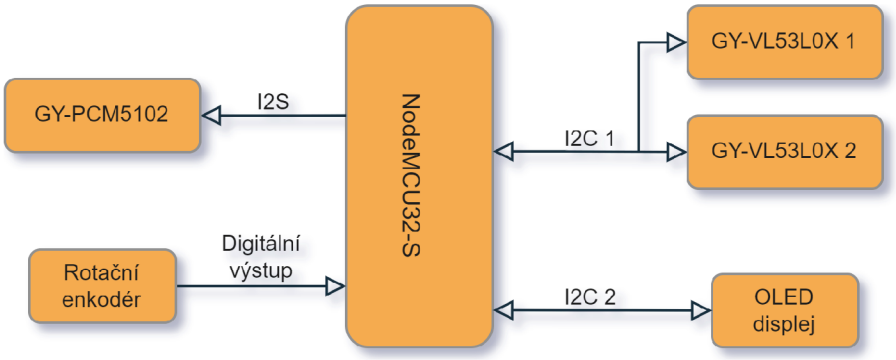
[27] Wikipedia – I2S

3 Návrh a realizace řešení

Tato kapitola se zabývá návrhem hardwaru, jeho sestavením a následně realizací softwarové části a celkovým otestováním funkčnosti práce.

3.1 Blokové schéma finálního nástroje

Součástí práce bylo vytvořit fyzický nástroj. Nejdříve je tedy nutné navrhnout možná spojení jednotlivých komponent nástroje. To je vyobrazeno pomocí blokového schématu níže.

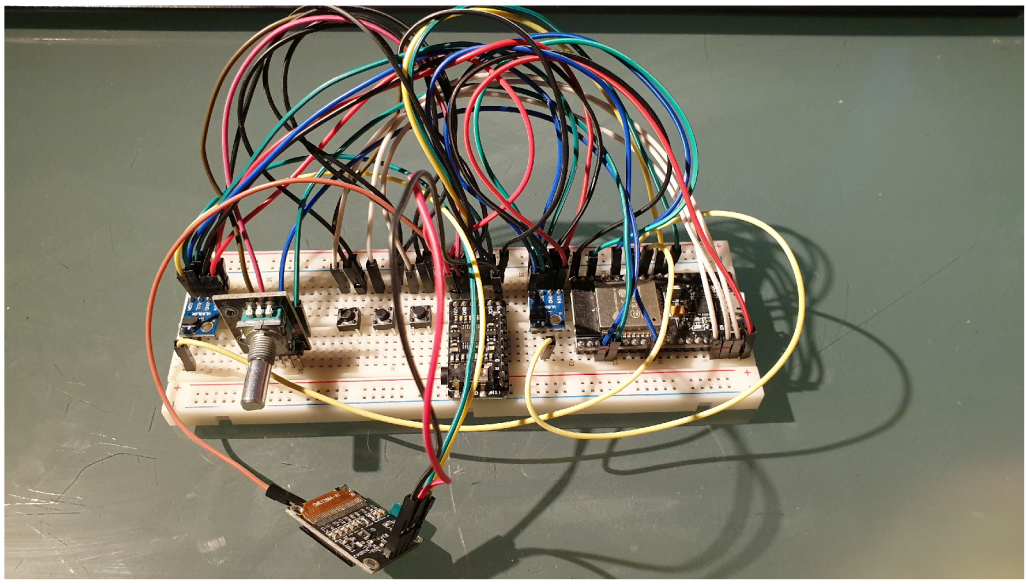


Obr. 3.1: Blokové schéma nástroje

3.2 Realizace nástroje

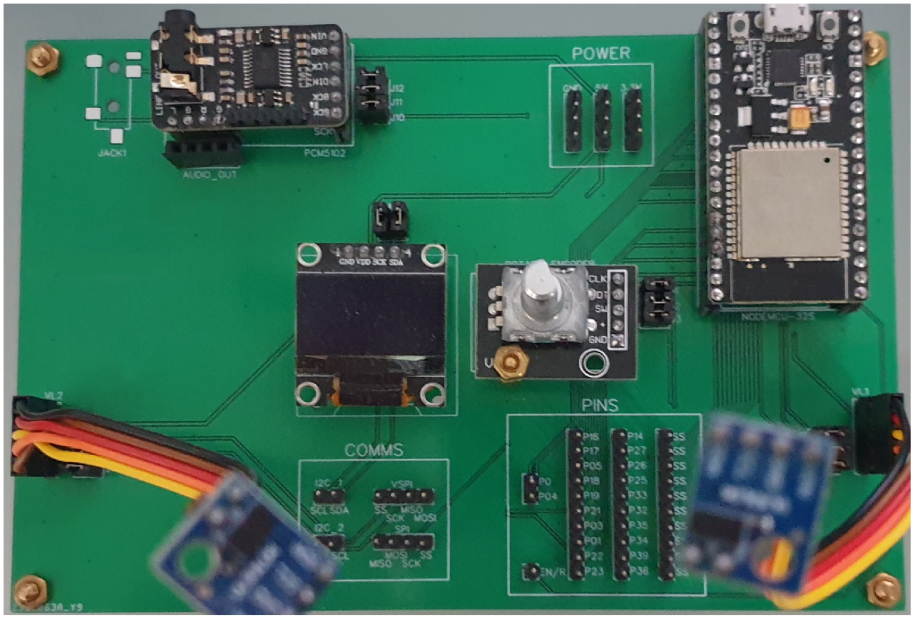
Po vytvoření blokového schématu je nutné toto schéma převést do reálného světa. Je tedy nutné hardware, který byl vybrán v sekci 2.5, propojit, aby mezi sebou mohly jednotlivé komponenty správně komunikovat. Senzory které byly vybrány používají pro komunikaci protokol I2C, je tak tedy nutné je napojit na správné piny u vývojové desky.Displej také používá pro komunikaci protokol I2C. Díky tomu, že protokol I2C využívá adresování, je možné použít jednu linku pro všechny tři zařízení. To byl aspoň můj původní odhad. Pro ovládání kontextové nabídky na displej byl použit rotační enkodér, který vyžaduje tři volné piny. Může se ale jednat o jakékoliv piny a nemusí mít žádnou specializaci. Pro výstup zvuku byl vybrán modul, který komunikuje pomocí protokolu I2S. Výstup tohoto protokolu je na desce omezen pouze na dva specifické piny. Jsou to piny vestavěného D/A[1] převodníku a není tak možné použít žádné jiné piny. První prototyp byl postaven na nepájivém poli a vypadal následnovně.

[1] Wikipedia – D/A převodník



Obr. 3.2: První prototyp na nepájivém poli

Toto byl první návrh, který ale nakonec nebylo možné použít. Při jeho testování se totiž narazilo na problém s displejem. Stále si nejsem jistý, čím byl tento problém způsoben, ale povedlo se mi nalézt řešení. Na displeji se z nějakého důvodu zobrazovaly rozbité/nevalidní znaky a různé čáry. To mohlo být způsobeno buďto rozbitým displejem, nebo problémem na I2C lince. Při testování samotného displeje se žádné podobné problémy neobjevily (displej byl testován samostatně a na více mikrokontrolérech). Jelikož I2C protokol používá při komunikaci adresování, neměl by zde být ani žádný problém s přeslechy. Nicméně ani po dlouhém testování a ladění v této kombinaci (dva senzory a displej) se mi nepodařilo tento problém odstranit. Jediné řešení, které tak zbylo a které také pomohlo, bylo použít další I2C linku pro komunikaci přímo s displejem. Deska podporuje dvě I2C linky současně na specifických pinech. Pomocí softwaru je ale možné tyto piny změnit a vytvořit i více linek (na úkor rychlosti). Senzory tak tím pádem měly vlastní I2C linku a displej také. Jelikož toto byl naštěstí jediný „hardwarový“ problém, bylo možné z tohoto upraveného designu udělat finální prototyp.



Obr. 3.3: Finální prototyp

3.3 Implementace a generování výsledného tónu

Pro jednodušší práci s displejem, senzory a rotačním enkodérem byly použity externí knihovny `U8g2`[2], `VL53L0X`[3] a `ESP32Encoder`[4]. Pro programování byl zvolen rámec Arduino (anglicky arduino framework). Je tak nutné mít dvě základní funkce **setup** a **loop**. Funkce `setup` se provede pouze jednou a používá se pro inicializací různých tříd a nastavení. Funkce `loop` se provádí donekonečna a volá se po funkci dokončení funkce `setup`. Při implementaci nástroje byl použil lehký operační systém pro vestavěná zařízení **FreeRTOS**[5]. Díky němu bylo možné rozdělit celkovou implementaci do několika základních úloh, mezi kterými operační systém postupně přepínal. Díky použití těchto úloh, které samy běží donekonečna, je možné nechat dříve zmíněnou funkci `loop` prázdnou (nebo ji dokonce i pomocí FreeRTOS odstranit). V práci se poté jedná o tři základní úlohy:

- Úloha pro generování konfiguračního menu a jeho ovládání `menuTask`
- Úloha pro interpretaci dat ze senzorů `sensorTask`
- Úloha pro generování tónu a jeho výstup `playTask`

Pro posílání dat mezi jednotlivými úlohami jsem se rozhodl použít globální proměnné. Je to z toho důvodu, že za běžného provozu do jedné proměnné může zapisovat pouze jedna úloha. Použití globálních proměnných by tak mělo být dostatečně bezpečné. Je zde ale také možnost použít nástroje, které nám dává OS, jako například semafory nebo speciální fronty.

[2] GitHub – olikraus/u8g2
[3] GitHub – pololu/vl53l0x-arduino
[4] GitHub – madhephaestus/ESP32Encoder
[5] freertos.org

3.3.1 Konfigurační menu (`menuTask`)

Pro jednoduší konfiguraci nástroje bylo v práci implementováno menu, které se zobrazuje na OLED dipleji. Je ovládáno pomocí rotačního enkodéru a používá se pro konfiguraci/modifikaci parametrů nástroje. Je zde možné tak změnit například typ vlny, kterou nástroj generuje, nebo upravit detekční a aktivační vzdálenost senzorů. Celé menu je implementováno jako samostatná knihovna `MenuLib`.

Pro nastavení, zobrazení a ovládání tohoto menu byla v práci vytvořena úloha `menuTask`. V ní se nejprve vygeneruje struktura menu. To se provádí pomocí pomocné funkce `addByName` z menu knihovny. Pomocí této funkce je možné jednoduše přidat jednotlivé vstupy/možnosti v menu. Následně je zde nekonečná smyčka, ve které se čte vstup z rotačního enkodéru a podle něho se menu ovládá.

3.3.2 Interpretace dat ze senzorů (`sensorTask`)

Pro získání a interpretaci dat ze senzorů byla v práci vytvořena speciální úloha `sensorTask`. Zde se jako první inicializují oba senzory. Jelikož se jedná o dva shodné senzory, mají oba stejnou I2C adresu. Nejdříve je tedy potřeba změnit adresu aspoň jedno ze senzorů. To se provádí pomocí pomocné funkce `registerSensor`. Jako argumenty se jí dá povolovací pin senzoru, nová I2C adresa a objekt senzoru. Následně je zde inicializace fronty pro průměrování dat ze senzorů a nekonečná smyčka. V té následně probíhá získávání dat a jejich interpretace. V práci zprvu byla použita čistá data ze senzorů. Jak již bylo zmíněno v podsekci 2.3.2, senzory při inicializaci provádí kalibraci a při měření samy provádí validaci získaných dat. Toto tak není potřeba provádět a implementovat. Problém zde ale je, že vyhodnocení měření trvá v nejrychlejším režimu minimálně 20ms a jedná se o nejméně přesný typ měření. Toto tak vedlo k problémům, kdy při přímém použití hodnot ze senzorů byla změna výsledné frekvence velice skokovitá. To je problém, jelikož je to velice nepříjemné na poslech. Z tohoto důvodu bylo potřeba implementovat interpolaci mezi získanými hodnotami ze senzorů. Jedná se zde o takzvaný *klouzavý průměr*[6]. Ten spočívá v tom, že průměry z předchozích výpočtů se použijí ve výpočtu následujícího průměru.

```
1  int array_len = 3;
2  deque<int> smoothing_array = {0, 0, 0};
3  while(true) {
4      distance = sensor.readRangeContinuousMillimeters() + CALIBRATION_OFFSET;
5      smoothing_array.pop_front();
6      for (int i = 0; i < array_len - 1; i++)
7          distance += smoothing_array[i];
8      distance /= array_len;
9      smoothing_array.push_back(distance);
10 ...
```

[6] Wikipedia – klouzavý průměr

Nejdříve se inicializuje fronta pro ukládání průměrů. Její velikost jsou 3 prvky. Pro tuto velikost jsem se rozhodl z toho důvodu, že při větších velikostech měla změna výstupního tónu znatelnou prodlevu vzhledem k pohybu hráčovy ruky. Následně přichází na řadu nekonečná smyčka. V ní se nejprve načtou data ze senzoru. Odstraní se první prvek z fronty. K hodnotě ze senzoru se přičtou všechny prvky z fronty. Nakonec se získá průměr těchto hodnot a ten se uloží na konec fronty. Toto se provede pro oba senzory a při každém novém čtení ze senzoru. Hodnota, která tímto vznikne se tedy zprvu rychle přibližuje k hodnotě získané ze senzorů. S postupným časem ale rychlost tohoto přiblížení zpomaluje. Výsledné hodnoty se tak stále přibližují hodnotě získané ze senzorů, ale jsou značně plynulejší než čistá data ze senzorů.

Jelikož při kontinuálním módu je přesnost senzoru nejhorší a z testů je vidět (viz podsekce 2.3.3), že čím je objekt dále, tím je větší počet chyb, rozhodl jsem se vstupní rozsah nástroje limitovat. Výchozí limit byl nakonec zvolen mezi 50-800mm nad senzorem. Tento limit se dá přímo na nástroji změnit a hráč si ho tak může jakkoliv upravit až do maximálního rozsahu senzoru (0-2000mm) [16]. Pokud se hráčova ruka nachází pod/nad limitem, bude získaná hodnota ze senzoru ignorována a místo ní se použije hodnota daného limitu. Podobným způsobem je implementováno utišení nástroje. Tento limit je nastaven na 1200mm a pokud senzor vrátí hodnotu, která je nad touto hranicí, nástroj se utiší. Tento limit je také možné nastavit přímo na nástroji.

3.3.3 Generovaní tónu (`playTask`)

Pro výsledný výstupní tón byla zvolena standardní vzorkovací frekvence 44100kHz, díky které je možné vygenerovat všechny člověkem slyšitelné frekvence.

Vyhledávací tabulka versus generování za běhu

V této práci se oproti řešením ze sekce 1.3 nepoužívá vyhledávací tabulka s předvygenerovanými hodnotami vln, ale hodnota vlny v daném kroku je přímo generována za běhu programu.

Při použití vyhledávací tabulky je nutné předem vygenerovat všechny požadované typy vln. Jelikož se toto musí dělat jako speciální krok, je potřebné si určit velikost tabulky, což určuje rozlišení možné vlny. Čím větší je tabulka, tím více zabírá místa v paměti, ale tím větší je také rozlišení uložené vlny. Vyhledávací tabulka se ale generuje pouze pro jednu frekvenci. Pokud chceme tabulku použít pro generování jiné frekvence, je nutné upravit krok, kterým se tabulka prochází. Pokud tak chceme získat například dvakrát vyšší frekvenci, než je frekvence vlny uložené v tabulce, je nutné dvojnásobně zvětšit krok procházení tabulky. Čím větší je tak chtěná frekvence, tím méně vzorků pro generování dané vlny máme k dispozici. Je

tak nutné zvolit dostatečnou velikost tabulky, aby nedošlo k problému kdy, frekvence bude mít například pouze jeden vzorek v tabulce.

Generování vlny za běhu je do jisté míry velice podobné jako použití vyhledávací tabulky. Místo toho aby se vlna vygenerovala ve speciálním kroku a následně uložila, generuje se pokaždé jenom chtěná hodnota v daném kroku. Při generování se zde ale postupuje stejně jako při generovaní u vyhledávací tabulky. Vybere se velikost *virtuální* tabulky. Díky tomu, že se ale tabulka nebude ukládat, je možné zvolit arbitrárně velkou hodnotu. Případně je i možné velikost tabulky měnit za běhu programu. Pomocí této velikosti, požadované frekvence a výstupní vzorkovací frekvence se následně vypočítá velikost kroku, o který se v této virtuální tabulce posuneme.

$$step = \frac{f_T * S_t}{f_{SR}} \tag{3.1}$$

Následně se vygeneruje hodnota požadované vlny pro nově získanou pozici ve virtuální tabulce. Pokud by nás následující krok dostal mimo tabulku, odečte se velikost tabulky od nynější pozice a pokračuje se dál. Výhodou toho přístupu je, že není nutné tabulku nikde ukládat. Je také možné měnit její velikost za běhu. Tím se dá docílit zajímavého efektu. Pokud totiž velikost tabulky zmenšíme, zmenší se i možný počet generovatelných frekvencí vlny. Velikou nevýhodou tohoto přístupu je ale výpočetní náročnost. Při generování složitějších vln (např. sinusovek) je výpočet velice hardwarově náročný a je nutné, aby byl mikrokontrolér dostatečně výkonný na tyto výpočty. Z tohoto důvodu byla v této práci snaha vybrat co nejvýkonnější mikrokontrolér.

Generovaní různých vln

Finální nástroj umí generovat čtyři různé druhy vln, mezi kterými je za běhu možné volně přepínat.

První z nich je čistá sinusovka. Pro získání hodnoty v dané pozici v tabulce je použita následující rovnice

$$val = \sin(2\pi * pos) * Amp \tag{3.2}$$

Pomocí této funkce se v dané pozici `pos` získá příslušná hodnota vlny v radiánech. K pozici se při každém kroku přidá velikost kroku pro postup tabulkou (viz sekce 3.1). Výsledná hodnota se poté vynásobí s hodnotou amplitudy. Za jednu sekundu se tak vygeneruje sinusová vlna o dané frekvenci a amplitudě.

Druhá vlna je obdelníková vlna. Zde stačí generovat podle hodnoty střídy jeden časový úsek hodnotu amplitudy a druhý časový úsek negativní hodnotu amplitudy. Střída 0 % generuje přímou vlnu s negativní amplitudou. Střída 100 % naopak generuje přímou vlnu s kladnou hodnotou amplitudy. 50% střída generuje obdelníkovou vlnu, která je polovinu času kladná a následující polovinu záporná.

Třetí je trojúhelníková vlna. U této vlny je možné změnit její sklon, takže se z ní může stát pilová vlna. Sklon je možné měnit od 0 po 100. Sklonění 0 % vytvoří pilovou vlnu, která je nejvíce skloněna doleva. Sklonění 100 % je naopak pilová vlna s největším skloněním doprava (zrcadlový opak 0% sklonu). Sklon 50 % je trojúhelníková vlna. Pro generování těchto vln je tak nutné zjistit, do jaké pozice v tabulce je nutné stoupat a od kdy je poté potřeba klesat.

Čtvrtá a poslední vlna se snaží aspoň částečně napodobit zvuk thereminu[7]. Jedná se o sinus sinusu a je tak tedy možné použít rovnici z první vlny a částečně ji upravit.

$$val = \sin\left(2\pi * pos + \sin\left(2\pi * pos\right)\right) * Amp \tag{3.3}$$

Výsledkem je jednoduchá skloněná sinusovka. Zvukově je tato vlna odlišná od čisté sinusovky a připomíná zvuk, který si člověk u thereminu představí. Toto je ale limit, kterého je vybraný mikrokontrolér schopný. Při složitějších výpočtech se úloha nestihne včas vykonat a celý program tak spadne.

Audio výstup (I2S)

Jako posledním krokem při generování tónu je odeslání získané hodnoty vlny do reproduktoru. Od toho má mikrokontrolér ESP32-S vestavěný I2S ovladač. Po jeho správném nastavení mu následně stačí předat získanou hodnotu vlny a on ji odešle na I2S modul.

Tento I2S ovladač je nastaven ve funkci `setup` (viz úvod této sekce). Jedná se o téměř výchozí nastavení tohoto ovladače pro jeden kanál převzaté a upravené z oficiální dokumentační stránky[8].

```
1  i2s_config_t i2sConfig = {
2          .mode                 = I2S_MODE_MASTER | I2S_MODE_TX,
3          .sample_rate          = SAMPLE_RATE,
4          .bits_per_sample      = I2S_BITS_PER_SAMPLE_16BIT,
5          .channel_format       = I2S_CHANNEL_FMT_ONLY_LEFT,
6          .communication_format = I2S_COMM_FORMAT_I2S,
7          .intr_alloc_flags     = ESP_INTR_FLAG_LEVEL1,
8          .dma_buf_count        = 4,
9          .dma_buf_len          = 64,
10         .use_apll             = true
11     };
```

Vzorkovací frekvence ovladače je nastavena na 44100Hz. Rozlišení jednotlivých kanálů je 16bitů. Používá se pouze jeden kanál a to levý. Jsou využity čtyři DMA zásobníky, každý o velikosti 64 vzorků.

[7] thereminworld.com – Tvar vlny thereminu Burns Pro

[8] ESP32 docs – Inter-IC Sound

Ovladač funguje následovně. Při předání hodnoty ovladač tuto hodnotu uloží do *DMA*[9] vyrovnávací paměti. Po zaplnění této paměti nebo po uplynutí časového okna, které je stanoveno vzorkovací frekvencí, odešle ovladač získaná data do externího I2S modulu (viz sekce 2.4). I2S modul tento signál následně zpracuje a vyšle přímo do reproduktoru.

[9] Wikipedia – DMA

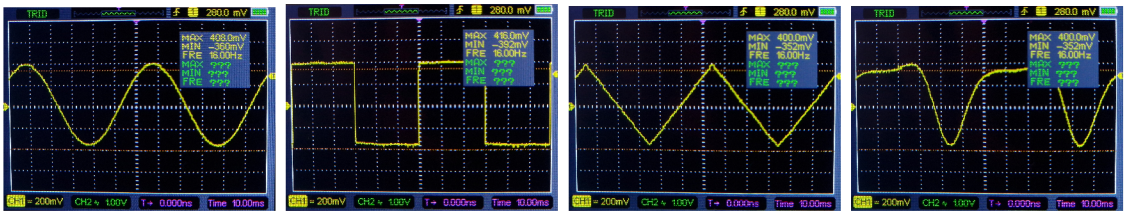
4 Testování nástroje a zhodnocení řešení

Pro otestování kompletně sestaveného prototypu nástroje se softwarem jsem si připravil několik testů, které měly ověřit nejpodstatnější funkcionality nástroje.

Podrobné testování senzorů bylo popsáno v podsekci 2.3.3. Zde se tak jednalo pouze o testování, jestli hodnoty ze senzorů správně mění výsledný generovaný tón. Stačilo tak prototyp zapnout a vyzkoušet, jestli se frekvence a amplituda mění, podle toho jak vysoko byla moje dlaň nad senzory. Vše zde fungovalo v pořádku.

Ovládání konfiguračního menu bylo také poměrně jednoduše otestováno. Pomocí rotačního enkodéru jsem se pokoušel navigovat v menu a nastavit jednotlivé parametry. Poté jsem otestoval jejich změnu tím, že jsem nástroj aktivoval a pozoroval, jestli se změnila například nástrojem generovaná vlna. Vše zde také fungovalo v pořádku. Příklad jak vypadá konfigurační měnu je možné najít v příloze P.1.

Konfigurační menu bylo následně použito i pro testování fixně generovaného tónu. Je zde totiž možné pro testování nastavit tón, který má nástroj generovat a nebude zpracovávat data ze senzorů. Generování tónu je také nejpodstatnější funkce celého nástroje. Pro toto testování byl nástroj nastaven na generování stabilního tónu o frekvenci 16Hz (nejbližší hodnota od noty C0), 440Hz (A4 – komorní A[1]) a 7902Hz (B8). Následně byly otestovány jednotlivé vlny, které nástroj generuje, aby se ověřila jejich skutečná frekvence. Pro ověření byl použit osciloskop Hantek 2D42[2], který snímal signál, který vycházel z I2S modulu do reproduktoru.



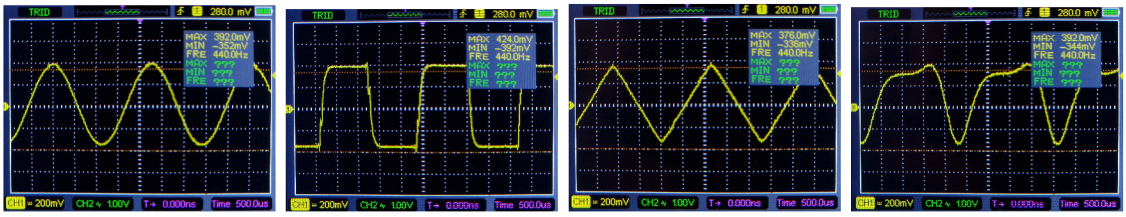
(a) Sinusová vlna (b) Obdelníková vlna (c) Trojúhelníková vlna (d) Vlastní vlna

Obr. 4.1: Test generování 16Hz vln

Z testů je vidět, že generování 16Hz tónu (nejbližší celé číslo od 16.35Hz) není pro nástroj problém a všechny vlny jsou velice dobře rekonstruovány.

[1] Wikipedia – A440

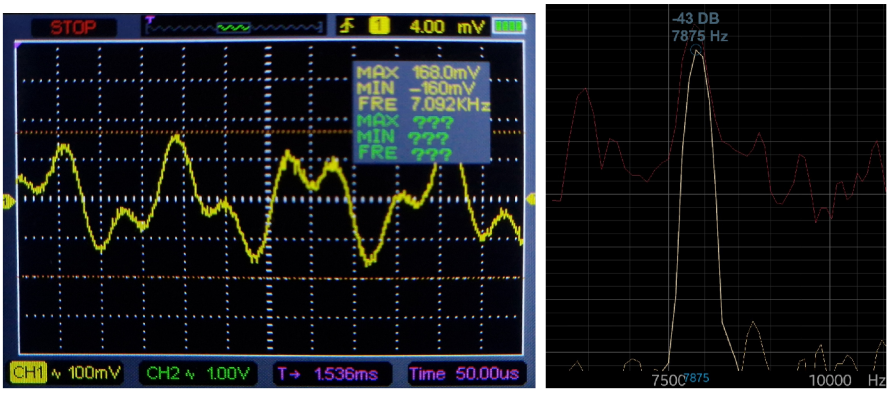
[2] Hantek.com



(a) Sinusová vlna (b) Obdelníková vlna (c) Trojúhelníková vlna (d) Vlastní vlna

Obr. 4.2: Test generování 440Hz vln (komorní A)

Z testů na frekvenci 440Hz je vidět, že generovaná obdelníková vlna již začíná mít menší problémy s počtem vzorků. Je ale stále dobře rekonstruovatelná a frekvence odpovídá generované frekvenci.



(a) Sinusová vlna–osciloskop (b) Sinusová vlna–Spectroid

Obr. 4.3: Test generování 7902Hz vln

Z posledních testů je vidět, že nástroj generuje již příliš málo vzorků, aby se daly vlny rekonstruovat na osciloskopu. Všechny vlny vypadaly prakticky stejně na tomto osciloskopu (proto je zde vyobrazena pouze sinusovka). Pro jistotu jsem se tak rozhodl využít aplikaci Spectroid[3], na které jsem testoval, jestli tón, který slyším, má požadovanou frekvenci. Na obrázku je sice vidět jiná frekvence než požadovaná. To je ale způsobeno tím, že aplikace má příliš veliký krok mezi frekvencemi, které dokáže zobrazovat, aby se požadovaná frekvence 7902Hz dala vyobrazit. Při nejhorším je tak frekvence zhruba 30Hz mimo. To je ale stále velice dobrý výsledek. Nástroj tak dokáže generovat i nejvyšší hudební notu, negeneruje ale dostatek vzorků pro rekonstrukci vlny v osciloskopu.

Celkově je z výsledků vidět, že při generování nejvyššího tónu se již negeneruje dostatečné množství vzorků pro spolehlivou rekonstrukci na osciloskopu. Jak je ale

[3]Google Play – Spectroid

vidět ze spektrální analýzy z telefonu, tón o frekvenci 7902Hz je generován. Cělkově se ale dá říct, že nástroj zvládá generovat všechny testované noty a je tak možné ho bez větších problémů používat.

Závěr

Celkově si myslím, že nástroj splnil požadavky práce a opravdu se jedná o digitální verzi hudebního nástroje theremin. Celkově jsem s nástrojem velice spokojen. Je zde ale několik míst pro případná rozšíření. Jedním z nich je udělat nástroj plně bezkontaktní. To by znamenalo pomocí senzorů naimplementovat možné ovládání menu systému, aby nebylo potřeba používat rotační enkodér (a tím se nástroje dotýkat). Dalším možným rozšířením by mohlo být ukládání záznamu, jak hráč hraje. Pro toto by bylo nejspíše potřeba přidat SD kartu pro ukládání těchto dat a data v nějakém formátu na kartu zapisovat. Nejsem si ale jistý, jestli by na to měl mikrokontrolér přebytečný výkon.

Nástroj samozřejmě také není bez problémů. Jak je vidět z testování finálního prototypu, při hraní noty B8 o frekvenci 7902Hz to vypadá, že se neprodukuje dostatek vzorků pro spolehlivou rekonstrukci na osciloskopu. Zvukově je tato frekvence ale generována. Další problém může nastat při odpojení senzoru. V práci se předpokládá, že senzory budou fungovat po celou dobu hraní. Není zde žádná kontrola, pokud by se senzor například odpojil během hraní. Stejně tak zde není žádná kontrola, že I2S modul opravdu funguje. Pokud tak z nástroje nejde žádný zvuk, může se například jednat o mrtvý I2S modul a hráč to nemá jak zjistit. Zajímavějším problémem je, že nástroj někdy úplně zamrzne. Pokud toto nastane, nástroj přestane reagovat na jakékoliv vstupy od hráče a bude vydávat tón, který hrál v době zamrznutí. Jediným řešením je zde nástroj resetovat. Při testování a ladění jsem zjistil, že mikrokontrolér naprosto přestane reagovat a ve chvíli zamrznutí zůstanou všechny výstupy ve stavu jakém jsou. Nepovedlo se mi najít způsob, jak z mikrokontroléru v tomto stavu získat nějaká data a je tak téměř nemožné zjisti, co se vůbec s nástrojem děje. Naštěstí je tento problém velice vzácný a za celou dobu práce jsem na tento problém narazil ani ne desetkrát.

Literatura

1. CAMPBELL, Murray; GREATED, Clive; MYERS, Arnold. *Musical Instruments: History, Technology, and Performance of Instruments of Western Music*. 1. vyd. Velká Británie: Oxford University Press, 2004. ISBN 0-19–8816504.
2. SACHS, Curt. *The History Of Musical Instruments*. New York: W.w.norton a Company, 1940. ISBN 0-393-02068-1.
3. GLINSKY, Albert. *Theremin: ether music and espionage*. 1. vyd. Spojené Státy Americké: University of Illinois Press, 2000. ISBN 0-252-02582-2.
4. *Theremin* [online]. [B.r.] [cit. 2022-03-11]. Dostupné z: `https://en.wikipedia.org/wiki/Theremin#Operating_principles`.
5. *Theremini user's manual*. Severní Karolína, 2014. Dostupné také z: `https://api.moogmusic.com/sites/default/files/2018-09/Theremini_Manual_6_25.pdf`. Uživatelský manuál.
6. PALENČÁR, Rudolf; VDOLEČEK, František; HALAJ, Martin. Nejistoty v měření I: vyjadřování nejistot. *Automa – časopis pro automatizační techniku*. 2001, roč. 2001, č. 7–8. ISBN 9771210959006.
7. *megaAVR® Data Sheet* [online]. 2020 [cit. 2022-04-05]. Č. DS40002061B. Dostupné z: `https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf`. Rev.: B.
8. *SAM D21/DA1 Family* [online]. 2021 [cit. 2022-04-05]. Č. DS40001882G. Dostupné z: `https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21DA1-Family-Data-Sheet-DS40001882G.pdf`. Rev.: G.
9. *ESP32 – S3 Series Datasheet* [online]. 2022 [cit. 2022-04-05]. Dostupné z: `https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf`. v1.0.
10. *Photoresistor* [online]. [B.r.] [cit. 2022-04-04]. Dostupné z: `https://eepower.com/resistor-guide/resistor-types/photo-resistor/#`.
11. *Data Sheet, NSL-06S53 TO-18 Hermetic Photocell* [online]. 2008 [cit. 2022-04-04]. Č. 103856. Dostupné z: `https://www.farnell.com/datasheets/1162377.pdf`. Rev.: 01.
12. *Linear Hall Effect IC* [online]. 2013 [cit. 2022-04-04]. Dostupné z: `http://www.socle-tech.com/doc/IC%5C%20Channel%5C%20Product/Sensors/Distance%5C%20Measuring%5C%20Sensor/Analog%5C%20Output/GP2Y0A710K0F_spec.pdf`. Rev.: 1.2.

13. *GP2Y0A21YK0F* [online]. 2006 [cit. 2022-04-04]. Č. E4-A00201EN. Dostupné z: https://www.pololu.com/file/0J85/gp2y0a21yk0f.pdf.

14. *User's Manual* [online]. 2013 [cit. 2022-04-04]. Dostupné z: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/view. V1.0.

15. *I2C MANUAL* [online]. 2003 [cit. 2022-04-04]. Č. AN10216-01. Dostupné z: https://www.nxp.com/docs/en/application-note/AN10216.pdf.

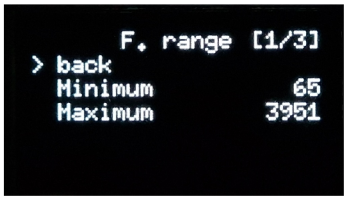
16. *World's smallest Time-of-Flight ranging and gesture detection sensor* [online]. 2021 [cit. 2022-04-04]. Č. DocID029104. Dostupné z: https://www.st.com/resource/en/datasheet/vl53l0x.pdf. Rev.: 3.

Přílohy

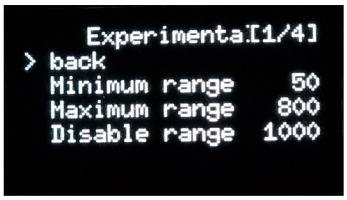
P.1 Konfigurační menu

(a) Hlavní menu

(b) Nastavení



(c) Frekvenční rozsah



(d) Experimentální možnosti

Obr. 4: Příklady menu systému

P.2 Struktura DVD

```
/
├──examples/
│   ├── menu.mp4      - Ukázka ovládání konfiguračního menu
│   └── play.mp4      - Ukázka ovládání a hraní na hotový nástroj
├──pcb/
│   ├──README.txt     - Návod na otevření souborů desky v EasyEDA
│   └──TherESPin.zip - Design desky (EasyEDA)
├──source/
│   ├──theremin/      - zdrojové soubory práce (VSCode + PlatformIO)
│   └──latex.zip      - zdrojové soubory textu (Latex)
└──README.txt         - Návod na instalaci a zprovoznění práce
```