VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Jaroslav Jablonický

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ZAŘÍZENÍ S MODELÁŘSKÝMI SERVY
SYSTEM WITH MODEL SERVOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jaroslav Jablonický**
AUTHOR

VEDOUCÍ PRÁCE — **Prof. Dr. Ing. Pavel Zemčík,**
ADVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Jablonický Jaroslav**
Program: Informační technologie
Název: **Zařízení s modelářskými servy**
System with Model Servos
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte způsob řízení modelářských serv a možnosti jejich ovládání počítačem.
2. Navrhněte jednoduchý strojek sestrojený z modelářských serv a způsob jeho ovládání (jezdící model, robotek, mechanickou ruku apod.)
3. Popište a diskutujte možnosti ovládání modelářskými servy.
4. Navržený systém implementujte a demonstrujte.
5. Diskutujte dosažené výsledky a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 7. listopadu 2018

ABSTRAKT

Táto práca je zameraná na skúmanie spôsobu ovládania modelárskych serv počítačom. Tiež bude slúžiť ako prehľad alebo príručka pre začiatočníkov. Práca obsahuje návrh jednoduchého robota, presnejšie sa jedná o jednoduchú robotickú ruku, ale najmä obsahuje návrhy ovládania z rôznych platforiem za použitia rôznych prostriedkov. V závere sú odporúčania pre jednotlivé platformy.

KLÍČOVÁ SLOVA

servo, Raspberry Pi, Arduino, FITKIT, PWM, robotické rameno, micro maestro

ABSTRACT

This work is focused on exploring how to control modeling servos by computers. It will also serve as an overview or a guide for beginners. The work includes the design of a simple robot, more precisely it is a simple robotic hand, but in particular it contains control designs from different platforms using different means. Finally, there are recommendations for each platform.

KEYWORDS

servo, Raspberry Pi, Arduino, FITKIT, PWM, robotic arm, micro maestro

JABLONICKÝ, Jaroslav. *Zařízení s modelářskými servy*. Brno, 2019, 58 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, . Vedoucí práce: Prof. Dr. Ing. Pavel Zemčík,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zařízení s modelářskými servy“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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

PODĚKOVÁNÍ

Rád by som poďakoval vedúcemu práce, Prof. Dr. Ing. Pavlovi Zemčíkovi, za ochotu, tlač demo modelu a rady, ktoré my poskytoval počas vypracovávania práce. Ďalej by som chcel poďakovať Pánovi Miroslavovi Martákovi za požičanie a pomoc s testami na osciloskope. Ďakujem.

Brno . . . .

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Každú chvíľu sa nájde nejaký nadšenec, ktorý si povie, že by chcel vytvoriť autonómneho robota, vylepšiť svoj RC model alebo si zmodernizovať život izbou, ktorá ho pozdravný odhrnie záves, zmení intenzitu svetla, či zamkne izbu. A práve všetky tieto veci majú spoločnú potrebu vyvinúť fyzické akciu na základe impulzu, či už z ovládača alebo senzoru. Týmto fyzickým činiteľom najčastejšie býva nejaký motorček alebo priamo modelárske servo zvládajúce viac než sa len točiť dokola ale aj nastaviť presnú polohu svojho vychýlenia. Nasledovne človek začne hľadať spôsob a možnosti ich ovládania. Existuje veľa roztrúsených článkov dokonca aj video príručiek pre ovládanie ale vždy predpokladajú určité porozumenie látke alebo sa jedná o platformu, ktorá mu nie je po chuti alebo ju nevlastní. A práve táto práca sa pokúša tieto články zjednotiť a poskytnúť záujemcovi základy pre tvorbu.

Čiže v práci sa budem zaoberáť skúmaním rôznych riešení z viacerých platforiem ich testovaním a vyhodnotením ich splnenie schopnosti ovládanie modelárskych serv. Medzi skúmané platformy zahrniem hlavne platformu o ktorej nie je veľa článkov a to je personálny počítač. Nasledovne zahŕňam aj klasické populárne platformy Raspberry Pi a Arduino aj keď platformu Arduino len okrajovo bez testov. Ako poslednú platformu zahrniem FITKIT, túto platformu zahŕňam hlave ako študijnú pomôcku na VUT FIT a jej možné použitie v cvičeniach. Obsahom cvičení by mohlo byť jednoduchá generácia signálu doplnením kódu buď z MCU alebo z FPGA. Všetky tieto skúmané riešenia by mali byť schopné generovať signál na ovládanie modelárskych serv i keď nemusia byť použiteľné pre všetky aplikácie.

Ďalej pre niektoré tieto skúmané a testované riešenia vyrábam demo kód a pripojím ich na demo aplikáciu. Táto aplikácia bude robotické rameno z modelárskych serv.

V Prvej kapitole je krátke zhrnutie teórie k servám ich parametre a ovládanie ale i pojmov ako pulzná modulácia signálu. Nasledovanou popisom jednotlivých platforiem. Každý popis platformy dube obsahovať všeobecné informácie o platforme jej vstupno-výstupné rozhranie a tiež popis existujúcich riešení. Tretia kapitola obsahuje podobné rozdelenie ako druhá a to platforma mnou vybrané riešenia, výsledky ich testov. Ďalej bude obsahovať popis demo aplikácie jej vývin a konštrukciu. Na záver bude práca ešte obsahovať rýchly prehľad riešený a základných informáciách o nich.

2 Serva a ich riadenie počítačovými systémami

Táto kapitola zhŕňa popis súčastného stavu techniky serv a ich riadenie pomocou počítačov. Práca nie je encyklopédiou ale poskytuje len základný popis a informácie pre pochopenie obsahu.

2.1 Modelárske Serva

Známe tiež pod názvom servo, sú motory u ktorých sa dá nastaviť presná poloha natočenia osy. Pre naše účely používame modelárske servá pre rádiom ovládané modely alebo robotiku. Klasický príklad použitia modelárskych servo je ich pripojenie k rádiovému prímaču od ktorého dostáva riadiaci signál pre určenie natočenia hriadele.

Typicky sa servá skladajú z elektromotoru na jednosmerý prúd, ozubených kolies reprezentujúcich prevodovku a z riadiacej elektroniky, ktorej základom je mikrokontróler.

Prevodovka plní rovnakú funkciu ako aj v aute to jest nastavení sily a rýchlosti. Jej ďalšou funkciou je točenie spätnovazobného potenciometru na základe uhlu nastaveného motorom.

Riadiaca elektronika spracováva vstupný riadiaci signál, z ktorého vyhodnotí požadované natočenie s odporom spetnoväzobného potenciometru a potočí motor požadovaným smerom.

Riadiaca elektronika sa dá detailnejšie rozložiť na monostabilný preklápací obvod, potenciometer, sčítačku a mostíkový spínač. Princíp spracovania vstupného signálu je nasledovný: impulz zopne preklápací obvod ktorý na základe súčasného stavu indikovaného potenciometrom generuje impulz s obrátenou polaritou oproti vstupnému. Oba tieto impulzy idú do sčítačky, z ktorej ide výsledný impulz do mostíkového spínača pre zosilnenie aby bol schopní roztočiť motorček a tým cez prevodovku pohnúť páčkou serva a súčasne potenciometrom pre spätnú väzbu predstavujúcu nový aktuálny stav. Tento proces neskončí lebo potenciometer znovu zopne preklápací obvod. Výsledkom je znovu pulz z obrátenou polaritou k vstupnému ale s dĺžkou bližšou vstupnému signálu. Ukončovacou podmienkou na zastavenie motorku je vyrovnanie dĺžok porovnávaných impulzov.

Digitálne servá síce používajú digitálnu riadiacu elektroniku, ale vstupný riadiaci signál majú rovnaký ako analógové čím je zabezpečená kompatibilita medzi nimi.

[1] source:`https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Exploded_Servo.jpg/1280px-Exploded_Servo.jpg`

Obr. 2.1: Rozobrané servo[1]

Všeobecne sú tieto servá schopné spracovať aj vyššiu frekvenciu signálu, sú presnejšie a silnejšie s čím je ale často spojená aj vyššia spotreba oproti analógovým.

Delenie serv

Ďelenie serv aj keď je tak nie je štandardizované takže sa od neho niektorý výrobcovia jemne odkláňajú.

1. **Podľa typu riadiacej elektroniky**
   - **Analógové**
   - **Digitálne**
2. **Podľa veľkosti**– servá sa štandarde delia do šiestich kategórií popísaných v tabuľke 2.1 nižšie. Delenie okrem veľkosti zahŕňa aj váhu serva. Kategórie sa môžu líšiť v závislosti od výrobcu, napríklad zjednotením viacerých kategórií do jednej.
3. **Podľa spôsobu pohybu hriadele**
   - **pozičné** – jedná sa o klasické najpredávanejšie servo, kde natočenie zodpovedá určitému stupňu z rozsahu, stupeň určuje riadiaci signál
   - **kontinuálne** – riadiaci signál určuje rýchlosť točenia, či už jedným alebo druhým smerom, typickým využitím je pohon modelu
   - **lineárne** – obsahuje zmenenú prevodovku aby pohyb osy tvoril priamku a nie časť kružnice, najčastejším použitím je vysúvanie podvozkov mode-

lov lietadiel

4. **Podľa materiálu prevodovky**
    - **Plastové** – všetky ozubené kolieska sú plastové, výhodov je najnižšia váha a cena
    - **Kovové** – prevody sú kovové, týto servá nesú označenie MG (Metal Gears), výhodou je ich odolnosť a pevnosť, sú drahšie a obvykle mávajú vyšší výkon lebo plastové sa pri vyššom výkone opotrebujú rýchlejšie
    - **Hybridné** – kombinácia predošlích dvoch kategórií, označujú sa HG (hybrid Gears)

Velkosť	**Váha** [g]	**Šýrka** [mm]	**Výška** [mm]	**Použitie**
Nano	<8	7,5	18,5	Mikro lietadlá, Mikro helikoptéry
Sub-Micro	8 – 16	11.5	24	Lietadlá s rozpätím 1400mm, 200-450 helikoptéry
Micro	17 – 26	13	29	až do 2000mm rozpätie krídiel, 500 helikoptéry
Mini	27 – 39	17	32.5	600 helikoptéry
Standard	40 – 79	20	38	>2000mm rozpätie aj turbínové lietadlá, 700-800 helikoptéry
Large	>=80	>20mm	>38mm	Lietadlá gigantickej veľkosti a tryskové lietadlé

Tab. 2.1: Rozdelenie serv podľa veľkosti

Odklonenia sú predovšetkým spojené s veľkosťou serva, určitý výrobcovia spájajú niektoré kategórie do seba.

2.2 Zapojenie a ovládanie RC serv

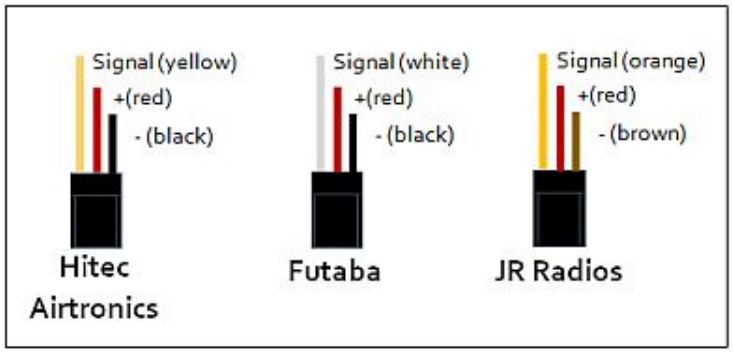
Pri zapojení sa môžeme stretnúť s tromi farebnými schémami.

Posun je určený na základe šírky vstupného obdĺžnikového signálu o frekvencii 50Hz, t.j. nástupná hrana každých 20ms. Riadiaci signál je popísaný pod sekciou PWM nižšie. Typický rozsah pohybu modelárskych serv je 0-180° niektoré servá aj keď majú rozsah pohybu 180°, tak za použitia len bezpečného signálu dosahujú pohybu 120°, dalo by sa tiež udávať pohyb od kľudovej/neutrále polohy serva, ktorej

[2]source:`https://www.swanrobotics.com/wp-content/uploads/2015/03/servoconnect.jpg`

Farebná schéma		
Zem	Napájanie	Signál
Hnedá	Červená	Oranžová
Čierna	Červená	Žltá
Čierna	Červená	Biela

Tab. 2.2: Tabuľka farebných schém káblov serv



Obr. 2.2: Zapojenie serva[2]

zodpovedá 1,5ms dlhý pulz, pre mnou použité servá to robí -60° až 60° , 1 až 2 ms dlhé pulzy respektíve.

Odporúča sa požívať dĺžky pulzov z rozsahu 1-2ms z dôvodu fyzického dorazu na servách. Pri dosiahnutí tohto dorazu sa môže stať jedna z nasledujúcich vecí. Za prvé sa zvýši odoberaní prúd a hrozí zhorenie riadiacej elektroniky. Za druhé ak sú použité lacnejšie servá s platovými prevodmi tak hrozí zlámanie zubov a s tým spojenou stratou presnosti natočenia.

Parametre serv

Medzi najčastejšie uvádzané parametre partia rozmery serva, jeho hmotnosť, uhol a typ prevodovky. Ostatné parametre väčšina predajcou neuvádza.

- **Rozmery** – Fyzické rozmery serva, bližší popis v kapitole s delením serv vyššie.
- **Hmotnosť** – Váha serva, súvisí s veľkosťou, materiálom prevodov, či typom motoru
- **Ťah** – Udáva váhu ktorú je servo schopné potiahnuť pri určitej dĺžke ramena. Najčastejšie sa udáva kilogram na centimeter. To značí počet kilogramov ktoré posunie keď má servo rameno o dĺžke 1cm.
- **Rýchlosť** – Udáva rýchlosť vychýlenia ramena. Udávané je to v sekundách

na stupne. Referenčné stupne sú 45 a 60 stupňov. Napríklad mnou použité servo GO-13MG má rýchlosť 0,14s/60°, čiže za 0,14s prejde 60°.

- **Typ prevodovky** – reprezentuje očakávanú odolnosť voči poškodeniu, popis v kapitole s delením serv vyššie.
- **Napájanie**
- **Uhol** – Operačný rozsah o ktorý je možné pohnúť hriadeľov, udávaný v stupňoch. Bežne sa jedná o 90 a 180° uhol.
- **Impulz pre strednú polohu** – Určuje dĺžku pulzu, ktorým uvedieme servo do neutrálnej polohy. Najčastejšie je to pulz o dĺžke 1,5ms, niekedy sa ešte udávajú aj pulzy pre krajné polohy t.j. 1-2ms alebo 1,25-1,75ms podľa operačného rozsahu.
- **Rozlíšenie** – Definuje presnosť umiestnenia hriadele po prijatí riadiaceho signálu. Typicky udávané v stupňoch. Bežné servá majú rozlíšenie v rozsahu 1 až 10 stupňov.

Ako zaujímavosť by som zmienil, že kovové prevodovky serv majú podtyp určený pre modely lietadiel najmä kvôli odolnosti a malej hmotnosti, týmto typom sú titánijové prevodovky so skratkou TG.

2.3 Pulz s moduláciou PWM

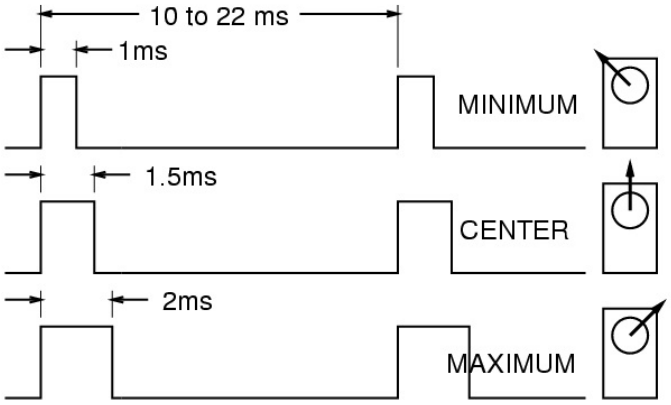
Moduláciou sa všeobecne rozumie proces ovplyvňovania nosného signálu. Nosný signál je ovplyvnený modulačným signálom. Typicky sa moduluje jedna z trojice charakteristík signálu. Tieto charakteristiky sú Fáza, amplitúda a frekvencia.

PWM alebo Pulse width modulation označenie metódy úpravy periodického signálu zmenením šírky impulzu v závislosti od vstupných veličín na vstupe za účelom prenosu informácie, či regulácie elektrického výkonu s vysokou efektivitou. Táto efektivita je daná tým, že regulátor je v ideálnych podmienkach vždy buď úplne otvorený alebo úplne uzavretý. Z tohto dôvodu v ňom nevznikajú tepelné straty na odporovom prvku spôsobené úbytkom napätia. Pre pochopenie základného princípu je treba si definovať pár pojmov. Minimálnym a maximálnym napätím rozumieme hodnotu napätia v lokálnom extréme. Amplitúda reprezentuje rozdiel maximálneho a minimálneho napätia. Cyklus je interval vlny v ktorom nájdeme jednu opakujúcu sa časť signálu. Perióda je čas za ktorý prebehne jeden opakujúci sa cyklus. Frekvencia je obrátená hodnota periódy. Pracovná doba reprezentuje dobu po ktorú je signál aktívni alebo ľudovo hore. Keď si nastavíme maximálne napätie na 5V s minimom v 0V s pracovnou dobou na 50% tak po pripojení voltmetra získame 2,5 V. Asi najjednoduchšie časté využitie tejto vlastnosti je s pripojením LED či už malého počtu alebo celého pásu a ovládania intenzity, či ľudovo ako veľmi svietia.

PWM na riadenie serv

Pre väčšinu ľudí sú pojmy ako PWM frekvencia a šírka impulzu niečo čomu nemusia plne rozumieť aby mohli upravovať RC modely ako hoby lebo tieto pojmi sú zabalené v rôznych predpripravených prípravkoch.

Pulzne šírkovo modulovaný signál je prenášaný cez signálový vodič (oranžový, žltý alebo biely záleží od výrobcu), tento pulz zodpovedá za nastavenie správnej polohy serva. Použití riadiaci PWM signál má frekvenciu 50Hz, čo zodpovedá 20ms perióde, pulzy sa môžu líšiť dĺžkou/šírkou pulzu tieto šírky sú štandardizované čiže všetky modely pracujúce s frekvenciou 50Hz by mali vychýliť svoje rameno do rovnakého uhlu. Veľmi často sa v špecifikáciách objavuje číslo 1,5ms (1500-1520µs) špecifikujúce šírku impulzu na vycentrovanie či nastavenie serva do neutrálnej polohy, ktorá sa nachádza v presnom strede medzi krajnými bodmi. Tento údaj sa špecifikuje z dôvodu, že niektoré serva môžu využívať kratší impulz na centrovanie. Skracovaním a predlžovaním impulzu je možné pohybovať ramenom do oboch strán v rámci operačného rozsahu. Pre dosiahnutie hraničných polôh sa pulz môže zmeniť až o 1ms. Ako som už spomenul skôr tak nútením serva za tieto hraničné polohy môže spôsobiť jeho vyhorenie či zničenie jeho prevodov, či už s dôvodov mechanickej zarážky alebo prekročenia limitov elektroniky.



Obr. 2.3: Ovládací pulz serv[3]

Toto platí pre klasické servá a s určitou abstrakciou aj na lineárne servá kde to nie je pohyb po kružnici ale pohyb po priamke. Výnimkou sú kontinuálne serva u ktorých sa význam riadiacich impulzov trochu líši. Pretože nenastavujú presnú polohu ale sú schopné sa neustále točiť dokola tak neutrálna poloha zodpovedá zastavenému motoru. Pri menení pulzu môžeme servo roztočiť na jednu, či na druhú stranu podľa toho ku ktorému hraničnému bodu sa blížime a zároveň čim sme tomuto bodu bližšie tým má servo vyššiu rýchlosť otáčania daným smerom.

[3]source:`http://wearcam.org/ece385/lecturelab6/servo_pwm_pulses.gif`

3 Prehľad existujúcich počítačových riešení riadenia serv

Táto kapitola predstavuje zjednodušení popis platforiem a existujúcich riešení na daných platformách.

3.1 Platforma FITKIT

Jedná sa o učebnú pomôcku založenú na Spartan 3,používanú pre HW predmety na FIT VUT Brno. FITkit je samostatný hardvér, ktorý obsahuje výkonný mikrokontróler s nízkym príkonom, hradlové pole FPGA (field programmable gate array), a mnoho periférií. Generovanie programovacích reťazcov pre FPGA v jazyku VHLD prebieha úplne automaticky s pomocou profesionálnych návrhových systémov. Softvér pre mikrokontróler je vytváraní v jazyku C a do spustiteľnej formy sa prekladá pomocou GNU (GCC), ktorý je možné používať zadarmo. Všetky potrebné softvéry pre návrh sú tiež zdarma.

Obr. 3.1: Platforma FITKIT[1]

Základné informácie o kite z oficiálnych stránok:

- MCU rodiny MSP430 (Texas Instruments)
- FPGA Spartan 3 XC3S50-4PQ208C nebo XC3S400-4PQ208C (Xilinx)
- USB převodník FT2232C
- audio rozhraní
- konektory PS2
- rozhraní VGA
- konektor RS232

[1]source:`https://merlin.fit.vutbr.cz/FITkit/imgs/uvod/001.png`

- DRAM 8x8Mbit
- Klávesnice
- Řádkový LCD displej
- Rozšiřující konektory

Podnetom vzniku tejto platformy bola nutnosť zabezpečiť hardvérovú platformu ktorá je ľahko dostupná a samostatná aby študenti nemali nutnosť vlastniť iné hardvérové prípravky, napríklad rôzne programátori.

Sprovoznenie

Doporučujem si stiahnuť jeden z virtuálnych diskov poskytnutých fakultou ja som použil Windows 7 približne 20GB, tento rok už sú k dispozícii len Windows XP a Windows 10. Ako prvé si do virtualboxu doinštalujte guest additions v novších verziách stačí pod zariadeniami , optickými zariadeniamy pridať dané ISO pre staršie verze je treba ISO stiahnuť. Následne môžete skúsiť kit pripojiť a spustiť Qdevkit, v ktorom by sa mal zobraziť ako dostupný ak nie tak skontrolujte či je USB kitu bežne FTDI Dual RS232 v hornej ponuke virtualboxu, zariadenia potom USB prepnuté na hosťovský pc alebo virtuálny stroj.

Obr. 3.2: Prepnutie FITKITu mezi hosťovským a virtuálnim systémom[2]

Ak tam nieje skontrolujte či ste nedostali kit s prehadzaními jumprami správne rozloženie nájdete na stránkach fitkitu. Občas je treba preinstalovať ovladač FTDI FT2232 pre kit. Správnu funkčnosť overíte nahraním jedného z demoprogramov a jeho nasledovným spustením.

[2]source:`https://merlin.fit.vutbr.cz/FITkit/`

FITKIT I/O

Fitkit bol navrhnutý s základnými perifériami na rok jeho vydania to jest 2008 pre verziu 2. FITKIT obsahuje porty PS2, USB, VGA, USB B, čo bolo na daný rok dostatočné. USB je použité na napájanie FITKITU a komunikáciu s počítačom s prepustnostou 1MS/s, ktorú potvrdzuje LED 8. Ďalej sem spadá konfigurovateľní dvojkanálový prevodník UART/FIFO, RS232 alebo RS422/RS485 a podporu virtuálneho COM.

LCD displej: Podporuje 16 znakov na riadov, verzia 2 má dvojriadkový displej. Displej obsahuje pamäte ROM na preddefinované ASCI symboly, RAM pre uchovanie 128 Bajtov s podporov definície vlastných znakov. Klavesnica: 4x4 maticová klávesnica VGA:Klasickí analógový rozhranie pre pripojenie obrazovky. R,G,B kanály obsahujú 3 bitový DAC pre každý kanál. Podporované rozlíšenie je 604x480 bodov s vykreslovacou frekvenciou 25MHz a snímkovou frekvenciou 60Hz.

FPGA

„Programovateľné logické obvody je skupina viacerých druhov digitálnych integrovaných obvodov, ktorých funkcia je určená užívateľom prostredníctvom predpisu (programu) definujúcemu prepoje jednotlivých blokov vo vnútri obvodu. Nemyslia sa tým však obvody postavené na mikroprocesoroch(mikrokontroléry).“[31] Jedná sa o programovateľné hardvérové pole, čo je hardvérová súčiastka ktorej štruktúru je možné definovať pomocou konfiguračného reťazca. Spadá do rodiny logických integrovaných obvodov. Štruktúru reprezentuje matica programovateľných blokov(CLB). Vďaka týmto bokom je možné tento hardvér meniť vďaka jazykom na popis hardvéru a netreba fyzicky prepájač, či prepájkovávať súčiastky. Tiež je možné za ich pomoci vytvoriť skoro akýkoľvek číslicový obvod, či už rôzne procesory alebo riadiace obvody. Táto vlastnosť je odlišnosťou od zákaznických integrovaných obvodov, ktorých funkcia je definovaná už pri ich konštrukcii. FPGA obvody majú široký rozsah uplatnení a to vo vývoji zariadení, málo kusových sériách, prototypoch.

Ako už bolo spomenuté vyššie pre syntézu sa využívajú takzvané HDL po slovensky jazyky popisujúce hardvér. K známym HDL patria AHDL, ktorý už je starší a zameranejší na analógové obvody a potom modernejšie VHLD. VHDL podporuje návrh i simulovanie digitálnych integrovaných obvodov. Ďalšou vlastnosťou je, že podporuje sériové aj paralelné popisy hardvéru. A nakoniec Verilog. Podobný jazyku C poskytujúci návrhy analógových, digitálnych aj zmiešaných obvodov s rôznou úrovňou

MCU

Mikrokontróler je jednočipový microprocesor prispôsobený pre špecifické koncové aplikácie. Bežne čip obsahuje procesorové jadro, pameti, časovače a programovadelné vstupno-výstupné periférie

V dnešnej dobe je už veľmi podobný ale stále menej zložitý ako SoC. SoC sú jednočipové systémy ktoré nájdeme napríklad v telefónoch alebo v Raspberry Pi. Jednoduchým rozdielom medzi nimi je že Soc väčšinou obsahuje, môže ale nemusí, mikrokontróler. Mikrokontólery môžeme nájsť všade okolo nás v takzvaných automaticky kontrolovaných produktoch a zariadeniach, ako sú riadiace systémy v automobiloch, v lekárskych implantátoch, kávovaroch, diaľkových ovládaniach, hračkách a v ostatných vstavaných systémoch. Mikrokontrólery pre zmiešané signáli sú veľmi rozšírené, lebo sú schopné pracovať s analógovými aj digitálnymi prvkami v ich okolí. A vďaka rozširovaní podpory internetu vecí sa ich použite stále zvyšuje. Okrem nízkej ceny je ich silná stránka aj malá spotreba a schopnosť spať, čiže bežia len vnútorné hodiny, počas ktorej ich spotreba klesá na rády miliwatov až mikrowatov.

Vstavaný systém je najrozšírenejšou formou rozšírenia mikrokontrólerov. I keď vstavané systémy môžu byť veľmi sofistikované, tak väčšina má minimálne požiadavky na pamäť, bez operačného systému a malou zložitosťou softvéru. Typicky mávajú na vstupy a výstupy pripojené relé, led diódy, malý LCD displej, rádio súčasti, rôzne senzory na teplotu, vlhkosť, akceleráciu. Ďalej v týchto vstavaných systémoch zodpovedá za odozvu na podnety. Na základe udalosti môže vzniknúť prerušenie pre ktoré je nutné začať obslužnú rutinu a pozastaviť aktuálnu akciu, po skončení môžu vrátiť obsluhu pôvodne obsluhovanému objektu. Prerušenia môžu signalizovať veci od potvrdení úspešnosti operácie cez rôzne chybové hlásenia až po zobúdzanie zo spánku a spätnému ukladaniu a čakaniu na ďalšiu udalosť na periférii.

Mikrokontróler na fitkite je z rodiny MSP430 Texas Instruments. Jedná sa o 16-bitový nízkonapeťový mikroprocesor s 92kB FLASH pamäti a 8kB RAM pamäti.

3.2 Platforma Raspberry Pi

Je veľmi populárny minipočítač. Ide o jednodoskový počítač o približnej veľkosti porovnateľnej ku kreditnej karte. Rozmery udávané výrobcom 85.60mm x 56mm x 21mm. Za tento malý pc vďačíme: Raspberry Pi Foundantion.

Táto malá krabička v sebe skrýva GPU s podporov OpenGL ES 2,0 hardvérovej akcelerácie a 1080p30 H.264 enkodér a dekodér. GPU tiež zvláda 1Gpixel/s 1,5Gpi-

[3]source:`https://cf3.s3.souqcdn.com/item/2016/04/07/10/52/20/94/item_XL_10522094_13680816.jpg`

Obr. 3.3: Platforma Raspberry Pi 3 model B[3]

xel/s alebo 24 GFLOPy pre všeobecné výpočty a viacero textúrových filtrov a DMA infraštruktúr. Toto je zhruba odpovedajúce výkonu pôvodného XBOXsu. Čo sa týka procesoru tak sa Raspberry Pi 3 model b má procesor s štyrmi jadrami o frekvencii 1,2GHz

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1 || 2			5v		
2	8	SDA.1	IN	1	3 || 4			5v		
3	9	SCL.1	IN	1	5 || 6			0v		
4	7	GPIO. 7	OUT	0	7 || 8	0	IN	TxD	15	14
		0v			9 || 10	1	IN	RxD	16	15
17	0	GPIO. 0	OUT	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	OUT	0	13 || 14			0v		
22	3	GPIO. 3	OUT	0	15 || 16	0	IN	GPIO. 4	4	23
		3.3v			17 || 18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19 || 20			0v		
9	13	MISO	IN	0	21 || 22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23 || 24	1	IN	CE0	10	8
		0v			25 || 26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27 || 28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29 || 30			0v		
6	22	GPIO.22	IN	0	31 || 32	0	IN	GPIO.26	26	12
13	23	GPIO.23	ALT0	0	33 || 34			0v		
19	24	GPIO.24	IN	0	35 || 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 || 38	0	IN	GPIO.28	28	20
		0v			39 || 40	0	IN	GPIO.29	29	21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM

Obr. 3.4: GPIO pini Raspberry Pi

Okrem jeho použiteľnosti ako domáceho servera, či média centra má ďalšiu silnú stránku a tou je jeho rozšírené vstupno-výstupné rozhranie, presnejšie nás bude zaujímať 24 pinov pre všeobecné vstupno-výstupné použitie ďalej len GPIO. Gpio podporuje 3,3V a 5V napájanie s maximálnym prúdom podľa použitého zdroja a možnosťou nastavovania logických úrovní a programovateľného chovania. Robia spolu s platformou Arduino dobrú voľbu pre výuku, či tvorbu nízkonapäťových obvodov.

Knižnica RPi.GPIO

V samotnom popise knižnice ktorý hovorí:„This package provides a class to control the GPIO on a Raspberry Pi .Note that this module is unsuitable for real-time or timing critical applications.. . . “[6] , sa môžeme dozvedať, že knižnica poskytuje triedu na ovládanie GPIO a že je táto metóda nevhodná pre časovanie pre kritické aplikácie. Ďalej popisuje nedostatky ako sú nepodporované SPI,I2C, hardvérové PWM, sériová funkčnosť. Sám autor knižnice doporučuje pre real-time časovače platformu Arduino. Hlavným dôvodom pre nevhodnosť riešenia na časovanie autor uvádza „garbage collector“ v jazyku Python a jeho nepredpovedateľné spúšťanie.

Knižnica RPIO.GPIO

Knižnica RPIO je pokročilý modul pre Raspberry Pi. Poskytuje PWM s pomocou DMA s použiteľným minimom zmeny signálu o 1µs. Ďalej predstavuje náhradu za RPI.GPIO pre ovládanie vstupov a výstupov. Tiež podporuje GPIO prerušenia a TCP sokety. Knižnica sa skladá z dvoch komponent, a to modulu do Python-u 2 a 3 a nástroju do terminálu pre manipuláciu s GPIO. Knižnica obsahuje triedu so základnými metódami prednastavenými pre ovládanie serv pracujúcich s 20 ms periódou. Okrem tejto triedy modul obsahuje aj iné triedy pomáhajúce s generáciu pulzov a poskytuje aj veľa príkladov s rozsiahlou dokumentáciou.

GPIO utilita

Tiež spojená s menom WiringPi. Táto utilita vytvára rozhranie pre komunikáciu GPIO pinov a tým pomáha lepšie a hlavne ľahšie komunikovať s GPIO Raspberry Pi. Táto utilita je pre shell-scripting v Bash-y. Príklad namiesto
„echo "out» /sys/class/gpio/gpio18/direction„ sa použije „gpio mode 18 out“. Manuál je samozrejme dostupný pod „man gpio“. Tiež poskytuje podporu a k rozširujúcim doskám pripájaním cez GPIO pini. Okrem PWM program podporuje aj kernel moduly ako sú spi a i2c. S týmto prístupom máme garantované hardvérovo časované pulzy lebo využívame primo jeden z PWM kanálov. Tomuto odpovedá aj počet pinov pre PWM a to 2, pin 23 a 26.

Projekt ServoBlaster

Softvérový modul na ktorom sú založené dve skúmané riešenia, konkrétne pigpio a pi–blaster. Modul poskytuje rozhranie pre ovládanie viacerých serv z GPIO pinov Raspberry Pi. Ovládanie je založené na posielaní príkazov ovládaču, ktorý na základe požiadavky vyrobí požadovanú dĺžku pulzu a udržuje ju. Modul je prednastavený a testovaný pre osem serv. Aj keď autor uvádza možnú prekonfiguráciu na až 21 serv. Ovládač si vytvára vlastný súbor zariadenia do ktorého sa zapijú príkazy. Formát príkazov je <servo-number>=<servo-position> alebo P<header>-<pin>=<servo-position> . Prednastavené hodnoty a označenia sa nachádzajú v tabuľke nižšie.

Servo number	Pin number	Pin in P1 header
0	4	P1-7
1	17	P1-11
2	18	P1-12
3	21/27	P1-13
4	22	P1-15
5	23	P1-16
6	24	P1-18
7	25	P1-22

Tab. 3.1: ServoBlaster: Predkonfigurované mapovanie pinov

Mapovanie pinov na číslo serva je popísané v súbore servoblaster—cfg ktorý je tiež vytvorený v priečinku zariadení. Toto mapovanie je možné zmeniť zadaním správneho príkazu. Všetky pipni ktoré bude možné používať a niesú v prednastavených musia byť zadané pred spustením samotného ovládača lebo ovládač si na základe počtu serv vytvorí ich rozloženie štartovných časov v perióde. Modul je schopný vytvárať PWM signál s pomocou DMA a pulzy pre DMA časuje buď z PWM kanálu alebo PCM periférie.

Knižnica piGPIO

Knižnica pigpio je phyton a C knižnica zahrnutá v systéme raspbian a iné distribúcie ju tiež môžu jednoducho stiahnuť. Táto knižnica podporuje hardvérové PWM, predpokladom je že využíva ten istý čip ako predošlé riešenie. Rozdielom oproti predošlému riešeniu, ktoré dovoľuje PWM len na dvoch pinoch je PWM na všetkých GPIO pinoch, čiže je možné ovládať 17 serv. Dosahuje toho využitím démona pigpiod, napísaného v jazyku C, bez ktorého zobrazí chybovú hrášku a žiada o jeho zapnutie. Tento démon zabezpečuje obsadenie hardvéru a komunikáciu s ním. Táto

knižnica má viac metód generácie samotného signálu, pre účely ovládania serv a jednoduchosť sa dá využiť funkcie „set_servo_pulsewidth“, ktorá má výstižné meno a nastavuje len dĺžku pulzu. Obmedzenia tejto funkcie sú , že posiela pulzy s rozsahom šírky 0,5 až 2,5 ms čo je rozsah pre servá s pohybom 180°.

Medzi hlavné funkcie knižnice patria: hardvérové vzorkovanie a časové značenie od 5µs nahor, hardvérové PWM, hardvérové pulzy pre servá, oznámenia cez komunikačnú rúru callbacky, zhvukový zápis ako jedinú operáciu, generáciu vĺn a správu sériovej linky. Ďalej s pripojení nadstavby piscope je možné vytvoriť logický analyzátor vĺn v reálnom čase zobrazujúci priebehy na GPIO pinoch.

Projekt piblaster

Medzi jedno z prvých riešený používajúce hardvérové časovanie ktoré som našiel patrí tento komunitný projekt, za ktorý vďačíme pánom: Thomas Sarlandie, Richard Hirst, Michael Vitousek, Pete Nelson, Edgar Siva, Alex Lennon, Lara Maia, Pattrick Hüper, Boyuan Yang, penfold42 , Thomas McVay, Pavle Petrovic. Popis od tvorcov je nasledovný: „This project enables PWM on the GPIO pins you request of a Raspberry Pi . The technique used is extremely efficient: does not use the CPU and gives very stable pulses. Pi-blaster project is based on the excellent work of Richard Hirst for ServoBlaster. Pi-blaster has also been forked and made to work with MQTT instead of a local device file. That project is available here.[4]“[9], z čoho vyplýva , že tento projekt dovoľuje PWM na GPIO pine, ktorý si vyžiadame od Raspberry Pi a použitá technika je efektívna lebo nepoužíva procesor a generuje stabilné pulzy. Potom nasledujú kredity a odkaz na projekt z rovnakého základu ale využívajúci protokol MQTT z čoho usudzujem že sa dá použiť tiež lokálne ale je predpripravený na preposielanie správ iným zariadeniam na správu.

Tento projekt využíva vlastného démona, ktorého nastavenie sú popísané v sekcii s riešením. Na časovanie pulzov používa techniku priameho prístupu do pamäte. Pred spustením je treba nastaviť všetky pini, ktoré chceme používať ako výstupy. Po spustení sa vytvorí rúra s FIFO radou pre zapisovanie požiadavkou. Formát požiadavkou je „echo "BCM=hodnota» /dev/pi-blaster“ kde hodnota je medzi 0 až 1. Táto hodnota predstavuje percentuálny podiel doby signálu v logickej 1.

Prvé zapnutie

Pre používanie tohto projektu je treba spraviť viac krokov ako stiahnuť a spustiť. Pre použitie pre modelárske serva je treba zmeniť základnú konfiguráciu.Prvé je treba si prepísať známe pini. pretože ich číslovanie sa zmenilo medzi jednotlivými verziami.

[4] `https://github.com/gherlein/pi-blaster-mqtt`

Pre túto zmenu najprv si nechať vypísať gpio označenie príkazom „gpio readall“ a zaujímajú nás riadky obsahujúce GPIO. a stĺpec BCM. Nasledovne si otvoríme „pi-blaste.c“ a prepíšeme si BCM čísla do štruktúry „known_pins[]“. Nasledovne treba zmeniť „CYCLE_TIME_US“ na 20000 a „SAMPLE_US“ na 20. A na koniec skompilovať a spustiť s elevációu.

Raspberry Pi I/O

Raspberry Pi 3 model B, ktorý som použil ma 4 USB A porty, fastethernet, 802.11ac/n wifi, HDMI v1.3a, mikro USB pre napájanie, 2.5 audio jack, fast ethernet. Iné modely síce uvádzajú že podporujú Gigabit ethernet ale je to prepojené s USB takže rýchlosť bude končiť okolo 400Mb/s. Samozrejme nesmiem zabudnúť na 40 pinov. Tieto pipi sa nachádzajú na okraji dosky. GPIO sa dá softvérovo prenastaviť ako vstup či výstup s širokou možnosťou uplatnenia. Z týchto pinov je 17 pre všeobecné použitie, 6x GND, 2*5V napájacie, 2*3,3V napájanie a ostatné pini sa dajú použiť ako GPIO pipi ale sú schopné aj alternatívnych funkcií alebo sú navrhnuté na pripojenie periférií, či prednastavené pre použitie určitých protokolov. Vstupy aj výstupy sú nastavené na spracovanie logických signálov. Výstupy sú navrhnuté pre logickú 1 na 3V3 a logickú 0 na 0V. Vstupy sú na tom podobne a tiež čítajú v rozmedzí od 0V do 3V3. Ďalej majú podporu nastavenia pull-up a pull-dowm registru. Až na výnimku dvoch gpio pinov je možné tento register nastaviť softvérovo. Ako bolo spomenuté vyžlie tak niektoré pini poskytujú alternatívne nastavenia ako sú PWM, SPI, I2C, Serial.

3.3 Platforma Personálny počítač

Osobný počítač tiež nazývaný komputer, computer anglicky presonal computer, so skratkou PC je počítač ktorého veľkosť, možnosti a cena dovoľujú jeho osobné použitie, či použitie jednotlivcom. Osobný počítače sú určené na prevádzku priamo koncovým užívateľom, a nie technikom alebo počítačovým expertom. Na rozdiel od mainframov, minipočítačov určených pre obsluhu mnohých požiadavkou a zdieľanie zdrojov medzi mnohými používateľmi sú osobné počítače určené práve pre jedného používateľa. Práve z tohto vychádza názov osobný počítač, lebo v čase jeho vzniku boli počítače veľmi rozmerné a nie len pre jedinú osobu. Osobný počítač vznikol kombináciou rôznych štandardou, z ktorých najväčšiu kontribúciu štandardou majú IBM a Apple. V dnešnej dobe je oveľa väčšie percento osobných počítačov a preto keď dnes niekto použije pojem počítač, tak vlastne používa synonymum pre osobný počítač alebo aj počítačovú zostavu s jej vstupnými aj výstupnými zariadeniami.

Ďalej sa tento termín rozšíril na konci sedemdesiatych a začiatkom osemdesiatych rokov minulého storočia, práve keď sa menil pomer počítačov v laboratóriách a domácnostiach. Spolu s prechodom počítačov do domácnosti prišla aj zmena významu z vedeckého ponímania na zábavné zriadenie. Dnešné osobné počítače sú ponímané a využívané ako zábavné elektrospotrebiče pripojené na internet a so schopnosťou prehrávania, modifikácie, ukladania rôzneho multimediálneho obsahu, tiež môže slúžiť ako prepojenie medzi ostatnými zariadeniami.

Osobné počítače v dnešnej dobe je postavený najmä na báze procesorov architektúry x86 objavujú sa aj procesory ARM ale len v menšom počte. Ďalej obsahujú výkonnú grafickú kartou, ukladacie médium s úložiskom v stovkách GB až jednotkách TB na starších mechanických diskoch alebo novších netočiacich sa diskoch SSD, operačnú pamäť v jednotkách až desiatkach GB, pripojenie do internetu a lokálnej počítačovej siete. Počítače tiež obsahujú porty na pripojenie periférií ako sú LCD monitory, tlačiarne, vstupné a výstupné zariadenia i iné zariadenia schopné komunikovať s počítačom ako mobilné telefóny či hudobné prehrávače.

V minulosti si užívatelia museli písať vlastné programy a aplikácie pre využitie hardvéru v počítači no dnes je programové vybavenie počítačov uspôsobené tak aby ich mohol ovládať i neznalý užívateľ. Toto zabezpečuje operačný systém počítača. Najrozšírenejším je systém spoločnosti Microsoft pod menom Windows, s najpopulárnejšie verzie patria 7 a 10. Ako alternatívne možnosti pre operační systém je macOS od spoločnosti Apple alebo jeden z Open Source systémov s Linux-ovým jadrom.

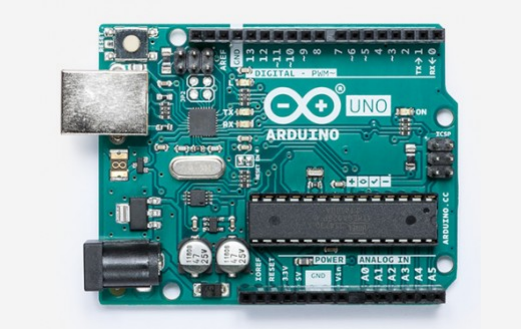
I/O personálneho počítača

Väčšina personálnych počítačov obsahuje rozhranie USB typu A cez ktoré je možné naviazať sériovú komunikáciu za pomoci virtuálneho COM portu. Veľmi staré modely mávali fyzický sériový port. Bohužiaľ dnešné personálne počítače nemajú podporu pre hardvérovo časované pulzy je síce možné vyrobiť softvérovo časovaný pulz ale tieto pulzy nebývajú tak presné a sú ľahko ovplyvniteľné prerušeniami či zaťaženým procesoru.[5]. Medzi ďalšie I/O ktoré sa dá nájsť je výstup pre zvuk, či už 3,5mm kombo(štvor pražcový) alebo klasický(dva s tromi pražcami) jack, fast/gigabit ethernet, výstup je obraz analógový(VGA, ...) alebo digitálny(hdmi, display port, ich mini varianty, DVI len na starých modeloch, ...), pre moderné prenosné počítače sa pre úsporu miesta používa USB typu C, ktoré zvláda niesť viacero tzpov signálov zároveň(napájanie, obraz, usb, ...). I/O personálnych počítačov nie je uspôsobené pre pripájanie nízkonapäťových obvodov a je treba využiť rôznych

[5]softvérové pulzy predvedené v sekcii Raspberry

programátorov, mikrokontrólerov alebo iných špecializovaných prípravkov prenášajúcich požadovanú komunikáciu cez a na USB zbernicu.

3.4 Platforma Arduino



Obr. 3.5: Platforma Arduino[6]

Arduino je open-source hardvér a softvér spoločnosť, projekt a užívateľská komunita, ktorá navrhuje a vyrába jednodoskové mikrokontrólery a prípravky s mikrokontrólermy pre rôzne skladby, či interaktívne objekty schopné vnímať a kontrolovať analógovo aj digitálne. Tieto produkty sú licencované pod GNU Lasser General Public License alebo pod GNU General Public License, ktoré dovoľujú ako výrobu Arduino dosiek tak softvérovú distribúciu kýmkoľvek. Arduino je na trhu dostupné ako predpripravený zložený výrobok alebo ako skladačka pre nadšencov.

Návrhy Arduino dosiek používa rôzne mikroprocesory a kontroléry. Dosky sú vybavené súbormi digitálnych a analógových vstupno-výstupných pinov cez ktoré je možné pripájať rôzne rozširujúce dosky, nepájavé polia a iné obvody. Dosky sú schopné sériových komunikácií, vrátane USB konektoru na určitých doskách. Tieto zbernice sú používané na nahrávanie programov z osobných počítačov. Mikrokontrólery sú typicky programované za využitia dialektu z programovacích jazykov C a C++. Navyše oproti využitiu tradičných kompilačných nástrojov projekty Arduina poskytujú integrované vývojárske prostredie na základe použitého jazyka projektu.

Projekt Arduino začal v roku 2003 ako program pre študentov na Interaction Design Institute Ivrea v Ivree, Taliansko. Cieľom bolo poskytnutie nízko-nákladnej a jednoduchej cesty pre začiatočníkov aj profesionálov k vytváraniu zariadení schopných interakcie s prostredím s využitím rôznych senzorov a motorov. Bežným príkladom zariadenia pre začiatočníka môže byť termometer, senzor pohybu alebo jednoduchí robotík. Meno Arduino je prebrané z mana baru v Taliansku, kde sa zakla-

[6]source:`https://store-cdn.arduino.cc/uni/catalog/product/cache/1/image/520x330/604a3538c15e081937dbfbd20aa60aad/a/0/a000066_featured_1_.jpg`

datelia projektu stretávali. Bar bol pomenovaný podľa Arduin of Ivrea, ktorý bol vojenským veliteľom z provincie Ivrea a kráľom Talianska v rokoch 1002 až 1014.

Arduino I/O

Väčšina dosiek Arduino má jednoduché I/O, ktoré bežne obsahuje sadu analógových a sadu digitálnych pinov vždy na opačných okrajoch dosky. Konektor na sériovú komunikáciu, napájací konektor a napájacie piný. Dnešné modely ponúkajú 14 pinov pre všeobecné vstupno-výstupné použitie a z toho 6 pinov s podporov PWM s skupinách po dvoch. Vždy dva pipi na jeden časovač. Ďalej má doska 6 analógových pinov, ktoré môžu zastať úlohy digitálnych pinov.

Analýza riešení

Nasledujúce riešenia budem ďalej skúmať a testovať, vynechávam servoblaster z dôvodu že pôsobí ako základ a je integrovaný v piGPIO a pi-blaster. Potom vypúšťam Arduino lebo je platforma dobre zmapovaná a existuje veľké množsto návodov.

Rpi.GPIO

Prvé riešenie pre Raspberry Pi a zároveň aj referenciu s ukážkou prečo nechceme PWM riadené a časované softvérovo tak som zvolil práve PWM časované softvérovo. Tu je predpoklad viditeľných záchvenou na servách. Tento riešenie je schopné softvérového PWM na všetkých GPIO pinoch(17+).

GPIO

Druhé skúmané riešenie je založené utilite zahrnutej v oficiálnom operačnom systému zvanej gpio. Požadované pulzy získame zo základnej frekvencie 19.2MHz napríklad kombináciou deličiek pwmc=1920 a pwmr=200. Iná kombinácia ak je treba viac menších krokov ja napríklad pwmc=8000 pwmr=48.

$$50Hz = 19200000/pwmc/pwmr$$

Druhá ďelička nám zároveň udáva rozsah vzoriek ktoré môžem využívať. Pre mňa to znamená pri tejto konfigurácii posúvanie sa po $2,5\mu s$.

$$20ms/pwmr(8000) = 2,5\mu s\ krok,$$

s týmto nastavením je možné hýbať servom po $\frac{1}{4}$ stupňa. Vycentrované servo dostaneme pri pulze o šírke 1,5ms zodpovedajúcej poslaním hodnoty 600 na daný pin. Veľkým obmedzením tohto riešenia je že mnou použitý model má len spomínané dva kanáli pre PWM čiže len 2 pini s priamou podporou, inak povedané len dva pini podporujúce mód PWM. Toto obmedzenie som obišiel pripojením demultiplexora. Pre daní demultiplexor je ale nutné obetovať pár pinov pre jeho ovládanie konkrétne tri pre selektor 1 pre aktiváciu a 1 pre PWM signál. Ďalej je treba si uvedomiť, že pri prepínaní medzi servami nemôžeme vypnúť daný demultiplexor. Takýmto prepojením je možné ovládať 8 serv z jedného pinu, čím sa dostanem k celkovému počtu 16 možných serv.

piGPIO

Riešenie časuje pulzy za pomoci DMA s minimálnim krokom 2µs a používa PWM alebo PCM na časovanie pre DMA, predvolene PCM lebo prvý PWM kanál je používaní pre audio. Táto knižnica má viac metód generácie samotného signálu, pre účely ovládania serv a jednoduchosť sa dá využiť funkcie „set_servo_pulsewidth“, ktorá berie dva parametre BCM číslo pinu a požadovanú dĺžku pulzu z rozsahu 1000-2000, zodpovedajúcej dĺžke 1-2ms s periódou 20ms. Tento riešenie je schopné hardvérového PWM na všetkých GPIO pinoch(17+).

Pi-blaster

Riešenie časuje pulzy za pomoci DMA s minimálnym krokom 2µs a používa PWM alebo PCM[7] na časovanie pre DMA, predvolene teda PWM. Takže tu je predpoklad kolízie s analógovým zvukom lebo oba môžu použiť prvý PWM kanál zároveň. Tento projekt má odlišný prístup od pigpio a nedovoľuje frekvenciu po spustení démona meniť. Po spustení je najjednoduchšie zavolať z terminálu:
„echo "BCM=hodnota» /dev/pi-blaster“
kde BCM je BCM číslo pinu a hodnota zodpovedá pracovnej dobe z rozsahu 0 až 1, tu 0.5 zodpovedá 50%. Tento riešenie je schopné hardvérového PWM na všetkých GPIO pinoch(17+).

FITKIT FPGA

Základná frekvencia čipu na FITKITe je 7.3728MHz. Riešenie pomocou FPGA má veľmi jednoduchý základ pre generáciu požadovaného signálu. Princíp spočíva z čítača, ktorý beží voľne a sa inkrementuje s každou nábežnou hranou hodinového signálu a nuluje sa pri dosiahnutí požadovanej periódy. Registra v ktorom je uložená šírka nášho požadovaného signálu. A poslodnou súčasťou je komparátor ktorý má na vstupoch čítač a register a jeho výstupom je môj signál s požadovanou frekvenciu a šírkou. Tento riešenie je schopné PWM na všetkých pinoch(4+[8]).

[7] v experimentálnom štádiu

[8] Štyri boli testované, teoreticky až 40 na ktoré sa dá nadpojiť

FITKIT MCU

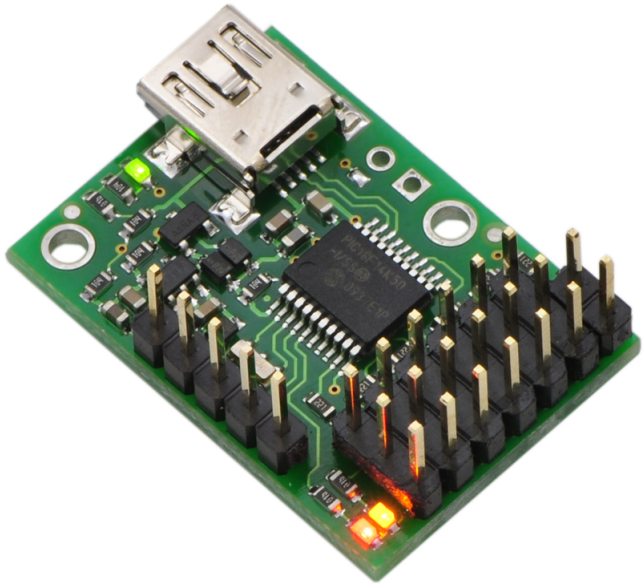
Riešenie využívajúce prerušení k časovaniu pulzov, je použiteľné len ak čip nemá inú väčšiu záťaž, čiže ak by tento čip ešte mal komunikovať cez zbernicu tak by to už pravdepodobne nestíhal a serva by sa chveli. Časovanie som založil na príklade zameranom na blikanie LED každej 0.5s, z ktorého som zistil princíp a základnú frekvenciu fitkitu. Kit má frekvenciu 32768Hz čo zodpovedá 0x8000 hexadecimálne. Z tejto informácie sa dá vypočítať počet tikov predstavujúcich požadovanú frekvencie 50Hz. Použitý prepočet je „frekvencia kitu/požadovaná frekvencia/počet vzoriek“ . Tento riešenie je schopné PWM na 4 pinoch.

Platforma Personálny počítač

Pre ovládanie serv z počítača je potrebná externá doska s mikrokontrólerom. Napr. Micro Maestro 6-Channel USB Servo Controller, ktorý poskytuje potrebné prepojenie aj časovače na ovládanie.

Pololu mikro maestro

Dosky s tejto rodiny boly navrhnuté pre ovládanie serv. Táto rodina má viac členov s rôznim počtom pripojiteľných serv.



Obr. 3.6: Pololu micro maestro[9]

[9]sourco:`https://a.pololu-files.com/picture/0J1951.1200.jpg?7e54ef728c9e5b6707a45ae5868bfb66`

Mikro maestro je možné pripojiť k počítaču pomocou mini USB. Možnosti ovládania sú celkom rozšírené konkrétne:

- grafické rozhranie
- skript
- ovládanie jazykmi s rozšírenou komunikáciu USB rozhrnia za pomoci balíčku pre vývojárov.

Grafické rozhranie je priamočiare a je prednastavené na rozsah predstavujúci bezpečný rozsah pohybu serva, čiže 50Hz o šírke 1ms až 2ms. Tento rozsah sa dá rozšíriť pre jednotlivé kanály kôli kalibrácii serv. Pre skriptovanie je potrebné aby nebol program v rozpoznávaní komunikácie a bola stanovená rýchlosť a ostatné parametre komunikácie. Použitý je špeciálny jazyk pre meastro dosky. Jedná sa o „maestro script language“, jazyk je štruktúrovo podobný assembleru.

Za použitia development kitu je možné písať vlastné aplikácie v napríklad C#, Visual Basic, .NET and Visual C++.

Parametre

Od dosky Pololu micro maestro 6 môžeme očekávať tieto parametre:

- **Samostatné kanáli**: 6
- **Podporované protokoly**: USB, UART
- **Kompatibilné napájanie**: 5–16V
- **Podporované frekvencie**: 33 – 100Hz
- **Minimálny krok**: 0,25µs

Nároky na demonštráciu skúmaných riešení

Pre demonštračné účely bude treba vytvoriť nejaký strojček. Rozhodol som sa pre robotické rameno. Po damon modele ramena budem potrebovať:

- Pohyb musí byť spôsobený modelárskymi servy.
- Ľahké vymieňanie riadiacej platformy.
- Model musí byť schopný zdvihnúť nejaký objekt.
- Model musí byť schopný udržať zdvihnutý objekt pri jeho premiesťnovaní.
- Finančná náročnosť
- Prenositeľnosť

Nakoniec z riešený vybratých po testoch je treba zabezpečiť riediaci program, či spôsob riadenia, ktorý dovolí demonštráciu presunu nejakého objektu.

Prehľad riešení k testovaniu

Tabuľka 3.2 zobrazuje prehľad riešení, ktoré budú podrobené testovaniu. Tiež zobrazuje základné vlastnosti týchto riešení.

Platforma	**Raspberry Pi**				**FITKIT**		**pc**
	rpi.gpio	**gpio**	**pigpio**	**pi-blaster**	**fpga**	**MCU**	**pc**
resource	CPU	PWM	PWM or PCM for DMA timing	PWM for DMA timing	FPGA	CPU interupts	external HW mikro maestro
kanály	17+	2	17+	17+	5+	4	6
base Langluage	py	bash	c	c	vhdl	c	its own
Lang. support	py	any that can use system() call	py or c	any that can write to file	hdl	C or py needs programer	c based, java, .NET ...
step	0.5ms[10]	<2 µs	2µs	2µs	<18µs	3µs	0.25µs

Tab. 3.2: Prehľad riešení k testovaniu

[10] in integer space, less in fload space

4 Zhodnotenie riadenia serv z rôznych platforiem

V tejto kapitole sa zaoberám zvolenými riešeniamy na platformách: Raspberry Pi, FITKIT, Personálnom počítači. Platformu Arduinu som netestoval z dôvodu, že na ňu existuje veľa návodov je dobre preskúmaná.

4.1 Riešenia platformy Raspberry Pi

Bola použitá verzia: Raspberry Pi 3 Model B s operačným systémom Raspbian, zdroj 5V,2.1A alebo power banka(16000mAh, 2A). O časovanie a generáciu sa stará čip BCM2837. Tento čip je napojený na kryštál s frekvenciou 19,2Mhz, tiež sa stará aj o iné periférie, aj o prerušenia a DMA. Schéma PWM kanálov čipu zobrazuje obrázok 4.1. Tento čip je schopný generovať PWM signál podľa dvoch algoritmov. Pre účely práce je vhodné použiť druhý algoritmus mód Mark-space, do ktorého sa dá prepnúť príkazom „pwm-ms“. Tento algoritmus posiela dáta s pomerom M/S, kde M sú dáta na poslanie a S je rozsah. Toto potom na výstupe spôsobí, že M vzorkou je hore z celkových S možných. Pre lepšiu predstavu majme M=2 a S=4, čo je presne 50% pracovná doba pulzu, t.j. polovicu periódy bude pulz v logickej jednotke a druhú v nule. Pre porovnanie tento algoritmus by zmenou M zmenil len pracovnú dobu a perióda by ostala konštantná, ale prvý algoritmus by zmenil aj samotnú periódu.

RPi pre generáciu a určenie frekvencie používa nasledovný vzorec „base frequency / pwmclock / pwmrange“. Kde base frequency zodpovedá 19,2Mhz a pwmclock s pwmrange sú deličky s ktorými môžeme manipulovať na dosiahnutie požadovanej frekvencie.

$$50Hz = 19200000/pwmc/pwmr$$

$$20ms/pwmr(8000) = 2,5\mu s krok.$$

Rozsah deličiek je pwmc

$$2 \ldots 4095$$

a pwmr

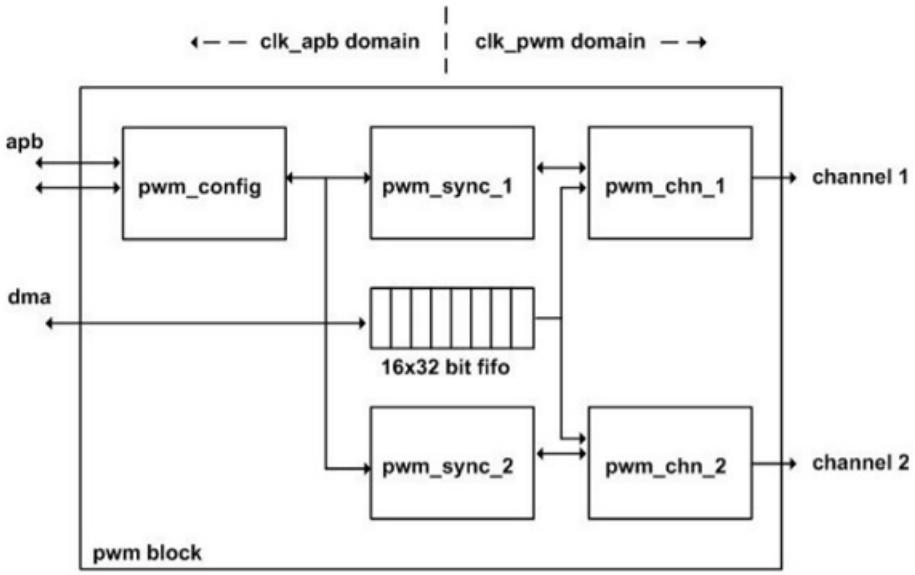
$$2 \ldots nešpecifikované$$

I keď je teda základná frekvencia 19,2 MHz tak minimálne nastavenia deličiek dovoľujú len štvrtinu, čo zodpovedá 4,8Mhz pri nastavení oboch deličiek na 2 a pracovným cyklom na 1, čiže 50%. Tieto nastavenia deličiek pre frekvenciu sú zdieľané pre obidva PWM kanály z čoho vyplýva, že zdieľajú frekvenciu a môžeme meniť dĺžku pulzu pre oba samostatne.

Výsledky testov platformy

Výsledky testov k samotným riešeniam nájdete v sekcii daného riešenia.

Ako základný test platformy som použil syntetický stress test. Pri testovaní stability pri dosiahnutých 4,8MHz stress testom na procesor boli zistené dve vlastnosti. Za prvé sa potvrdil kryštál ovládaný čipom BCM2837, ktorý pri zahriatí spôsoboval chvenie v desatinách mHz a nižšie. Druhou je napájací obvod cez mikro-USB. Nielen že cestičky na doske od USB k procesoru výraznejšie tenšie ale aj idú dlhšou cestou oproti cestičkami medzi procesorom a GPIO a týmto pádom cez USB sú nielen väčšie straty ale aj nižšie možné napájanie oproti napájaniu z GPIO. Z tohto dôvodu odoporučujem napájanie cez GPIO ak chcete používať Raspberry Pi dlhodobejšie s vyšším zaťažovaním procesoru.

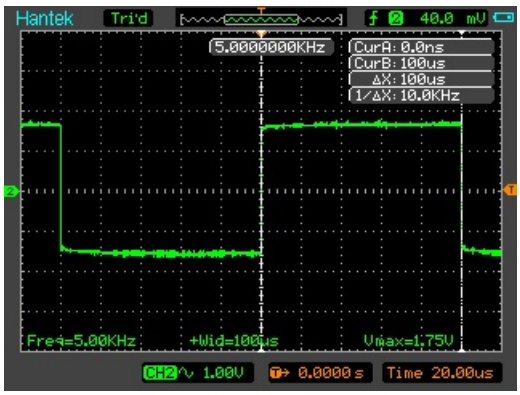


Obr. 4.1: Raspberry Pi schéma PWM kanálov[1]

Druhotné testy na osciloskope pod zaťaženým som sa rozhodol spraviť z dôvodu prehriatia a vypnutia systému za použitia syntetických testov. Moje testy mali za cieľ

[1]source:`https://cs140e.sergio.bz/docs/BCM2837-ARM-Peripherals.pdf`

zaťažiť procesor niekde v rozsahu 80-90% na každom jadre. Dosiahol som toho za pomoci VNC serveru pre vzdialenú plochu a prehrávaným HD(720p) alebo fullHD(1080p) videí obe 60fps bez požitia hardvérovej akcelerácie. Pri HD to robilo 90-98%. A zároveň som zaťažoval USB zbernicu prehrávaním týchto súborov z prenosného média.



Obr. 4.2: Raspberry Pi PWM kanál pod záťažou(druhá séria testov)

Druhotné testy som tiež pripojil na generovanú 5kHz frekvenciu kde som sledoval stabilitu signálu pod touto záťažou. Výsledkom bola nemerateľná zmena lebo ani frekvencia ani šírka sa nezmenila. Najmenšia zmena ktorú som bol schopný zmerať na kHz bola 0,1mHz, čo sa taká zmena nestala. I keď sa signál generuje pomocou kryštálu tak som žiadne zmeny na signály nenameral, čo predstavuje dostatočnú stabilitu pre modelárske serva.

Knižnica RPi.GPIO

Túto knižnicu používam ako základ prečo nechceme používať softvérovo časované pulzy.

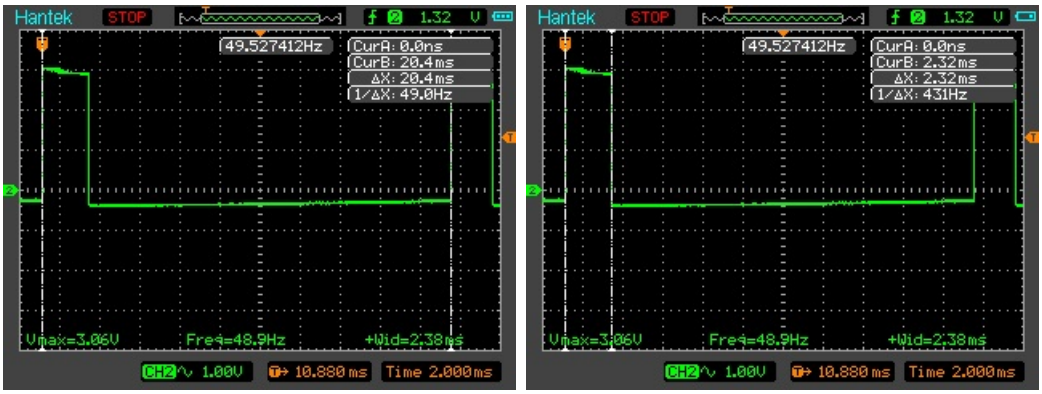
konfigurácia a implementácia k testom:

- **nastavenie frekvencie**: GPIO.PWM(servo_pin, 50)
- **zmena dĺžky**: p.ChangeDutyCycle(10), povolený rozsah pre serva 5 až 10
- **počet kanálov**: 1, jedná se len o referenčné hodnoty
- **dĺžka pulzu**: 2ms
- **Pripojených serv**: 1
- **testy**
  1. test naprázdno: CPU len s minimom programov
  2. test pri 20% záťaži na CPU: test bez merania len pozorovanie serva
  3. test po záťažou: druhotný test platformy, t.j. prehrávanie HD videa bez HW akcelerácie = CPU 90+%

Od knižnice neočakávam prívetivé výsledky lebo je časovaná softvérovo a môžu ju ovplyvniť systémové prerušenia či aj garbage collector integrovaný v Pythone.

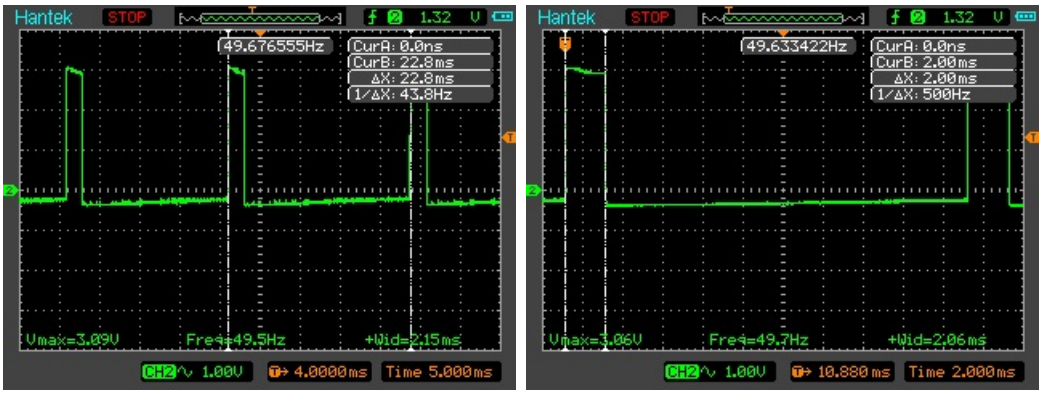
Výsledky testov

Výsledok zodpovedá predpokladu s tým že pri jedinom serve a s Raspberry Pi v úplne kľudovom stave to jest bežiaci len operačný systém sa dané servo natáčalo správne bez príliš očividného chvenia. Ale v okamžiku keď som sa pripojil pomocou ssh a dal si vypísať bežiace procesy čím vyskočilo zaťaženie iba jedného jadra na 20% sa servo začalo chvieť a meniť polohu i keď som nemenil nastavenia jeho polohy. Pri testoch na osciloskope bolo dokonca zjavné že aj keď som s ničím nehýbal bola generovaná frekvencia nestabilná a stále sa menila. Najvyššia zaznamenaná odchýlka na periódе bola až 2.8ms a najmenšia pri prvotných testoch na menej kvalitnom osciloskope bola 0.16ms ktorý nezaznamenal jej zmenu počas testov. Ďalším nedostatkom objaveným na kvalitnejšom osciloskope bola aj občasná zmena amplitúdy pohyby o 0.2V. Zábavným pozorovaním bolo keď sa jedno najväčšie servo samé od seba začalo hýbať vždy od 0° po asi 90° kde malo nastavenú polohu.

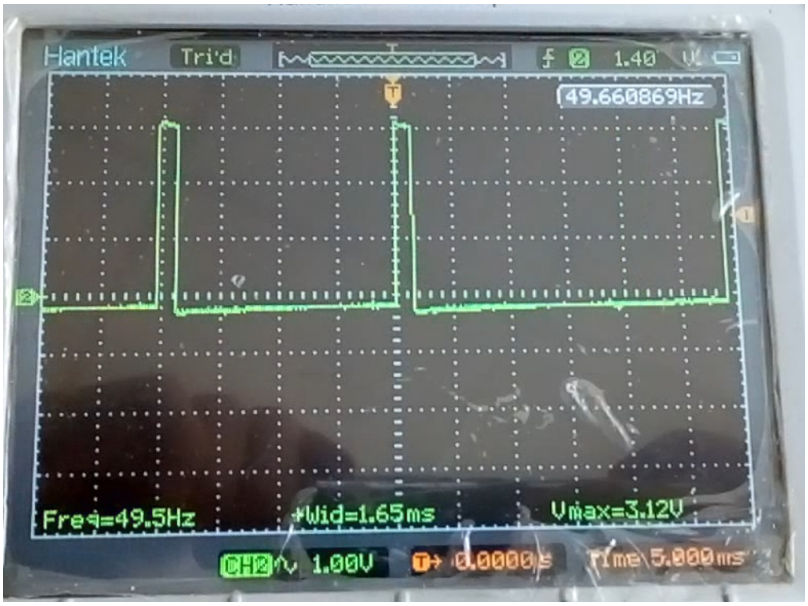


Obr. 4.3: RPi.GPIO bez záťaže na procesore

Druhotné testy som na toto riešenie aplikoval aj keď jeho použiteľnosť vyvrátili už prvotné testy. Výsledkom bol veľmi nestabilný signál, ktorého priebeh som zachytil pomocou mobilného telefónu na video s ktorého som vystrihol priebeh s veľkým odskokom periódy.



Obr. 4.4: RPi.GPIO pod záťažou



Obr. 4.5: RPi.GPIO pod záťažou, vystrihnuté z videa

RPi.GPIO	**na prázdno [ms]**	**pod záťažou [ms]**	**chyba [|ms|]**
perióda	20,4	22,8	0,16 - 2,8
šírka	2,32	2,2	0,06 - 0,32

Tab. 4.1: Výsledky merania: RPi.GPIO

Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrázkov. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.

Utilita GPIO

Tento program využíva priamo natívneho PWM takže očakávam rovnakú stabilitu ako pi testoch samotnej platformy.

konfigurácia a implementácia k testom:

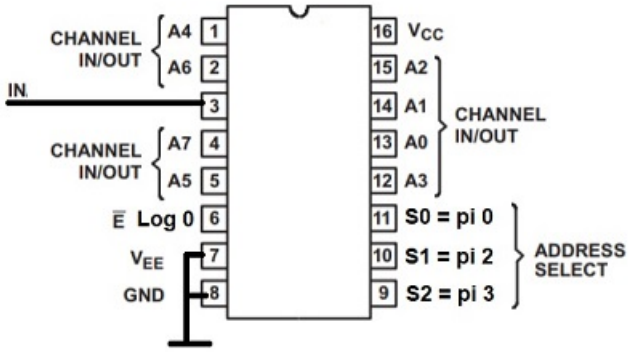
- **nastavenie frekvencie**: $50Hz = 19200000/pwmc/pwmr$
  - gpio mode pin PWM – prepnutie pinu 23 alebo 26 do funkcie PWM
  - gpio pwmc=48 – rozsah 2 až 4095
  - gpio pwmr=8000 – rozsah 2 až ∞[2]
  - gpio pwm-ms – použitie Mark-space algoritmu
- **zmena dĺžky**: gpio pin PWM 800, rozsah pre serva je 400 až 800
- **počet kanálov**: 2, len dva PWM kanály
- **Externý hardvér: demultiplexor(16 nôh)**
  - Jeden PWM kanál 8 serv
  - ovládanie – 3 pini nutné pre selekt signál
  - prepínanie mezi servamy nasledovne: gpio pin PWM 0 následne nový selekt
- **krok**: 2,5µs , iná konfigurácia dovoľuje aj menší krok
- **Pripojených serv**: 4
- **dĺžka pulzu**: 1,5 a 2ms
- **testy**
  1. test naprázdno: CPU len s minimom programov
  2. test po záťažou: druhotný test platformy, t.j. prehrávanie HD videa bez HW akcelerácie = CPU 90+%
  3. test so zvukom: pripojenie jacku a prehranie zvuku, lebo 1 PWM kanál sa používa pre analógový zvuk

Z dôvodu PWM len na dvoch pinoch používam externý hardvér. Konkrétne demultiplexor cd74hc4051 a skúmať, či spôsobí skreslenie.

Výsledky testov

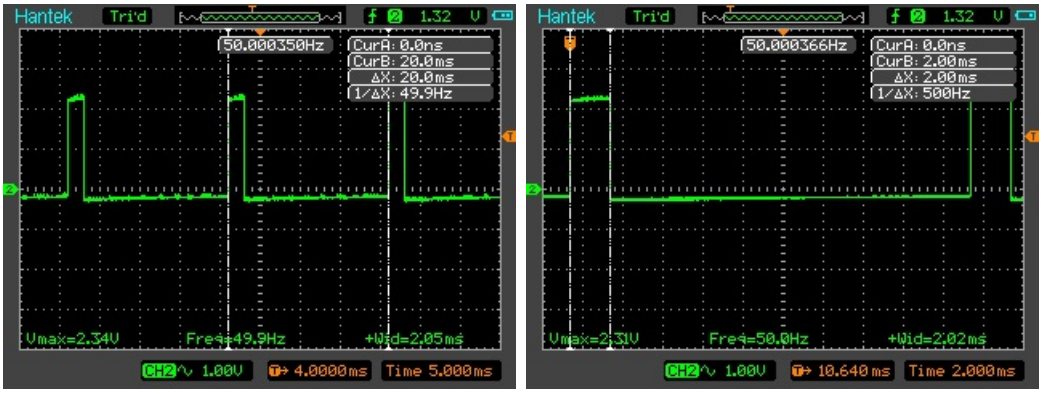
Čip demultiplexora priebeh neskresľuje oproti priamemu pripojeniu na pin Raspberry Pi. Pri vypnutí demultiplexora cez enable signál čip prepúšťa signál a skresľuje ho, preto je treba mať čip stále zapnutý. Pretože sú servá pripojené cez demultiplexor tak je signál posielaný vždy len na jediné servo, tým pádom ostatné servá majú väčšiu vôľu k pohybu pri záťaži. Výsledky zobrazené na osciloskope zodpovedajú predpokladu hardvérového PWM.

[2]Horná hranica nie je špecifikovaná testoval som 10^6



INPUT STATES				ON CHANNEL
ENABLE	S_2	S_1	S_0	
L	L	L	L	A0
L	L	L	H	A1
L	L	H	L	A2
L	L	H	H	A3
L	H	L	L	A4
L	H	L	H	A5
L	H	H	L	A6
L	H	H	H	A7
H	X	X	X	None

Obr. 4.6: Pripojenie demultiplexora k GPIO

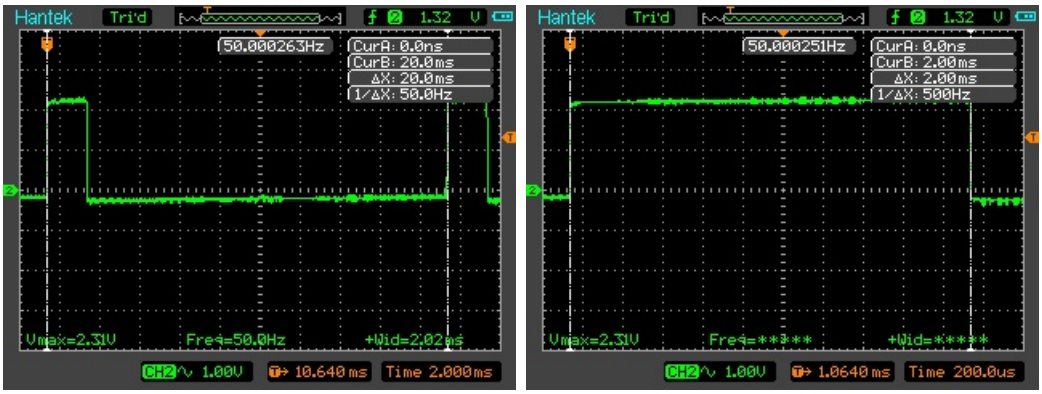


Obr. 4.7: GPIO utilita bez záťaže na procesore

Druhotné testy odhalili správanie utility pri viacnásobnom prepínaní módov. Pri prepnutý pinu na PWM keď sa už móde PWM nachádza spôsobí zmenu na deličkách a prepne PWM na výpočet pomocou prvého algoritmu.

Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrázkov. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs. Veľkým nedostatkom je, že ak je na serve väčšia váha a nedostáva signál tak závaže môže zmeniť polohu serva, čo sa môže stať pri použitý

[3]Najmenší merateľný diel.



Obr. 4.8: GPIO utilita so záťažou na procesore

GPIO utility	**na prázdno [ms]**	**pod záťažou [ms]**	**chyba [|µs|]**
perióda	20	20	<0.015
šírka	2	2	$<8^3$

Tab. 4.2: Výsledky merania: GPIO

demultiplexora a preto by som doporučil toto riešenie na zariadenia kde váha nie je priamo na serve alebo predimenzovať dané servo.

Knižnica piGPIO

Riešenie pomocou tejto knižnice a jazyku Python spočíva v zapnutí démona a zavolania funkcie na nastavenie šírky pulzu. V mojom prípade ešte v programe robím umelé spomalenie pre pohyb z dôvodu ilúzie plynulého pohybu. A to menením šírky po malých krokoch, po každom kroku nasleduje krátke aktívne čakanie. Krok a čakanie opakujem kým nedosiahnem požadovanej šírky.

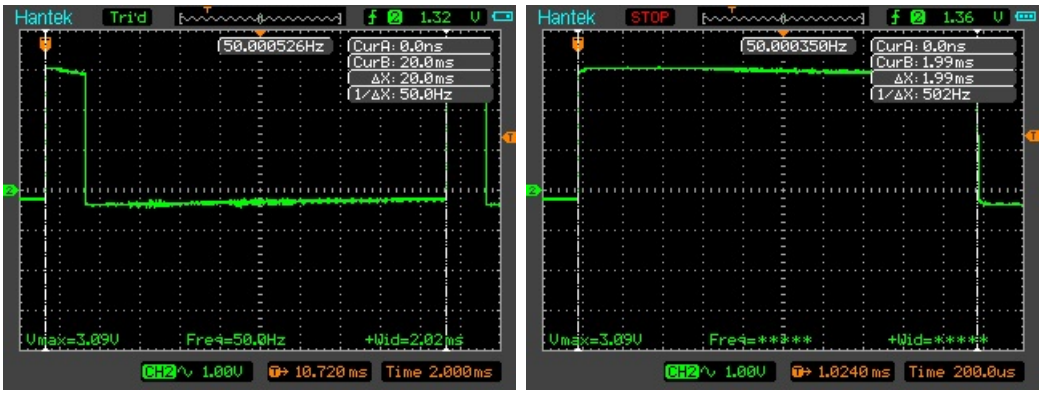
konfigurácia a implementácia k testom:

- **nastavenie frekvencie**: set_servo_pulsewidth má prednastavenú periódu 20ms
- **zmena dĺžky**: set_servo_pulsewidth(pin,1500), rozsah 1000 až 2000
- **počet kanálov**: 26, všetky GPIO pini
- **Pripojených serv**: 4
- **krok**: 2µs
- **dĺžka pulzu**: 2ms
- **testy**
  1. test naprázdno: CPU len s minimom programov
  2. test po záťažou: druhotný test platformy, t.j. prehrávanie HD videa bez HW akcelerácie = CPU 90+%
  3. test so zvukom: pripojenie jacku a prehranie zvuku, lebo 1 PWM kanál sa používa pre analógový zvuk
  4. sledovanie systémových zdrojov počas predošlých testov

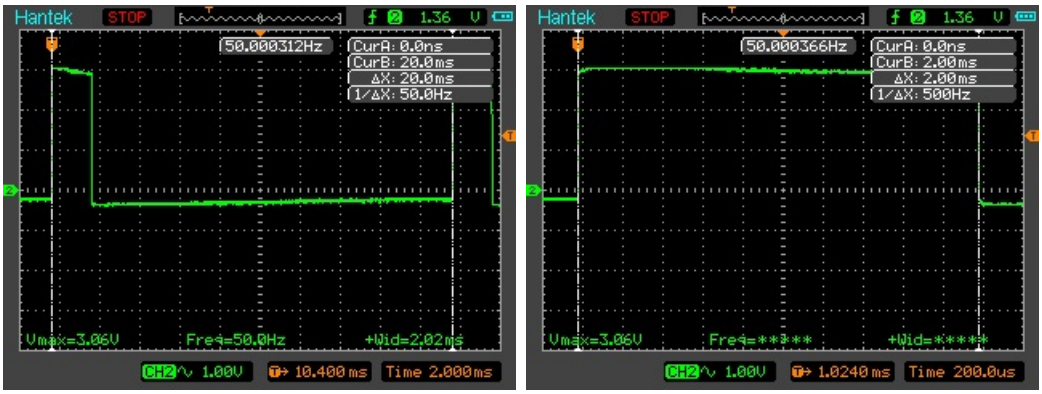
Tento program využíva priamo DMA pre generáciu takže očakávam generáciu stabilných pulzov.

Výsledky testov

Testy bez záťaže boli robené iba ako referencia k porovnaniu k záťažovým testom.

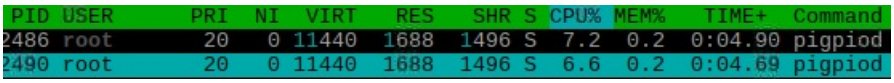


Obr. 4.9: piGPIO bez záťaže na procesore



Obr. 4.10: piGPIO so záťažou na procesore

Počas testov som zaznamenával zaťaženie systémových zdrojov, ktorých výsledok je zachytený obrázkom 4.11. Pri stress testoch sa táto záťaž tiež nemenila a ostávala v okruhu 7% pre jednu inštanciu a celkovo to robilo 14% zaťaženie na CPU. Dru-



Obr. 4.11: piGPIO systémové zdroje

hotné testy spôsobili len dlhšiu dobu prvotného zapnutia a vypnutia. Táto doba je pochopiteľná lebo je treba spustiť démona. Záťaž na procesore výstup nezmenila dostatočne pre detekciu servom. Najväčšia odchýlka na perióde bola 0,21µs. Pre šírku to bolo 8µs a menej. Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrázkov. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.

piGPIO	**na prázdno [ms]**	**pod záťažou [ms]**	**chyba [|µs|]**
perióda	20	20	<0.21
šírka	1,99-2	2	<8[4]

Tab. 4.3: Výsledky merania: piGPIO

[4]Najmenší merateľný diel.

Projekt piblaster

Pre tento projekt som nevytváral žiadne zdrojové súbory. Dôvodom je nepríjemný integrácie, nutný reštart systému po jeho použití a základ jadra zdielaný s pigpio z pôvodného servoblaster. Všetky testy som robil za pomoci terminálu. **konfigurácia a implementácia k testom:**

- **nastavenie frekvencie**: Zmenením parametrov pi-blaster.c
- **zmena dĺžky**: echo BCM=duty > /dev/pi-blaster, rozsah 0.050 až 0.1
- **počet kanálov**: 17, všetky GPIO pini
- **Pripojených serv**: 4
- **krok**: 2µs, minimum doporučené autorom projektu servoblaster z ktorého pi-blaster vychádza
- **dĺžka pulzu**: 2ms
- **testy**
    1. test naprázdno: CPU len s minimom programov
    2. test po záťažou: druhotný test platformy, t.j. prehrávanie HD videa bez HW akcelerácie = CPU 90+%
    3. test so zvukom: pripojenie jacku a prehranie zvuku, lebo 1 PWM kanál sa používa pre analógový zvuk
    4. opakovaný test 2 s vylúčením audia
    5. sledovanie systémových zdrojov počas predošlých testov

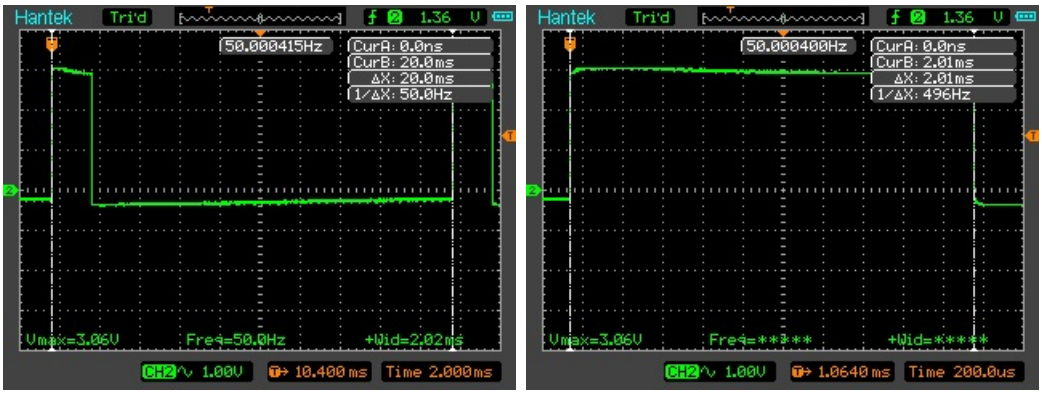
Podobne ako pigpio je použité DMA, tak predpokladám stabilné pulzy ale do testov pridávam test chovania pri použití periférie pre zvuk, lebo je použité PWM predvolene pre časovanie DMA.

Výsledky testov

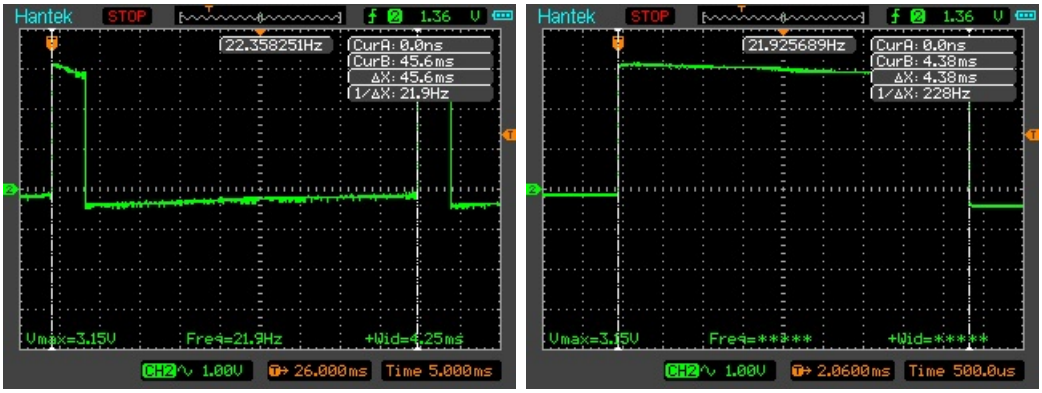
Testy na osciloskope dopadli rovnako ako aj pre iné hardvérové riešenia a to ostré hrany so správnou frekvenciou a šírkou. Testy bez záťaže boli robené iba ako referencia k porovnaniu k záťažovým testom. Druhotné testy pod záťažou potvrdily ďalší nedostatok spojený a prebraný od projektu servoblaster. Tento nedostatok je neošetrenie prístupu k prvému PWM kanálu pre generáciu analógového zvuku. Tento kanál je použitý projektom na časovanie DMA. Pri teste sa tento prejavilo ako skok frekvencie k 20Hz. Táto chyba sa objaví vždy ak sa pripojí akýkoľvek jack i keď sa nič neprehráva alebo ak sa dá prehrať súbor zo zvukom. Tomuto chovaniu sa dá vyhnúť zmenou konfigurácie systému aby predvolene používal digitálny výstup pre HDMI namiesto analógu(predvolený), čo potvdilo opakovanie druhého testu, ktorý obsahoval zvuk[5].

[5]žiaden zvuk nebol prehrávaný

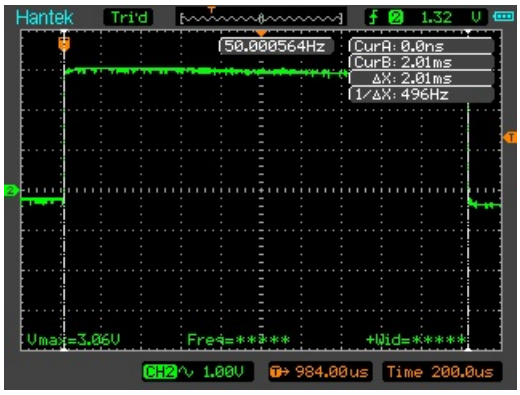
[6]Graf je zhodný s testom bez audia



Obr. 4.12: pi-blaster bez záťaže na procesore



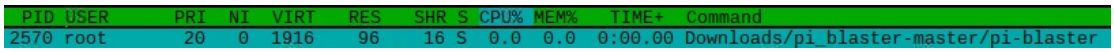
Obr. 4.13: pi-blaster so záťažou na procesore



Obr. 4.14: pi-blaster so záťažou na procesore s HDMI výstupom predvoleným[6]

Počas testov som zaznamenával zaťaženie systémových zdrojov, ktorých výsledok je zachytený obrázkom 4.15. Pri stress testoch sa táto záťaž tiež nemenila. Zaujímavosťou je, že tento démon nedostal žiadny čas procesoru.

[7]Najmenší merateľný diel.



Obr. 4.15: pi-blaster systémové zdroje

piblaster	**na prázdno [ms]**	**pod záťažou [ms]**	**chyba [|µs|]**
perióda	20	20	<0.21
šírka	2-2.01	2	$<8^7$

Tab. 4.4: Výsledky merania: piblaster

Najväčšia odchýlka na perióde bola 0,21µs. Pre šírku to bolo 8µs a menej. Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrázkov. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.

4.2 Riešenia platformy FITKIT

Bola použitá jeho verzia 2.0 s kombináciou Windows 7 predpripravenou FIT VUT. Pre túto platformu som sa rozhodol testovať po jednom riešení pre FGPA a MCU bez ich kombinovania. Takto som sa rozhodol lebo je možné použiť podobný čip, ktorý nie je súčasťou FITKITu samostatne.

FPGA

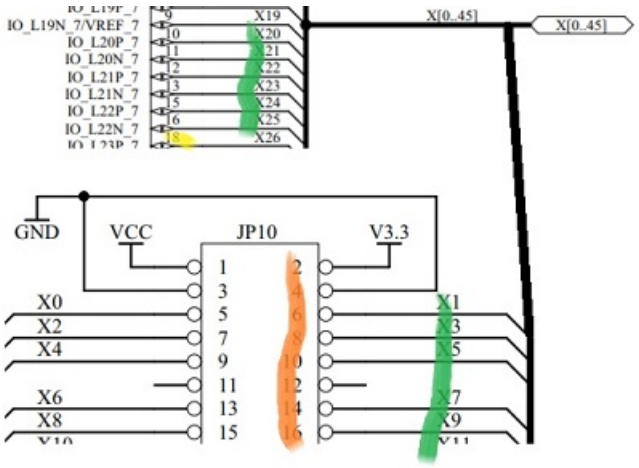
FPGA riešenie by malo byť schopné generácie pulzov bez záchvevov pomocou vytvorených logických obvodov v jeho vnútri. Takže by nemalo byť ovplivnené ani inou komunikáciou ako MCU s prerušeniami.

konfigurácia a implementácia k testom:

- **nastavenie frekvencie**: pomocou 19bit čítača na zníženie základnej 7.3728 Mhz frekvencie
- **zmena dĺžky**: Porovnanie registra s čítačom, rozsah x"1ce7"až x"39ce", t.j. 1-2ms
- **počet kanálov**:4, možné všetky pini JP10[8]

[8]teoreticky, jedena z možných nadstavieb na túto prácu

- **Pripojených serv**: 4
- **kanály s posunom**
- **krok**: 18µs, zodpovedá môjmu kroku 0x85, krok ide znížiť pod 1µs
- **dĺžka pulzu**: 1-2ms
- **testy**
    1. test naprázdno

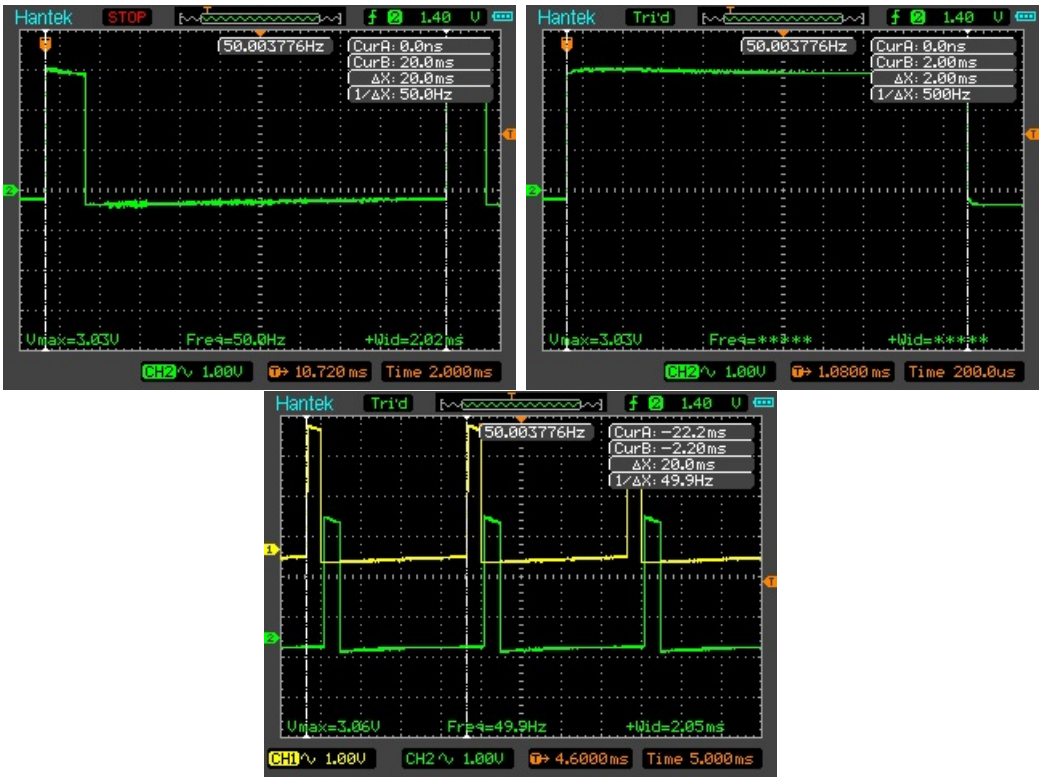


Obr. 4.16: Vizualizácia mapovania FPGA pinov na konektor JP10

Vizualizácia predstavuje mapovanie zbernice JP10, jej pinov(oranžová) na prepoje(zelená) na PXX čísla(žltá) použité v ucf súbore. Od riešenia očakávam ostré pulzy, lebo kódom sa vytvára zoskupenie logických obvodov, rovnako ako keby sa pospájali externé čipy medzi sebou.

Výsledky testov

Testy na osciloskope potvrdili stabilitu signálov. Najväčšia odchýlka na perióde bola 0,15µs. Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrázkov. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.



Obr. 4.17: FITKIT FPGA prvý a druhý kanál osciloskopu

FPGA	**na prázdno [ms]**	**pod záťažou [ms]**	**chyba [|µs|]**
perióda	20	20	<0.15
šírka	2	2	<8[9]

Tab. 4.5: Výsledky merania: FPGA

Testy ktoré by prekazovaly použiteľnosť tohto FPGA riešenia by mali skúsiť zaťaženie systémových zdrojov. Iný typ testu by mohol skúšať maximálny počet pripojiteľných serv, bez nutnosti externej batérie alebo DC adaptéra a či má FITKIT dosť pamäte pre všetky signály serv.

[9]Najmenší merateľný diel.

MCU

Toto riešenie trpí nielen pre jeho ovplyvniteľnosť inou záťažou ale aj ako pri rozširovaní pomocou demultiplexora pre GPIO riešený z platformy Raspberry je možné mať aktívny len jeden kanál.

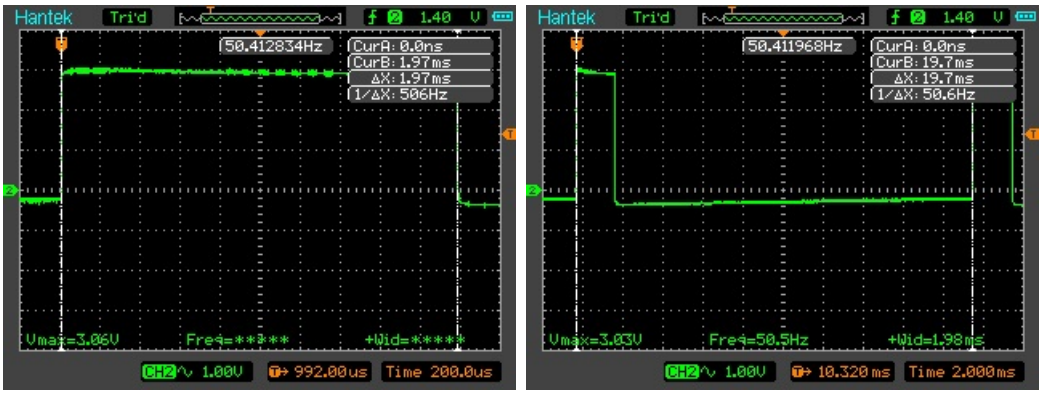
konfigurácia a implementácia k testom:

- **nastavenie frekvencie**: ticks = 32768/50/SAMPL
- **zmena dĺžky**: CCR0 += ticks*duty_cycle, manipulujem len s duty
- **počet kanálov**: 4 pini na JP9
- **Pripojených serv**: 4
- **krok**: 0,003ms, čo je možné zmeniť premenú SAMPL
- **dĺžka pulzu**: 2ms
- **testy**
    1. test naprázdno: žiadna komunikácia

Od MCU očakávam možné záchvevy na signáli, z dôvodu použitia prerušení na časovanie a možného príchodu prerušenia s vyššou prioritou.

Výsledky testov

Osciloskop potvrdil predpoklad nestability riešenia, za pomoci prerušení. Dôvodom tohto tvrdenia je chyba periódy s maximom 0,163ms a zároveň jej neustály pohyb približne o 0,08ms. Pre šírku robila odchyľka 0,02ms. Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrazovky. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.



Obr. 4.18: FITKIT MCU bez zátaže

MCU	na prázdno [ms]	pod záťažou [ms]	chyba
perióda	20	20	<163ns
šírka	2	2	0.02ms

Tab. 4.6: Výsledky merania: MCU

Medzi ďarlšie testy k MCU by bolo možné zaradiť: test simulujúci komunikáciu a test simulujúci komunikáciu s ďalšou výpočetnou záťažou na procesore.

4.3 Riešenia platformy Personálny počítač

Použitá bola doska Pololu mikro maestro 6. Miesto externej batérie som použil na napájanie Raspberry alebo fitkit, Raspberry bolo použité pri prvotných testoch

konfigurácia a implementácia k testom:

- **nastavenie frekvencie**:
  - za pomoci grafického rozhrania
  - skript – frekvencia je prednastavená
- **zmena dĺžky**:
  - za pomoci grafického rozhrania
  - skript – 8000 [servo_num] servo, rozsah 4000 až 8000
- **počet kanálov**: 6
- **Pripojených serv**: 5
- **krok**: 0,25µs
- **dĺžka pulzu**: 1-2ms
- **nastavenie rýchlosti**: 30 [servo] speed
- **testy**
  1. test naprázdno: žiadna komunikácia

2. test na posun medzi kanálmi

Od dosky očakávam dobré výsledky lebo bola na tento účel navrhnutá, má vlastný časovač aj mikrokontróler.

Výsledky testov

Po pripojený k osciloskopu bol signál s ostrými hranami aj požadovaných parametrov. To znamená perióda 20ms aj keď s odchýlkou 21µs a šírkou signálu 2ms.

Počas testov bola amplitúda 4,8V, čím sa ukázalo že doska berie napájanie bez toho aby ho štandardizovala na 3,3v, modulovaný signál totiž vzniká z 5V napájania z raspberri pi.

Ďalej sa prejavilo že čip používa posunu o 3.34ms medzi jednotlivými kanálmi na ušetrenie energie a procesorového času.

Obr. 4.19: Micro maestro prvý a druhý kanál osciloskopu

PC: Micreo maestro	na prázdno [ms]	pod záťažou [ms]	chyba [|µs|]
perióda	20	20	<21.5
šírka	2	2	<8[10]

Tab. 4.7: Výsledky merania: PC

Chyba periódy je vypočítaná zo zachytenej frekvencie funkciou counter. Táto hodnota je zobrazená v strede obrazovky. Chyba v dĺžke signálu je získaná z osciloskopu za pomoci trigeru a kurzorov a najmenší možný merateľný posun, keď bol celý signál viditeľný je 8µs.

Zhrunutie testov

Pre platformu FITKIT som skúmal a popísal po jednom riešení pre FPGA a MCU. Záverom je že MCU s prerušeniami nie je dobré riešenie ak je nutné komunikovať s okolím alebo ak je na procesore iná záťaž tak daný čip nemá dostatočný výkon. Preto je určite lepšie riešenie s využitím FPGA nielen že je stabilnejšie a poskytuje viac možných samostatných serv ale je aj efektívnejšie z pohľadu napájania. Posledná zmienka k FITKITu je že na MCU je možné použiť DMA kanály pre generáciu pulzov ale toto riešenie som bližšie neskúmal.

Na platforme personálneho počítača je vždy potrebná externá doska ktorá poskytuje časovanie. Veľmi dobrou možnosťou je doska Micro maestro od Pololu kde najmenšia poskytuje šesť kanálov. Pre komunikáciu s personálnym počítačom je nutné si nainštalovať ovládač. Odporúčam najskôr nainštalovať ovládač a až nasledovne pripojiť dosku, vyhnete sa možnému problému s všeobecným USB ovládačom a strate informácie o COM kanále. Plus je treba spraviť prvotnú konfiguráciu pre správnu funkciu skriptov

Platforma arduino má najviac fór a článkov na ovládanie serv za jeho pomoci. Túto platformu som netestoval lebo je už dobre preskúmaná a zdokumentovaná. Samotné výstupy PWM kanálov arduina je šesť na troch rôznych časovačoch, ktoré neposkytujú frekvenciu používanú modelárskymi servami. Z pomedzi riešení použiteľných sa javili riešenia pomocou konverzie čítaného napätia idúceho cez potenciometer a jeho prepisu na požadovaný rozsah dobrou voľbou. Ďalšie riešenia boli robené za pomoci knižníc servo.h a PWM.h. Samotné výstupy PWM kanálov arduina aj to šesť kanálov na troch rôznych časovačoch tak neposkytujú frekvenciu používanú modelárskymi servami.

Platforma Raspberry Pi bola podrobená najviac testom a skúmal som na nej aj najväčšie množstvo riešení. Okrem riešený som testoval aj samotnú stabilitu PWM

[10]Najmenší merateľný diel.

signálov pri frekvencii 5kHz kde som predpokladal že sa záchvevy zobrazia skôr než na testovaných riešeniach. Najlepšie z testovaných dopadlo riešenie pigpiod ktoré udržovalo stabilný signál pod záťažou a poskytuje PWM na všetkých GPIO pinoch za pomoci DMA. Nasledujú riešenia: pi-blaster ak je zabezpečené, že sa neprehráva zvuk a je odpojený jack tak poskytuje veľmi dobré výsledky. Pi-blaster poskytuje PWM na všetky GPIO pini za pomoci DMA. Obe riešenia pi-blaster aj pigpiod vychádzajú z riešenia servoblaster z ktorého preberajú základ časovania DMA. U utility GPIO je vhodné spomenúť že má len dva PWM kanály čiže len dva pini, čo sa týka signálu tak je to PWM priamo z čipu ktorého stabilitu som testoval aj na vyšších frekvenciách a vďaka tomu dosahovalo najlepšiu stabilitu periódy. Posledné hardvérové riešenie, ktoré nebolo kompatibilné s mojou verziou je RPIO.GPIO a preto bolo vypustené z testov. Testy som robil aj na softvérovom PWM Rpi.GPIO pre referenciu chovania.

Pri testoch platformy sa prejavila chyba v napájacom okruhu cez USB.

Testy som robil väčšinou v troch fázach. Prvé len na školou poskytnutým osciloskope kde som testoval len základnú frekvenciu. V druhej fáze boli testy na kvalitnejšom osciloskope na prázdno a pod syntetickou záťažou. Pod záťažou sa Raspberry prehrialo a vyplo čo dovolilo odhaliť spomínanú chybu napájania. Posledné testy boli pod mnou vyrobenou záťažou a na osciloskope. Posledná séria testov odhalila nové nedostatky niektorých riešený.

Pre budúce použitie by som odporučil kombináciu Raspberry a polulo maestro spojených uartom alebo Raspberry s externým DMA/MCU a využívať Raspberry na riadiaci program a externý čip pre generáciu.

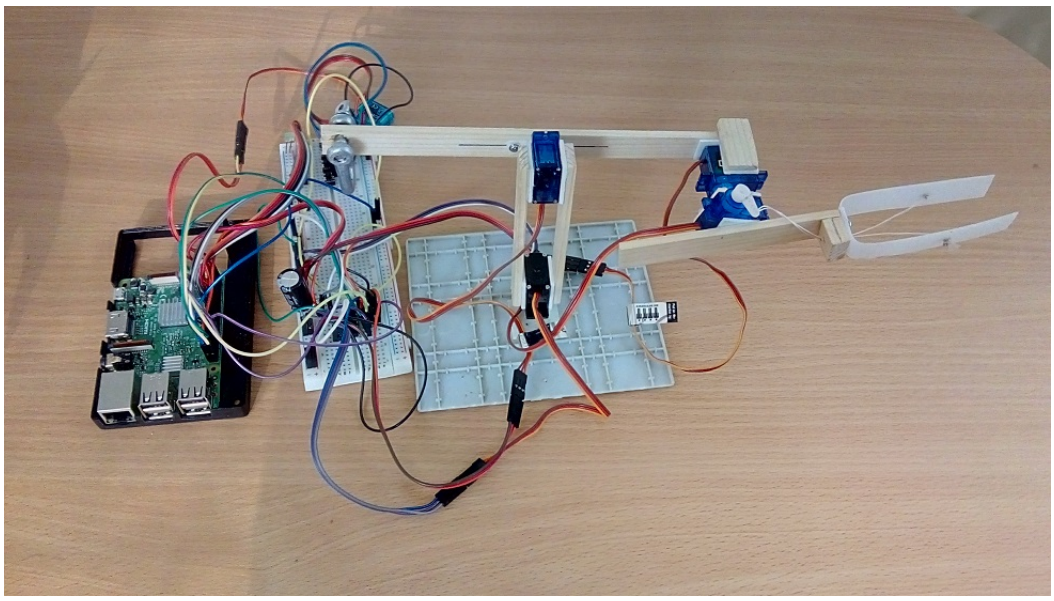
* Najmenší merateľný diel

Raspberry Pi		**GPIO utility**	**servo blaster**	**pigpio**	**pi-blaster**
	resource	PWM	PWM or PCM for DMA timing	PWM or PCM for DMA timing	PWM for DMA timing
	channels	2	17+	17+	17+
	Langluage	BASH / any that can use sys call	Py or C	Py or C	any that can write to file
	step	<2,5µs	2µs	2µs	2µs
	period accurency	<0.015µs	–	<0.21µs	<0.21µs
	width accurency	<8µs*	–	<8µs*	<8µs*
	note	audio uses PWM channel 1	base for pigpiod and pi-blaster		audio creates problems
FITKIT		**FPGA**	**MCU interupts**	**MCU DMA**	
	channels	all pins	4+	all pins	
	Langluage	VHDL	C	C	
	step	<18µs	3µs		
	period accurency	<0.15µs	<163µs		
	width accurency	<8µs*	0.02ms		
	note		can't be under load		
arduino		**read potenciometer**	**pwm.h**	**servo.h**	
	pins	all pins	all pins	all pins	
pc	**step**		<1µs	<2µs	
		micro maestro	**note**		
	channels	6 / 12 / 18 / 24	supports UART		
	step	<0.25µs			
	period accurency	<21.5µs			
	width accurency	<8µs*			
	Langluage	Pololu own / any with USB support	needs external battery for servo pwr		

Tab. 4.8: Zhrnutie riešený

4.4 Demo aplikácia

Moje demo bude robotická ruka z troch uzlov. Pri návrhu a konštrukcii som dbal na tieto už spomenuté body: jednoduché vymieňanie medzi jednotlivými riešeniami, jednoduchá konštrukcia a samozrejme menšia finančná náročnosť.

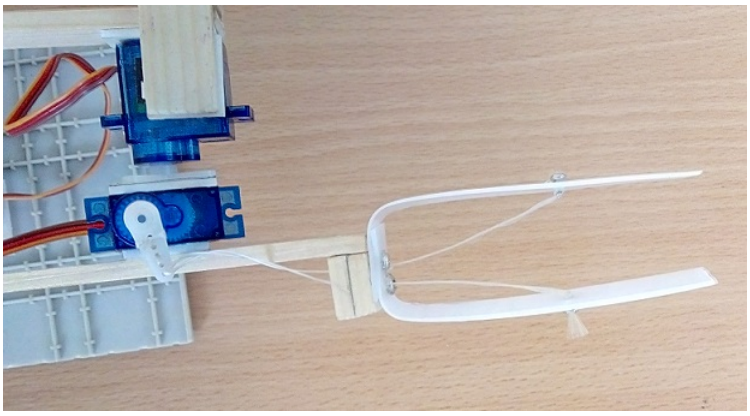


Obr. 4.20: Drevená demo aplikácia

Pre zostavenie ramena som potreboval: drevení nosník 4*15mm asi 1m sa predával, využil som asi polovicu, 3*9g serva, 1*12g, plastovú dózu zo zmrzliny, obojstrannú lepiacu pásku, protizávažie, nôž, skrutkovač a nylonovú niť. Čo sa týka samotnej konštrukcie tam som sa snažil držať nákresu ktorý som vytvoril a sprehľadnil pre prezentáciu. Pri samotnej konštrukcii som robil zmeny podľa uváženia ako aj využitia protizávažia pre zmenšenie námahy pre pohyb samotného serva. Ďalšou úpravou je zdvojenie nosníkov na prvom uzle kôli stabilite celého ramena. Nedostatok ktorý som objavil po skonštruovaní je že servo ovládajúce zatváranie klieští nemá dostatočnú silu na udržanie ťažších predmetov lebo nedostatočne zovrie kliešte. Úprava ktorá by uvoľnila určitú záťaž je spojiť ovládaciu niť to tvaru T.

Druhý návrh demo aplikácie pomocou programu Solid edge a Blender

Druhý návrh demo aplikácie pomocou programu Solid edge a Blender je založený na základe zistených poznatkov z prvého dreveného modelu a podnetu od vedúceho som vytvoril 3D model demo aplikácie. Pre návrh 3D modelu som Použil program Solide Edge. Solid edge je program plný nástrojov pre riešenie návrhu výrobkov. Z týchto nástrojov bol využitý najme 2D a 3D kresliaci priestor.



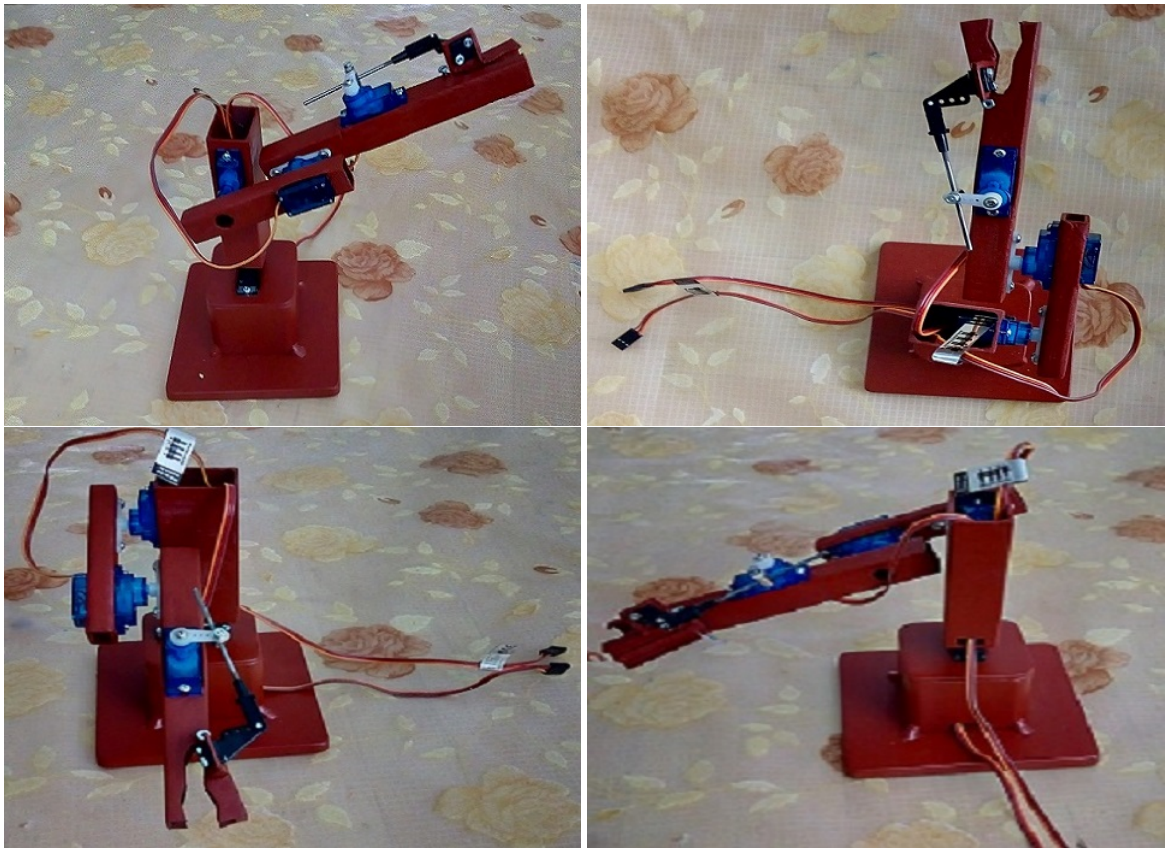
Obr. 4.21: Drevená demo aplikácia pohľad zhora

S výrobou modelu v profesionálnom programe Solid edge my pomáhal otec lebo v danom programe často pracuje. Pri kreslený sa zmenil celý návrh s prihliadnutím na to, že súčiastky modelu budú tlačené na 3D tlačiarni s maximom schopnej tlačenej plochy 16x16cm. Oproti pôvodnému drevenému modelu bolo spravených mnoho zmien. Základňa zmenená na kockový tvar ktorý bude možné vytlačiť a vložiť do nej závažie. V mojom prípade bude základňa kovová. Ramená zmenené do tvaru dutého kvádru. Takto do nich môžem zasunúť servá a zároveň ťahať kabeláž vnútrom pre úhľadnejší cable manažment. Pôvodné „kliešte“ som mohol len zatvárať, pretože sa otvárali len vnútorným napätím materiálu. Nový spôsobom pre ktorý som sa rozhodol je mať jednu časť pevnú a hýbať len jedným tiahlom ku ktorému bude pripojená časť na čape.

Po návrhu a nakresleniu modelu boli súčiastky exportované k 3D tlači do .stl súboru. Stl súbory sú štandard pre 3D tlač ale pred ich samotou tlačou ich bolo nutné ešte upraviť. Úprava zahŕňala zmenu merítka modelu, kontroly úplnosti modelu t.j. či je model jeden kus a nemá chybné nadpojenia. A poslednú úpravu ktorú som aplikoval bola pridanie hmoty modelu tu som zvolil hrúbku stien 1,5mm. Všetky tieto úpravy som aplikoval v open source programe Blender. Blender je veľmi rozsiahly a populárny nástroj pre 3D tvorbu s rozhraním nad OpenGL. Blender má podporu pre modelovanie, animácie, simulácie, vykresľovanie, skladanie zvukov, video editing, či tvorbu hier a ďalšie. Pre jeho široké použitie má podporu pre veľké množstvo súborov. Ďalej patrí medzi cross-platform programy a je možné ho používať na Linuxe, Windowse aj Macku. Tento program som využil lebo som ho v minulosti použil pre SOČ v Unity 3D a zároveň má podobné rozhranie ako Unity.

Samotnú tlač zabezpečil vedúci práce. Po vytlačení nasledovala konštrukcia podľa návrhu. Pri konštrukcii som zistil že i keď je stena 1,5mm hrubá tak nie je to plný materiál a v určitých častiach bol výtlačok dutý. Z tohto dôvodu sú všetky steny obalené lepidlom a posypané stužovacým práškom, ktorý používam na lepenie plas-

tov kde chýba materiál, týmto je zabezpežená pevnosť a integrita modelu. Model som natrel červenou základovou farbou, túto farbu som vybral lebo sa príliš neleskne a nelúpe sa.



Obr. 4.22: Výsledný model demo aplikácie

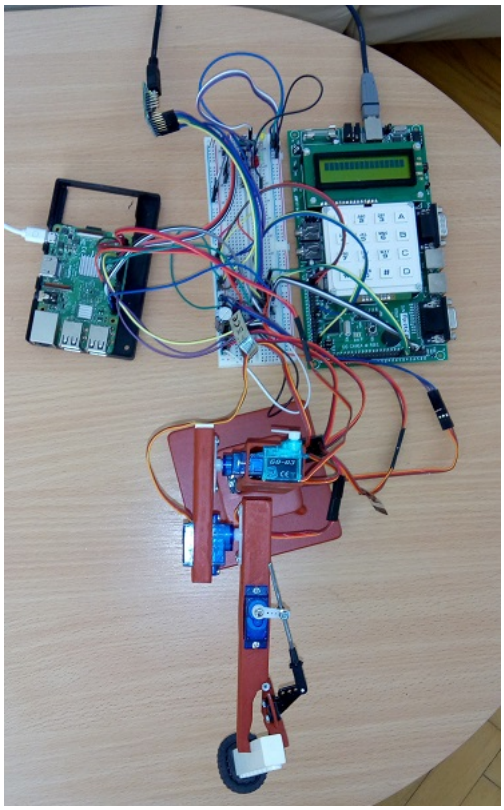
Súbory k tlači i návrh nájdete na odovzdanom médiu. S súbore print sú stl súbory mnou predpripravené k tlači, doporučujem si ich skontrolovať. Medzi ostatnými súbormi je an 3D model vo formáte pdf aj originálne súčiastky z Solid Edgeu a s pôvodne exportovanými stl súbormi. Ak si chcete model vytlačiť je treba brať v úvahu otvory predpripravené pre mnou použité serva: GO-13 a 3x SUMO 1199BU.

Pripojenie dema k platformám

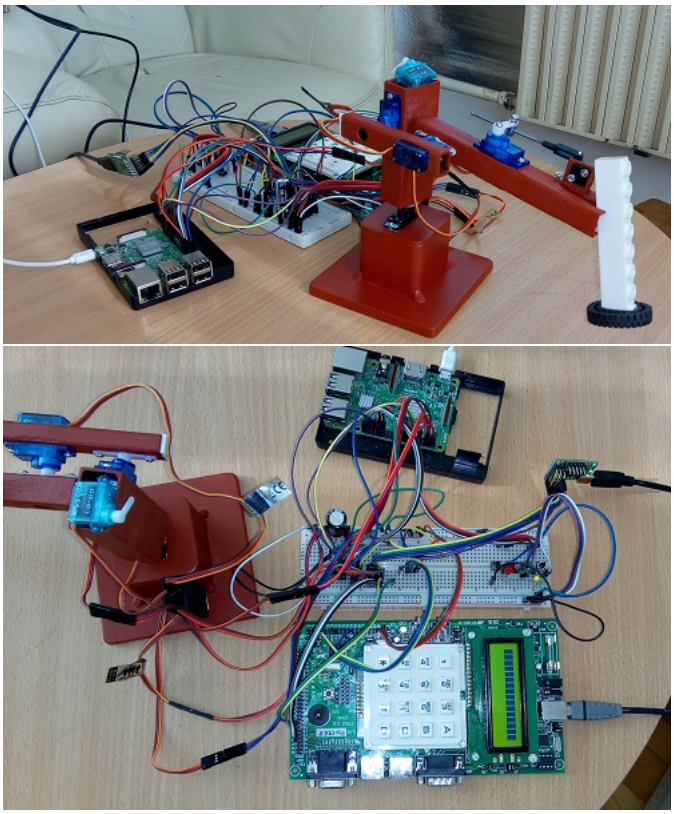
Pre ukážky jednotlivých riešení som prepojil platformy cez nepájavé pole. Napájanie je z pinou zbernice JP9 na FITKITe. Všetky platformy zdieľajú zem. Demo nie je pripojené k všetkým riešeniam zároveň. Riešenia sú rozdelené do troch skupín:

- RPI riešenia: RPi.GPIO, GPIO, pigpiod
- FITKIT FPGA, Pololu micro maestro
- FITKIT MCU prerušenia

V zapojení na obrázku je demo pripojené k FITKITu FPGA a Pololu micro maestru. Pre tieto platformy a riešenia som sa rozhodol pre ich dobré výsledky počas testov, s výnimkou RPi.GPIO(sw PWM) ktoré je pripojené na samostatné servo a MCU s prerušeniami, ktoré síce nie je vhodné ako reálne riešenia ale dá sa použiť k demonštrácii pohybu.



Obr. 4.23: Pripojenie demoaplikácie k platformám, zapojené pololu maestro(stand by) a FITKIT fpga(signál) 1/2



Obr. 4.24: Pripojenie demoaplikácie k platformám, zapojené pololu maestro(stand by) a FITKIT fpga(signál) 2/2

Demo s FITKITom

Zapojenie káblov je popísané v tabuľke nižšie. Pre ovládanie som zvolil vstavanú klávesnicu na fitkite. V MCU používam klávesy A, B, C, D na zvolenie serva, ktorému chcem poslať signál. Klávesy 1, 2 a 3 sú krajné polohy a stred. Samotné ovládanie posunu je namapované na klávesach 4 a 6. FPGA mapuje klávesy 7, 9 pre servo 4, 4 a 6 pre servo 2 a tak má každé servo samostatné ovládanie bez nutnosti medzi nimi prepínať.

Demo fitkit						
	Zem	Napájanie	Signál			
FPGA	JP9 40	JP9 39	JP10 39	JP10 41	JP10 43	JP10 45
MCU	JP9 40	JP9 39	JP9 7	JP915	JP9 23	JP9 37

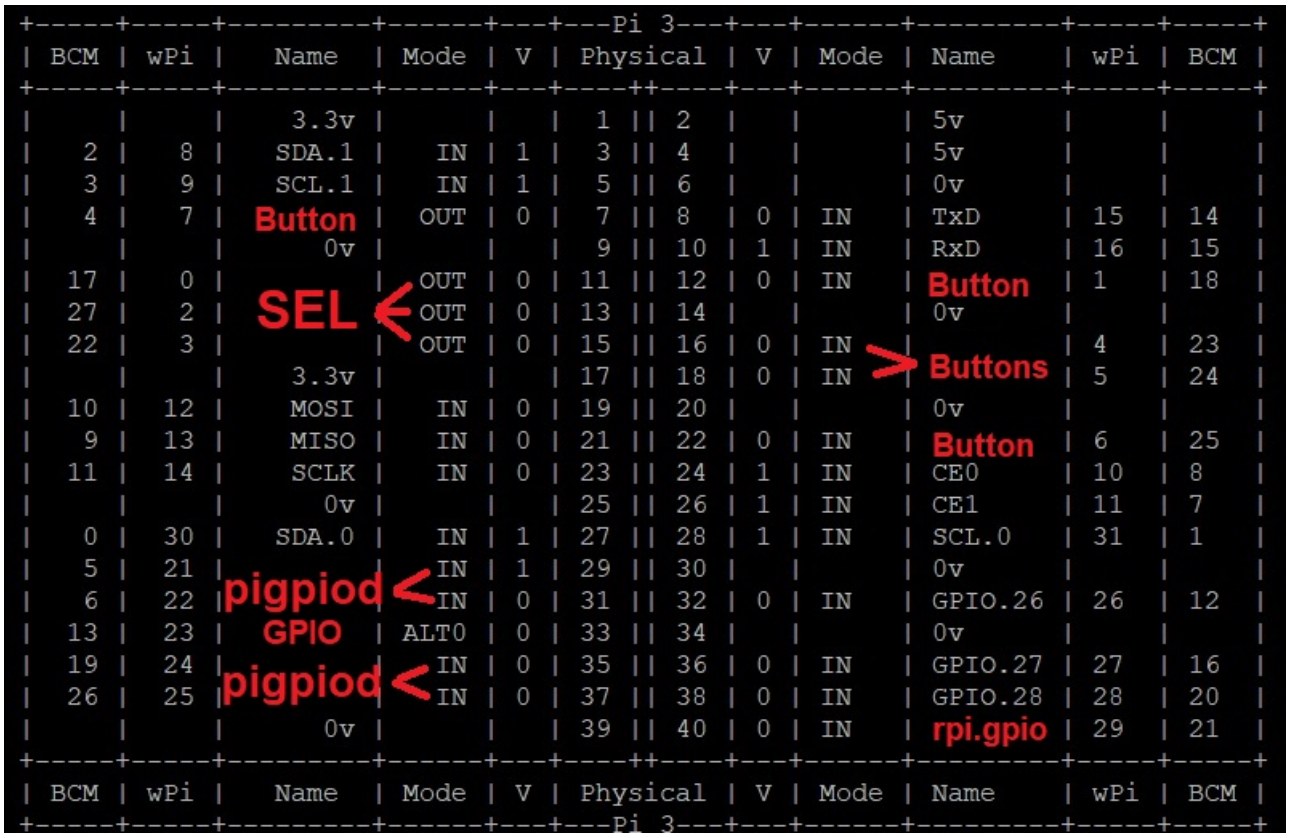
Tab. 4.9: FITKIT: Aktívne pni konektorov pre demo

Demo s PC

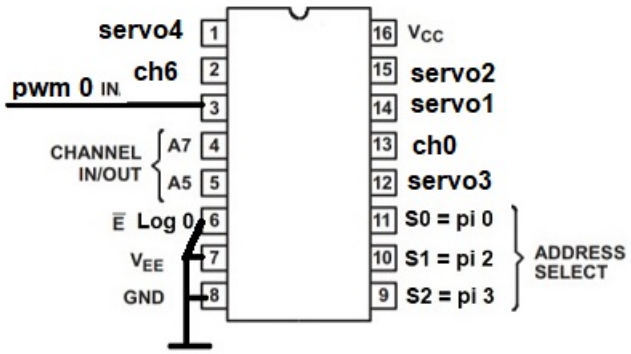
Pololu micro maestro bolo pripojené iba prinamo na signál k servam a samotné napájanie zabezpečoval FITKIT. Pre demo som použil grafické rozhranie od výrobcu.

Demo s RPi

Demo na Raspberry Pi som spravil skriptom, z ktorého spúšťam jednotlivé riešenia. Výber riešení som zariadil stlačením jedného z externých tlačítiek. Možnosti sú nasledovné. Softvérové PWM na ktorom je len jedno servo pre ukážku chvenia. Pod druhým je GPIO utilita. Tretie skrýva démona/ovládač pigpiod. Posledné tlačidlo nie je pripojené lebo riešenie pi-blaster pri testoch ukázalo nedostatok v návrhu a zároveň blokuje prepínanie medzi riešeniami a bol by nutný reštart.



Obr. 4.25: Pripojenie demoaplikácie na Raspberry



Obr. 4.26: Pripojenie demoaplikácie na demultiplexor

ozdielom v zapojení demultiplexora oproti zapojeniu u testov je prepojenie aktivačnej nohy na zem lebo i tak za zapína logickou nulou. Ďalej na nohách pomenovaných servoX boli pripojené výstupy ovládané riešeným pigpiom, ktoré vždy zaručilo odpojenie PWM z GPIO riešenia.

Záznam k Demu

Výsledný záznam k demu som zachytil za pomoci kamery VEGA 5 fun na statickom stojane. Pomocou riešení z predchádzajúcich sekcií menovite ako sú zoradené v zázname: PC s mikro maestrom, FITKIT na MCU, FITKIT na FPGA, Raspberry Pis knižnicou pigpio a Raspberry Pis GPIO utilitou. Záznam bol upravený a zostrihaný v programe Blender, ktorý bol použitý i pri návrhu k tlači. Hudba pochádza od Royalty Free Music from Bensound[12].

Výsledný záznam bol nahraný na platformu YOUTUBE a je dostupný na adrese: `https://youtu.be/zP1TQnDVWzc`

[12] `www.bensound.com`

5 Závěr

Cieľom tejto bakalárske práce bolo skúmanie spôsobu ovládania modelárskych serv z rôznych počítačových platforiem. Tento cieľ bol splnený, zároveň boli splnené čiastkové podciele.

Prvý podcieľ bol preštudovanie spôsobu riadenia modelárskych serv a možností ich ovládania počítačovým systémom, zhrnutie tohto preštudovania tématu sú v kapitolách 1 a 2. Návrh robotického ramena som popísal v kapitole 4 sekcii 4, Tento návrh dreveného ramena bol určený ako základ k testom na jednotlivých platformách. Ďalej som popísal možnosti ovládania modelárskych serv z platforiem: FITKIT, Raspberry Pi a z personálneho počítača. Tento popis sa nachádza v kapitole 3. Druhá iterácia návrhu robotického ramena bola založená na poznatkoch z dreveného návrhu a z testov jednotlivých platforiem.

Medzi zajujímavé výsledky patrí presnosť dosiahnutá riešeniami na platforme Raspberry Pi menovite: piGPIO, GPIO utility a pi-blaster[1], dosiahnutá presnosť na šírke impulzov bola porovnateľná s komerčným riešením micro maestro a to 8µs čo bola najmänšia jednotka na osciloskope pre daný signál. Presnosť 8µs sa podarilo dosiahnuť aj na FITKITe s použitím FPGA. A k ideálnej perióde 20ms bola najbližšie utilita GPIO[2] a to o 0,1µs.

Tiež by som zmienil projekty, ktoré sme s majiteľom osciloskopu Miroslavom Martákom diskutovali počas meraní. Prepojenie rpi s gps modulom z vyzdvihnutých meteorologických balónov hlavne ich komunikácie lebo z tohto modulu sa dajú získavať nielen gps súradnice ale napríklad aj čas z atómových hodím. Nadstavba na RC podvozok z rpi, pololu micro maestra a kamery VEGA(alternatíva GOpro).

Nadväzujúce práce by mohli moju prácu rozšíriť, či prehĺbiť o testovanie MCU a FPGA FITKITu pod záťažou buď z výpočtu alebo komunikáciu po zbernici. Ďalej by bolo možné skúmať prepojenie serv a personálneho počítača inou doskou ako je Pololu micro maestro.Tiež by mohlo byť zaujímavé skúmanie maximálneho počtu pripojiteľných serv k týmto platformám, lebo riešenia v tejto práci maly pripojené maximálne 5 serv súčasne a udávam teoretický počet maximálneho počtu.

[1] len v istých situáciách

[2] Tiež pod menom WiringPi

Literatura

[1] Jan Malášek: *"Servo control interface in detail"*,Pololu Corporation, 9/February/2011, Web 2018/December
https://www.pololu.com/blog/17/servo-control-interface-in-detail

[2] *Návody – FITKIT*, Fakulta informačních technologií VUT Brno 2006, Web 2018/November
http://merlin.fit.vutbr.cz/FITkit/navody.html

[3] Dana Sorani *Electrical Engineering forum*, Stack Exchange Inc 19/September/2014, Web 2018/December
https://electronics.stackexchange.com/questions/129961/how-to-get-the-pwm-frequency-and-duration-of-each-pulse

[4] *High speed pulse generation*,RASPBERRY PI FOUNDATION 27/Jan/2014 , Web 2018/October
https://www.raspberrypi.org/forums/viewtopic.php?t=67741

[5] *RPi Low-level peripherals*,elinux.org 4/Nov/2013, Web 2018/October
https://elinux.org/RPi_Low-level_peripherals#C

[6] *pigpio library*abyz.me.uk 22/Oct/2013, Web 2018/December
http://abyz.me.uk/rpi/pigpio/download.html

[7] *Stack Exchange forum: Control Hardware PWM frequency*, Exchange Inc 8/February/2013 , Web 2018/December
https://raspberrypi.stackexchange.com/questions/4906/control-hardware-pwm-frequency

[8] *Physical computing with Raspberry Pi*, cl.cam.ac.uk 2013/09/17, Web 2019/March
https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/cheat_sheet/

[9] *github: pi-blaster*,GitHub, Inc. 19/August/2012, Web 2019/January
https://github.com/sarfata/pi-blaster

[10] Juan Felipe Proaño *FPGA VHDL PWM pulse width modulation waveshare development board implementation xilinx Spartan 3*, quitoart.blogspot.com, 2/November/2017, Web 2018/November
http://quitoart.blogspot.com/2017/11/fpga-vhdl-pwm-pulse-width-modulation.html

[11] *Demultiplexer datasheat, Texas Instruments Incorporated February/2017*, Web 2018/December
http://www.ti.com/lit/ds/symlink/cd74hc4051.pdf

[12] *MSP430x2xx family guide* Texas Instruments Incorporated July/2013, Web 2019/January
https://www.ti.com/lit/ug/slau144j/slau144j.pdf

[13] sergio.b *BCM2837 ARM Peripherals*, Tucows Domains Inc. 2012, Web 2019/April
https://cs140e.sergio.bz/docs/BCM2837-ARM-Peripherals.pdf

[14] Andrei Chichak: *DMA - A LITTLE HELP FROM MY FRIENDS*, EPAG Domainservices GmbH 2017/2/22, Web 2018/December
https://www.embedded.fm/blog/2017/2/20/an-introduction-to-dma

[15] *Using DMA for pulse counting on Kinetis*,Arm Limited 01/2015, Web 2019/February
https://os.mbed.com/media/uploads/GregC/an5083-using_dma_for_pulse_counting.pdf

[16] Jason Long Embedded in Embedded,Elektor International Media BV United Kingdom 2018 , ISBN: 978-1-907920-73-8
http://www.ganssle.com/articles/adma.htm

[17] *RPi.GPIO 0.6.5*,Python Software Foundation 2015-07-24, Web 2018/November
https://pypi.org/project/RPi.GPIO/

[18] *SERVO SIZE EXPLAINED*,Motion RC February/18/2018, Web 2018/November
https://www.motionrc.com/blogs/motion-rc-blog/servo-size

[19] John Salt *Your Guide To RC Servos*,RC Helicopter Fun.com April/2018 Web 2018/December
https://www.rchelicopterfun.com/rc-servos.html

[20] *Jak fungují modelářská serva*webz.cz 07.07.2015, Web 2018/November
http://vlastikd.webz.cz/bastl/serva.htm

[21] *Serva* 2011-02-11,PELIKAN DANIEL Web 2018/December
http://www.pelikandaniel.com/?sec=page&id=22

[22] Howard Eglowstein *Introduction to Servo Motors*,Science Buddies 2013, Web 2018/December

https://www.sciencebuddies.org/science-fair-projects/references/introduction-to-servo-motors

[23] Josef Paláček *Umíte si dobře vybrat servo: e-book*,modelorlicko.com 15/8/2015 ,Web 2019/Jan
http://www.modelorlicko.com/phocadownloadpap/03-rady-navody/prislusenstvi/Umite-vybrat-servo.pdf

[24] *SERVO MOTORS*,robotiksistem.com 209-05-11, Web 2019/february
http://www.robotiksistem.com/servo_motor_types_properties.html

[25] BARNA, Andrej *Zařízení s modelářskými RC servy*[Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií]Brno 2017
https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=159114

[26] *Direct memory access*, Wikimedia Foundation, Inc 7/June/2002 , Web 2019/April
https://en.wikipedia.org/wiki/Direct_memory_access

[27] *FPGA*, Wikimedia Foundation, Inc 10/4/2006 , Web 2019/March
https://sk.wikipedia.org/wiki/Programovate%C4%BEn%C3%BD_logick%C3%BD_obvod

[28] *FPGA*, Wikimedia Foundation, Inc 10/August/2001, Web 2019/Marchhttps://en.wikipedia.org/wiki/Field-programmable_gate_array

[29] *MCU*, Wikimedia Foundation 25/February/2002, Inc, Web 2019/April
https://en.wikipedia.org/wiki/Microcontroller

[30] *MCU*, Wikimedia Foundation 10/4/2006, Web 2019/April
https://sk.wikipedia.org/wiki/Mikrokontrol%C3%A9r

[31] *Logické obvody*, Wikimedia Foundation 10/apríl/2006, Inc, Web 2019/March
https://sk.wikipedia.org/wiki/Programovate%C4%BEn%C3%BD_logick%C3%BD_obvod

[32] *Servo (radio control)*, Wikimedia Foundation 9/November/2011, Inc, Web 2018/November
https://en.wikipedia.org/wiki/Servo_(radio_control)

[33] *Pulse-width modulation*, Wikimedia Foundation, Inc 6/September/2002, Web 2018/November
https://en.wikipedia.org/wiki/Pulse-width_modulation

[34] *Osobný počítač*, Wikimedia Foundation, Inc 3/marec/2005, Web 2019/April
https://sk.wikipedia.org/wiki/Osobn%C3%BD_po%C4%8D%C3%ADta%C4%8D

Seznam symbolů, veličin a zkratek

DMA	Direct memory access, Priamy prístup do pamäte
USB	Universal Serial Bus
COM	Comunication por, toznačenie sériového portu, virtuálneho či fyzického
PWM	Pulse with modulation
GPIO	General purlose input/output, Vstupy a výstupy pre všeobecné použitie
RC	Radio control
RAM	Random access memory
CPU	central processing unit
Soc	System on a chip
PCM	Pulse-code Modulation
BCM	Broadcom pin number