Bakalářská práce

České
vysoké
učení technické
v Praze

F2 Fakulta strojní

Brankový senzor pro stolní fotbal

Jakub Valenta

Vedoucí práce: Ing. Pavel Trnka, Ph.D.
Červen 2019

Poděkování

Děkuji panu Ing. Pavlovi Trnkovi Ph.D. za podporu, trpělivost a cenné připomínky při tvorbě této práce. Také bych rád poděkoval celému ústavu Automatizační techniky za možnost vypracovat tuto práci a za umístění stolu v budově školy.

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a veškeré literární prameny a zdroje informací, které jsem použil, cituji a uvádím v seznamu použité literatury a zdrojů informací.

V Praze 10.5. 2019

Jakub Valenta

Abstrakt

Tato bakalářská práce se zabývá navržením a zkonstruováním systému brankového sensoru v bráně stolního fotbalu. Systém, který je schopen zaznametat polohu procházejícího míčku a jeho rychlost na brankové čáře. Šestnác fototranzistorů a pás IR diod vytvářejí optickou závoru, která zaznamenává přechod míčku přes brankovou čáru. Výsledkem této práce jsou naměřená data.

Klíčová slova: Arduino, interrupt,I2C,SPI, automatizace, bakalářská práce

Vedoucí práce: Ing. Pavel Trnka, Ph.D.

Abstract

This bachelor thesis deals with design and construction table football goal sensor system. System has been found out position and velocity of shotted ball on goal-line. The light barrier has been created by sixteen fototranzistors and IR diods belt. It recognizes ball crossing over goal-line. The result of this thesis are measured data.

Keywords: Arduino, interrupt,I2C,SPI automatization, optimization, Bachelor thesis

Obsah

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Valenta** Jméno: **Jakub** Osobní číslo: **438334**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Strojírenství**

Studijní obor: **Informační a automatizační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Brankový senzor pro stolní fotbal

Název bakalářské práce anglicky:

Goal sensor for table football

Pokyny pro vypracování:

Cílem práce je navrhnout a realizovat zařízení pro testování průchodu střely brankovou čárou ve stolním fotbalu. Zařízení má sloužit jako pomůcka hráčům k vyhodnocení jejich střelby.
1) Navrhněte a realizujte zařízení pro testování průchodu střely brankovou čárou,
které bude plnit následující dílčí úkoly:
a) Zjistit polohu míčku při překročení brankové čáry.
b) Změřit rychlost míčku.
c) Vizualizovat výsledky na displeji.
2) Navrhněte postup archivace výsledků pro účely následného zpracování.

Seznam doporučené literatury:

VODA, Zbyšek. Průvodce světem Arduina. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8
MARTINEK, Radislav. Senzory v průmyslové praxi. 1. vyd. Praha: BEN - technická literatura, 2004. ISBN 978-80-73001-14-8;80-73001-14-4

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Trnka, Ph.D., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.04.2019** Termín odevzdání bakalářské práce: **12.06.2019**

Platnost zadání bakalářské práce:

Ing. Pavel Trnka, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

26.4.2019 Datum převzetí zadání	Podpis studenta

Kapitola 1
Úvod

Úkolem této práce je navrhnout vhodný způsob automatizace měření polohy a rychlosti vystřeleného míče, zobrazení výsledných dat a vhodný způsob jejich archivace. Automatizace stolního fotbalu. Využitím vhodných komponent a součástek - mikrokontroleru, senzorů, diod, tlačítek, displejů, softwaru se trénink této zábavné hry, ve které se spojuje strategie a perfektně natrénované jednotlivé střely, může zautomatizovat. Jednoduchý brankový senzor naměří důležitá data, která se mohou podrobit následné datové analýze pomocí například jazyka Python.

Jaká střela je nejúspěšnější? Jaká střela dosahuje největší rychlosti? Jaká strategie vyhrává zápasy? Jakou techniku střelby se naučit? Jaké výsledky má tento hráč? Toto jsou otázky, na které bych rád našel odpověď. K tomu potřebuji automaticky získávat potřebná data. V této práci změřím data, zjistím, která ze střel je průměrně nejrychlejší a vyhodnotím umístění jednotlivých střel. Veškeré výsledky budou zobrazeny na dvou displejích. Hráč tak bude mít možnost kontrolovat přesnost umístění střely a zjistí i jakou rychlostí přešla přes brankovou čáru.

1.1 Motivace

Několik let organizuji turnaje ve stolním fotbalu a několik let hraji ligu. Používám střelu pinshot a rád bych zjistil, zda je tato střela nejrychlejší. Jakožto student automatizace, bych nové technologie rád přinesl i do světa stolního fotbálku. Chtěl bych vytvořit tréninkový systém, který by mohl hráčům ukázat, jak přesně střílejí. Systém, který by změřil rychlost jejich střel. Systém, který by vyhodnotil jejich data a mohl o určitém hráči zjistit jeho návyky.

1.2 Druhy stolů

Stolní fotbal by jen málokdo označil za sport, ale již dlouhou dobu není jen zabavou v restauračních zařízeních. Pořádají se v něm mistrovství světa, různé světové série a, ač je to k nevíře, tak se pomalu stává uznávaným sportem. V České republice se mu věnuje jen malá komunita lidí. Přesto jsou Češi aktuálně mistry světa v kategorii národů.

Na světě existuje mnoho druhů stolů, na kterých se hrají profesionální turnaje. Němci mají svůj Leonhart, Rakušané Ullrich, Američané Tornado a v Praze se hraje na Rosengart stolu. Stůl je vybaven žlutočernými panáčky se speciální zkosenou hranou pro hru s pogumovanými míčky.

V této práci se zaměřím jen na stůl zkonstruovaný společností Rosengard.Na obrázku 1.1). Na tomto stole se v Praze odehrává i ligová soutěž, která má celkem čtyři úrovně. První liga, druhá liga, třetí liga a divize.

Obrázek 1.1: Stůl Rozendgard

1.3 Strategie

Stolní fotbal není až tolik vzdálen od šachové partie. Když rozebereme klasickou hru dva na dva, tak zde máme dvě hráčské pozice. Útočník a obránce. Útočník, by měl mít co největší procentuální úspěšnost střelby a obránce mít zase co nejvyší úspěšnost na chycené střely. Útočník rozehrává z tyče o pěti hráčích. Jeho úkolem je projít přes soupeřovu pětku a nahrát si přesně na tyč se třemi hráči. Počet úspěšných nahrávek je roven počtu potenciálního pokusu o vstřelení gólu. Úkolem druhého útočníka, je zase snížit procento úspěšných přechodů. Když se útočníkovi podařilo prohodit na tyč se třemi hráči, tak se může připravit ke střele. O tom do jaké části branky bude střela umístěna rozhoduje brankář! Ten se snaží snižovat pravděpodobnost zakrýváním a odkríváním volného prostoru na střelu. Přesné umístění nahrávky a střely je věc tréninku, zvyšování pravděpodobnosti, je věc strategie a čtení soupeřovi hry.

1.4 Ligové utkání

Od září až do června následujícího roku probíhá CFO liga. Jedná se, jak již bylo zmíněno, o čtyř úrovnňovou soutěž, ve které je zaregistrováno přes 500 aktivních hráčů, kteří hrají ve 40 týmech. Ligový tým je složen minimálně ze čtyř hráčů. Tento tým se přihlásí do ligy, kde ho čeká hra, každého s každým a to dvoukolově. Na domácí půdě a na půdě soupeře. Týmy se setkají na předem domluveném hracím místě. Většinou jsou to restaurační zařízení, ve kterých je umístěno více fotbalových stolů. Ligový zápas obsahuje deset malých pod zápasů. Mezi těmito zápasy se můžou střídat nasazení hráči. Střídání během rozehraného zápasu není povoleno. Zápas se hraje na dva vítězné legy do šesti vítězných míčů. Při stavu 1:1 musí výsledkem posledního legu být minimální rozdíl o dva vstřelené góly. Pokud tomu tak není, hraje se do 10 na poslední vítězný gól. Před začátkem zápasu se týmy přivítají, aby vyjádřili respekt k soupeři. Vylosuje se, kdo začne s rozehráváním. Rozehra probíhá z tyče, na které je umístěno pět hráčů. Je třeba se dotknout minimálně třech hráčů a až následně pak může dojít k přímému útoku na soupeřovu branku.Šestkrát se zde hrají dvojice, třikrát jeden na jednoho a poslední je speciální disciplína Two-ball anebo čtyři na čtyři. Two-ball se hraje se dvěma míčky ve hře. Při čtveřicích každý drží jen jednu tyč. Zápasy se hrají na dva vítězné sety do šesti. Zdání klame, ale není nadstandard hrát takovýhle zápas i několik hodin.

1.5 Omotávky

I v tomto sportu existuje několik pomůcek pro lepší efektivitu a přesnost střelby. Hráči používají omotávky navržené na squashové rakety. Madla stolu Rosengart jsou dřevěná. Omotávka chrání před mozoly, pocením ruky a zlepšuje přesnost střelby. Někteří hráči používají i speciální golfové rukavice.

1.6 Druhy střel

Dříve se ve světě stolního fotbalu preferovaly střely bez použití hrany. Hrálo se s plastovým míčkem, který nemá dostatečnou adhezi. V moderním pojetí stolního fotbalu dominují tři varianty vstřelení gólu: Pinshot, Pulshot a Snakeshot. První a poslední zmíněná střela se hrají z hrany. Míček hráč zpracujete na zadní hraně figurky hráče a následně se připraví na střelu. Penshot je střela, která se hraje celým ramenem. Důvodem je dosažení dostatečného momentu. Stahovačka je prudké potažení míče. Hráč se dostane před míč a následně přichází úder zápěstím. Snakeshot je jediná střela u které dochází k rotaci. Dochází zde k pootočení o 359. Jediný regulérní typ protočení. Tyto tři střely porovnám a následně vyhodnotím, která z nich dosahuje největší rychlosti.

Kapitola 2
Koncepce

2.1 Způsob řešení

Během příprav zadání této práce došlo k několika nápadům realizace systému, který by byl schopen snímat rychlost a polohu míče. Realizace měla splňovat základní předpoklady: 1) být levná, 2) být co nejpřesnější, 3) nezasahovat do hrací plochy stolu. Rád bych tento systém využil i při oficiálním zápasu.

2.1.1 Vysokorychlostní kamera

Prvním nápadem bylo využití vysokorychlostní kamery. Konstrukce by se připevnila na okraj stolu a informace o rychlosti by snímala kamera z výšky jednoho metru. Tento koncept má hned několik problémů. Vysoká pořizovací cena kamery, která by byla schopna zvládnou vzorkovací periodu. Zásah do konstrukce stolu a znemožnění snímání během běžného ligového zápasu.

2.1.2 Akcelerometr

Druhým nápadem bylo využití akcelerometru, který by byl umístěn v zadní části brankové konstrukce. Tato varianta je mnohem levnější a díky nárazů bych byl skutečně schopen zjistit rychlost. Bohužel se rychlost nedá změřit přesně kvůli problému rušivého šumu. Z tohoto důvodu nepovažuji tento způsob řešení za přínosný.

2.1.3 Optická závora

Třetí možností řešení bylo vytvoření "optické závory", tj. pole mezi přijímači, řadou fototranzistorů osvěcovaných permanentním infračerveným (dále jen "IR") světlem pomocí vysílačů - IR diod. Při přerušení této optické závory jsme schopni zjistit polohu dané střely. Při jedné řadě senzorů však nejsme schopni vypočítat následnou rychlost. Volbou vhodného řídícího systému je však možné použít daný senzor několikrát pomocí vektoru přerušení. Tato interrrupt funce umožní zjistit několik časů, dvě polohy a následně i rychlost procházejícího míče brankovou čárou. Řešení optickou závorou je nejlevnější, není třeba nijak významě zasahovat do konstrukce stolu a takovýto obvod by mohl být přítomný i během ligového zápasu. Nijak totiž neovlivňuje hřiště. V bráně je navíc stín a nedochází tedy k rušení okolním světlem. Toto je zvolený způsob řešení mého projektu.

2.1.4 Skener

Posledním možností realizace bylo využití skeneru z tiskárny a vytvořit tak senzorické pole. Tento návrh se objevil až v průběhu realizace optické závory

a nebyl dále rozpracován. Do bududoucna jej však nezavrhuji.

2.2 Senzory

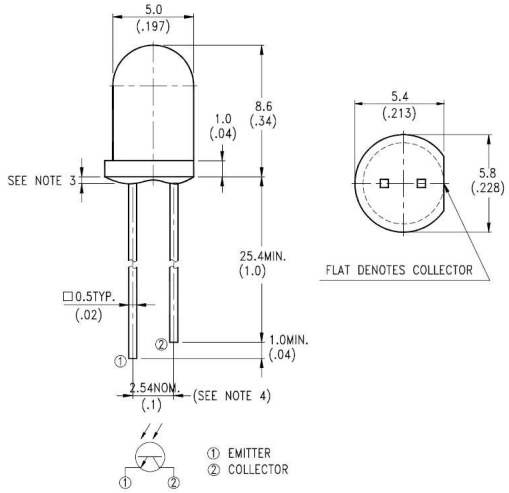
Senzory, ze kterých jsem vybíral ideální komponent, fungují všechny na fotoelektrickém jevu díky zdroji elektromagnetického vlnění světla, ať již ve formě klasického anebo infračerveného. Fotony dopadají na PN přechod, kde narážejí na elektrony, ze kterých se stávají elektrony volné a opouštějí krystalickou mřížku atomu. Na místě elektronu se vytváří díra. Volný elektron se stává nosičem elektrického náboje a snižuje odpor přechodové vrstvy. Pokud nejsou tyto senzory pod zdrojem světla, tak fungují jako pasivní součástky. Intenzitou osvětlení klesá odpor daných součástek.[19]

2.2.1 Fototranzistor

Jedná se o optoelektrickou součástku, která částečně funguje jako fotodioda. Když je osvětlen, vytváří se páry elektron - díra. Je tedy možnost řídit intenzitu dopadajícího elektrického záření.

Obrázek 2.1: Fototranzistor [17]

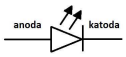
Nepřivádí se proud do báze jako u běžného tranzistoru. Při vzrůstu světelného záření se zvětšuje i světelná citlivost[17]



Obrázek 2.2: Fototranzistor dsh 520[19]

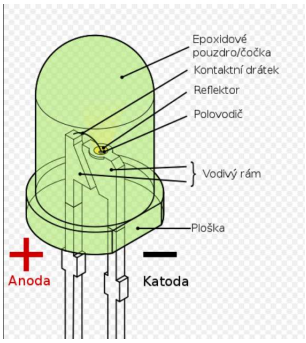
2.2.2 LED dioda

Led dioda je elektrická součástka, která emituje světlo. Funguje na principu PN přechodu. Stejnosměrný proud je veden pouze jedním směrem. Je levná, výkonná a mechanicky odolná.



Obrázek 2.3: Schématické značení LED diody[19]

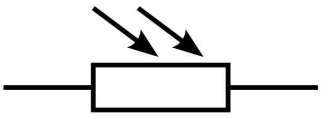
V tomto projektu je využita na emitaci infračerveného záření a společně s fototranzistorem vytváří optickou světelnou závoru. Je třeba zvolit takové parametry emitovaného kužele světla, aby dioda neovlivňovala sousední senzor. Tedy alespoň to jsem si myslel. V realitě toto nerozhoduje o správnosti měření. Zvolená dioda emituje světlo pod úhlem 20. Pro správnou realizaci zkusím zapojit 16 diod. [17][18]



Obrázek 2.4: Schématické značení LED diody[17]

2.2.3 Fotorezistor

Fotorezistor je optoelektrická součástka založená rovněž na fotoelektrickém jevu. Foton předává energii elektronům, aby přešly do vodivostního pásu. Pokud na fotorezistor dopadá více světla, tedy více fotonů, tak se uvolnuje i více elektronů a dochází k větší vodivosti. tato součástka je pasivní. [16][19] [15]



Obrázek 2.5: fotorezistor schema [15]

Obrázek 2.6: fotorezistor [15]

2.2.4 Fotodioda

Fotodioda rovněž funguje na implementaci fotoelektrického jevu. Foton, volný elekron, PN přechod. Fotodiody se používají jako zdroje napětí. Známým

příkladem je napájení kalkulaček. Solárná panel je také složem z fotodiod, slouží tedy k výroě elektrické energie pro domácnosti. Tato účinost je něco mezi 8 až 15 procenty. V tomto projektu, by byla fotodioda použita jako odporová součástka. Při porovnání s ostatnímy optoelektrickými součástkami, je nejdražším řešením. [16]

Obrázek 2.7: fotodioda [16]

2.2.5 Výběr senzoru

Zmíněné součástky fungují na podobném principu. Energie infračerveného světla je menší než energie viditelného světla. Když vytvoříme optickou závoru z IR diod a Fototranzistorů, tak světelně neovlivníme hřiště. Brána je dobře odstíněna a jiné světlo, než to emitované na senzory nedopadá. Pro tento projekt je tedy využito šestnáct fototranzistorů a IR diod, které vytvářejí neviditelnou světelnou závoru.

2.2.6 Shrnutí

Pro tuto práci byl vybrán fototranzistor, který funguje na principu fotoelektrického jevu. Infračervenou diodou zde vytvoříme permanentní zdroj světla. Tyto součástky připájím na speciální shield a vytvořím šestnáctičlenou optickou závoru.

Důvody výběru: Tento typ je vhodný pro následující řešení a zároveň je velmi levný.

Kapitola 3
Řídící systém

Koncept senzoru snímajícího polohu a rychlost je již navrhnut, nyní je třeba vybrat správný řídící systém, který bude kompatibilní, levný a zvládne náročnost projektu. Během vývoje bylo vybíráno z několika mikrokontrolerů. Mezi srovnávané desky byl zahrnut i jeden mikropočítač. Tuto kapitolu cituji z práce: Srovnání desek.[23]

3.1 Úvod do Arduina

První Arduino bylo vyvinuto v italském městě Ivrea v roce 2005. Základní myšlenkou byl vývojový set pro studenty, kteří byli limitováni cenou desek. Arduinu se podařilo přijít na trh s velice levnými mikrokontrolery. Dnes již patří mikrokontroler Arduino mezi známé učební pomůcky využitelné pro výuku studentů ve škole, pro zapojování jednoduchých elektrických obvodů a pro jednoduché programování, stejně jako pro programování vlastnoručně sestavených 3D tiskáren. Z těchto důvodů jsem ho použil pro tvorbu projektu "Snímání, polohy a rychlosti míče v bráně stolního fotbalu". Vzhledem k tomu, že veškeré platformy společnosti Arduino jsou open-source, tak si vlastní desku může doma sestavit každý. Další výhodou je sdílení projektů ostatních tvůrců a volný přístup k samotným programům.[1]

3.2 Arduino desky

V dnešní době jsou Arduino desky velmi rozšířeny. Ve výrobě je mnoho různých druhů desek a každá má své speciální využití. Desky se od sebe liší velikostí, počtem pinů, výkonem, taktovací frekvencí, typem procesoru a převodníkem. Všechny desky mají procesory od fIRmy Atmel. Procesor je obklopen dalšími elektronickými komponenty. Na většině desek je mimo hlavního čipu ještě převodník, díky kterému můžeme komunikovat s počítačem, u některých desek i s chytrým telefonem. Existují i desky, které externí převodník nemají. Je totiž zabudován přímo v čipu procesoru a tak se přibližuje mikropočítači Raspberry Pi. Arduino desky mají k dispozici různé druhy pinů. Analogové, digitální piny. Piny pro komunikaci se sběrnicemi SPI a I2C jsou velmi důležité pro práci s daty a vizualizaci výsledků. SPI pro komunikaci s SD kartou. I2C se používá pro připojení I2C převodníku na displej.[1]

3.2.1 Arduino Mini

Arduino Mini je nejmenší verzí Arduino desek navrženou pro co největší úsporu místa. Naprosto ideální pro aplikaci chytrých vypínačů anebo ovladačů.

Deska však postrádá USB port. Je vhodná pro jednoduché projekty napájené stejnosměrným napětím. (na obrázku 3.1) [1]

Obrázek 3.1: Arduino Mini[1]

3.2.2 Arduino Nano

Z hlediska výkonu se deska Arduino Nano (na obrázku 3.2) neliší od Arduina Mini . Zásadním rozdílem je, že deska Arduino Nano obsahuje USB port a převodník. Tato deska je vhodná pro připojení více zenzorů a následnou kontrolu přes sériovou linku. [1]

Obrázek 3.2: Arduino Nano[1]

3.2.3 Arduno Uno

Nejčastěji využívaný typ desky pro základní aplikace a studijní účely. Ve spojení s nepájivým polem si mohou začátečníci zapojit jednoduché elektrické obvody. Arduino uno (na obrázku 3.3) je hlavním pokračovatelem vývojové linie desek Arduino. Obsahuje sériový port místo USB portu. Další možností je napájení 5-tivoltovým adaptérem, anebo napájení z baterie.

Pro nahrání programu do desky je potřeba externí USB převodník. Z hlediska výkonu je na tom stejně jako většina desek Arduino.Na desce najdeme procesor ATmega328 s taktovací frekvencí 16MHz, Ethernet port. Jedná se o 8-bitový systém.[1]

Obrázek 3.3: Arduino Uno[1]

3.2.4 Arduino Mega

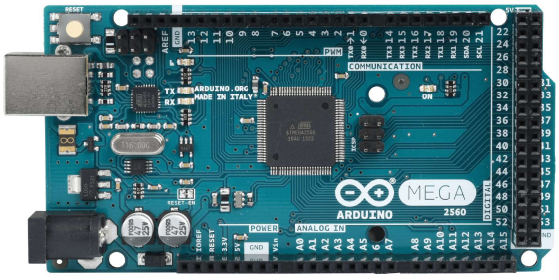
Arduino Mega(na obrázku 3.4) vzniklo prodloužením desky Arduino Uno (na obrázku 3.3). Najdeme zde více pinů a výkonnější čipy. Vlastnosti této desky jsou níže popsány dopodrobna. Dále budeme tuto desku porovnávat s deskou Arduino DUE.(na obrázku 3.5) Desky vypadají velice podobně, ale Mega vlastní, jako všechna doposud představená Arduina, 8-bitový procesor s architekturou AVR.[1]

1. mikropočítače:
   Atmel ATMega2560 256kB flash paměť 8kB RAM paměť 4kB EEPROM paměť

2. Architektura: Atmel AVR

3. Napájení: 5V přes USB, 7-12V adaptér,

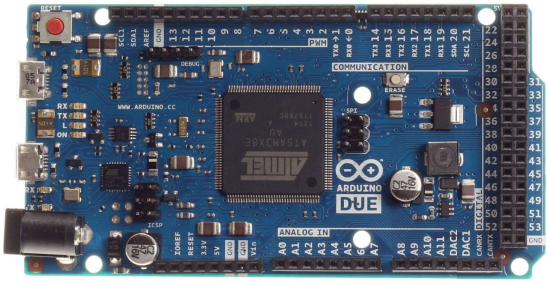
4. Piny I/O piny, 54 digitálních, 4 hardwarové UART, 14 s podporou PWM, I2C sběrnice, 16 vstupních analogových, jen na 6 pinech možnost přerušení programu a tím asynchronní vykonávání obsluhy události. [1]



Obrázek 3.4: Arduino Mega[1]

3.2.5 Arduino Due

První 32-bitové Arduino s procesorem založeným na architektuře SAM. Arduino DUE (na obrázku 3.5) je vývojová deska založená na procesoru Atmel SAM3X8E ARM Cortex-M3. Má 54 digitálních I/O pinů (z nichž 12 je možné použít pro PWM výstupy), 12 analogových vstupů, 4 UART (hardwarové sériové porty), taktovací frekvenci 84 MHz, USB umožňující OTG připojení, 2 DAC (digital to analog) piny, 2x TWI, napájecí konektor, SPI čtečku, JTAG čtečku, resetovací tlačítko a tlačítko pro smazání paměti. Nejdůležitější vlastností pro můj projekt je však možnost přerušení na všech pinech, což na samotné desce Arduino Mega jde jen na 6 speciálních pinech. Paměť 512 KB flash a 96 KB SRAM je přizpůsobena vysoké rychlosti procesoru.[1, 3]



Obrázek 3.5: Arduino Due[1]

Na rozdíl od ostatních desek používá Arduino DUE napájecí napětí 3,3 V, což je i maximální napětí na I/O piny. Vyšší napětí může poškodit vývojovou desku. Dále jsou na desce Arduino DUE dva USB konektory, jeden pro nahrávání programu a komunikaci s PC, druhý jen pro klasické napájení.

Arduino DUE má dostatek výkonu a signálů i pro ten nejnáročnější Arduino projekt. Stačí připojit k počítači, zdroji napětí a můžete začít programovat, tvořit. Arduino DUE je kompatibilní s Arduino shieldy na 3,3 V.[1, 3]

Obrázek 3.6: Arduino Wifi shield [1]

(Na obrázku3.6) se nachází shield s možností posílat data přes Wifi. Jeho využití je ideální v případě, že data chceme posílat přímo na síť.

3.3 ESPRESSIF ESP32-DEVKITC

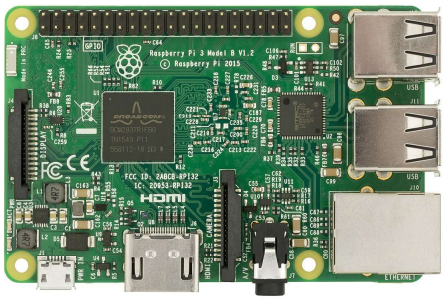
Během tvorby projektu byla vydána nová zajímavá deska, která podporuje Arduino IDE. Deska je velmi výkoná díky dvoum procesorům. Má spoustu

GPIO pinů s podporou SPI, I2C, PWM i interrrupt. Deska je vybavena Wifi modulem, bluetooth pinů. Deska komunikuje při napětí 3,3 V, stejně jako Arduino Due. Deska nepracuje v reálném čase, takže by byl stále zapotřebí RTC modul. Velice zajímavé je propojení této desky s aplikací thingspeak Ta umožňuje zobrazení dat v reálném čase přes knihovnu matlabu. Pro tento projekt tato deska nemá dostatečný počet GPIO pinů s funkcí interrrupt.[22]

Obrázek 3.7: ESP[21]

3.4 Raspberry Pi

Raspberry je jedním ze zástupců skupiny mikropočítačů. Na rozdíl od Arduina, zástupce mikrocontrolerů, vlastní operační systém. Dá se říci, že Raspberry Pi je plnohodnotným počítačem běžícím na Linuxu. Možnosti vstupů pro video/audio, HDMI SDcards a ethernet. Raspberry má daleko méně pinů než Arduino. Jeho vhodné využití je pro projekty zaměřené na video, kamery, komunikaci s několika obrazovkami přes HDMI a vysoké matematické operace. Pro zapojení senzorů, mini LCD displejů a servomotorů je daleko vhodnější Arduino.[4]



Obrázek 3.8: Rasberry Pi[4]

3.5 Shieldy Arduino

Při srovnávání desek Arduina a Raspberry Pi je zřejmé, že Raspberry je nejen daleko výkonnější, ale i funkčně vybavenější deskou. Jedná se o příklad spojení mikrokontroleru (Arduino) a mikropočítačů (Raspberry pi). Připojení k Wifi,

Bluetooth a SD karet Arduino řeší buď připojením samostatných externích modulů, anebo rozšířením desky o takzvaný shield, který má tyto moduly vestavěny. Po připojení shieldu je možné na desku připojit některé další komponenty o dost jednodušeji. Příkladem může být připojení servomotoru jen pomocí čtyř vodičů. Takto se dá stvořit jednoduchý robot. Shieldy jsou kompatibilní s většinou desek Arduino. [1]

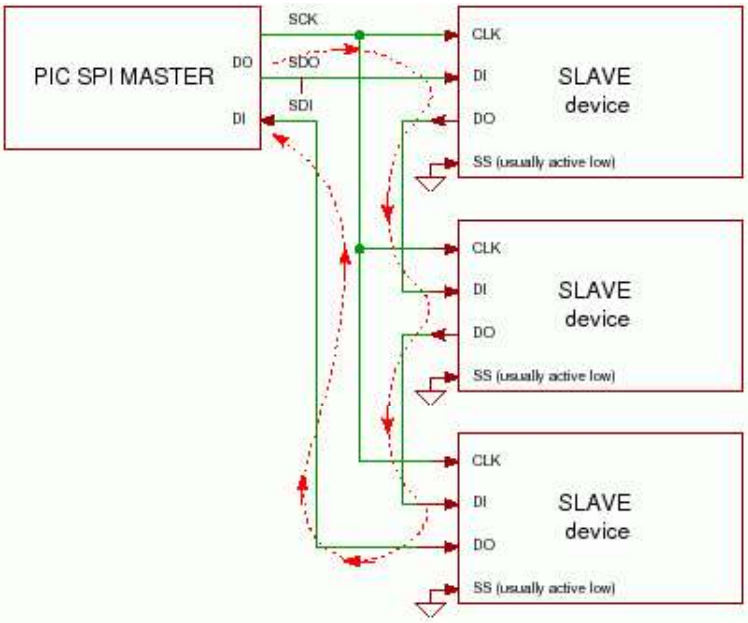
3.6 Výběr vhodné desky

Výběr mikrokontroleru Arduino byl pro aplikaci k tvorbě systému, který dokáže snímat polohu a rychlost míče v bráně stolního fotbálku, primární volbou. Nejedná se o tak složitý projekt, na který by bylo potřeba použít mikropočítač Rasberry Pi. V kapitole Sběrnice I2C a SPI se dozvíte, jak jednoduché je k Arduinu připojit displej a SD kartu. Slouží zde jako vhodná vizualizace a ukládání dat. Problém nastal pří výběru z typů desek Arduino. Výsledný systém bude složen z několika optických závor vytvořených mezi IR diodou a fototranzistorem, dvou displejů, tlačítek a několika diod. Je tedy potřeba deska, která má dostatečný počet pinů pro zapojení veškeré elektroniky. Zbývají nám tedy již jen dvě desky. Arduino Mega a Arduino Due. Rozhodující, je však výpočet rychlosti, který se bude realizovat přes funkci interrupt. Arduino Mega má jen 6 pinů na kterých je možné tuto funkci zavolat, zastavit nahraný program a následně se vrátit zase zpět. Do brány stolního fotbalu je umístěno 16 senzorů a je tedy potřeba 16 pinů s funkcí interrupt. Jedině Arduino Due má možnost tolika pinů. Výhodou, kterou jsem tímto výběrem získal, je několikrát rychlejší deska než je Arduino Mega. Nahrání programu a komunikace s připojeným displejem je více než 5x rychlejší. [2, 3]

3.7 Komunikace přes sběrnice I2C a SPI

3.7.1 Sběrnice SPI

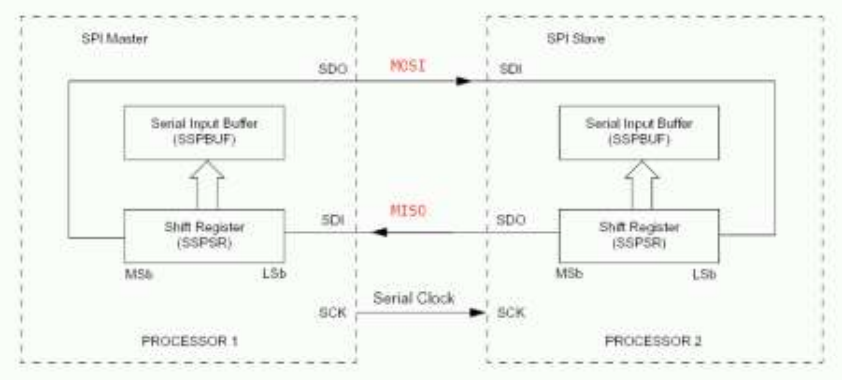
Sběrnice je určena pro připojení vnějších pamětí. Maximální frekvence hodinového signálu je 2 MHz. Stejně jako u I2C používáme Arduino jako master (generátor SCL signálu) Jednotlivé obvody jsou propojeny čtyřmi vodiči. Datový výstup MOSI (Master out, slave in) připojíme na vstup obvodů slave. Výstup SCK se připojen na vstup slave. Každé zařízení typu slave na vstup SS (slave select). Zde je dost patrný rozdíl mezi I2C, která hledá zařízení dle hexadecimální adresy, zatímco u SPI k tomu slouží SS. Přímo určí se kterým zařízením master nyní komunikuje. Pokud je SS v neaktivní úrovni je rozhraní SPI daného obvodu neaktivní. Vstupy SS jsou propojeny přímo s arduinem (masterem). Jelikož se SS připojen na arduíno na určitý pin, je velice snadné vybrat obvod, se kterým budeme komunikovat. Přenosy na sběrnici SPI tedy probíhají mezi masterem a jedním, či více zařízení slave najednou. Arduino (Master) nejprve zapíše do paměti povel pro čtení a adresu dat. Až pak jsou v paměti přečtena data. Sběrnice SPI si tedy můžeme představit jako několik externě propojených registrů. Posun mezi registry se řídí pomocí hodinového signálu. Data jsou vysílána obousměrně. Není potřeba řešit přepínání mezi vysíláním a přijímáním dat. SPI je tedy sběrnicí duplexní. (Na obrázku 3.9) vidíte propojení masteru a tří slave zařízení.



Obrázek 3.9: SPI[8]

Do registru SSPSR je zapsán byte, který byl korektně přijat, ale ještě nebyl zpracován, tj. mikrořadič si jeho obsah nepřečetl. Tento registr tedy slouží jako jednoprvková fronta zabezpečující, že při korektní obsluze nedojde ke ztrátě dat.
Na obrázku 3.10 jsou vidět dva registry – datový záchytný registr Serial Input Buffer – SSPBUF a posuvný registr Shift Register – SSPSR.

Posuvný registr SSPSR slouží současně k vysílání i příjmu jednoho bitu z celé osmice přenášených bitů – každý posun obsahu tohoto registru doprava



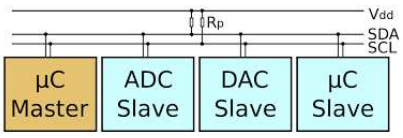
Obrázek 3.10: SSPBUF a SSPSR[8]

znamená, že se vysunutý bit pošle na pin SDO (v případě uzlu typu master se jedná o vodič MOSI) a naopak logická hodnota přečtená na pinu SDI (MISO) je zapsána do nejnižšího bitu posuvného registru. Jeden z uzlů pracuje v režimu master, druhý v režimu slave. Vysílání i příjem jednoho bitu je tedy nedělitelná operace, která vždy proběhne ve stejný okamžik.[7, 8, 9]

3.7.2 Sběrnice I2C

I2C (Inter-Integrated CIRcuit) V určitých ohledech se jedná o sběrnici podobného principu jako SPI. Je také založena na SCL (hodinovém signálu) a jediném uzlu typu master), ovšem některé vlastnosti těchto sběrnic jsou odlišné. I2C je nedílnou součástí každého počítače. Jednouše inplementovaná z hlediska hardwaru a programu. Je vybavena hodinovým signálem s maximální frekvencí 100KHz nebo 400 KHz. Přístup na sběrnici má mechechanismus adresování a arbitraci jednotlivých připojených zařízení. Neboli každé zařízení má vlastní adresu o délce 7 nebo 10 bitů. Slouží k výběru zařízení se kterým chceme komunikovat. Jednotlivá zařízení jsou propojena vodičem SDA (data line) a hodinovým signálem SCL. Zatímco u sběrnice SPI byl umožněn obousměrný přenos dat díky použití dvojice vodičů MISO a MOSI, je sběrnice I2C vybavena pouze jedním datovým vodičem SDA, z čehož vyplývá, že se data přenášejí poloduplexně. Při průběhu přenosu jsou na SDA vysílány jednotlivé datové bity. Platí zde pravidlo logické úrovně. Logická úroveň na SDA se smí měnit pouze s logickou úrovní SCL v úrovni L (Low). Jedinou výjimkou kdy mohou mít obě stejnou hodnotu je start arbitrace a její stop. Pro každou ze specifikovaných frekvencí je určená doba setrvání SCL v úrovni L a H. Každé zařízení připojené na I2C si při arbitraci i při přenosu dat odměřuje vlastním časovačem. Ten měří dobu setrvání H na SCL od okamžiku dosažení této úrovně. Můžeme zde přIRovnat SCL k otevřenému kolektoru. Doba je stejně odměřována i v úrovni L. Díky znalosti tohoto mechanismu můžeme zpomalit přenos některého ze zařízení.

Základ přenosu dat:
Přenosu předchází podmínka start. Dále je vyslána 7-bitová adresa příjemce a 1 bit pro Read/Write. Následně 1 bit ACK na úrovni H. Slouží pro potvrzení přijímání zařízení. Každý Byte je následován 1 bitem ACK. Po ukončení přenosu je vyslána podmínka STOP. Pomocí čtyř vodičů(SDA,SCL,VCC,GND) můžeme na Arduino (zařízení fungující jako Master, viz obrázek 3.11) připojit až 127 zařízení. Na sběrnici včetně masteru může být připojeno až 128 zařízení. Master funguje jako generátor hodinového signálu. Zbylých 127 zařříření funguje jako slave. Jako příklad zde uvedu několik zapojených displejů. Sběrnice je určena pro komunikaci na kratší vzdálenosti. Pro zapojení na delší vzdálenosti je potřeba SDA a SCL odporem připojit VCC. Takto se dá komunikace rozšířit až o několik desítek metrů. [7, 8, 9]



Obrázek 3.11: I2C sběrnice[10]

Na sběrnici není použit výběr zařízení typu slave pomocí zvláštních signálů, protože každému uzlu je přiřazena jednoznačná hexadecimální adresa – kromě elektrických charakteristik je totiž přesně stanoven i komunikační protokol, což je další rozdíl oproti výše popsané sběrnici SPI. Obecně je možné říci, že I2C je sice poněkud složitější, ale za to flexibilnější sběrnice.

3.7.3 Shrnutí

V této kapitole jsem představil jednotlivé typy desek Arduino, které řadíme do kategorie mikrokontrolery. Dále jsem popsal Raspberry Pi, které řadíme do kategorie mikropočítačů. Na jednoduchém zapojení jsem představil sběrnice I2C a SPI. Zapojení displejů a SD karty. Porovnal jsem parametry jednotlivých desek a vybral ideální, pro tento úkol. Arduino deska Due, která má možnost aktivace funkce interrupt na všech digitálních pinech, je více než 5x rychlejší než Arduino Mega a také má mnohem větší paměť.

Kapitola 4
Realizace

Realizace této práce se skládá z několika samostatných témat. Výběr hardwaru, který je schopen splňovat nároky obtížnosti projektu. Volba vhodného softwaru, ve kterém se jednoduše napíše kód pro obsluhu veškerých komponet a logické řízení. Logická část posloupnosti vykonávaných procesů. Jelikož se snažíme najít vhodný vztah mezi polohou, rychlostí a úhlem, tak je zapotřebí využít i kinematické znalosti. Poslední částí realizace je způsob ukládání naměřených dat. Tuto kapitolu cituji z práce: Srovnání desek. [23]

4.1 Hardware

Volba hardwaru je základním pilířem úspěchu projektu. Periferie v tomto projektu nejsou nijak náročné na použití sběrnic, takže postačí prostý microcontroler pro naměření a výpočet dat a jejich základní vizualizaci. Mikropočítač byl proto zavrhnut hned zpočátku.

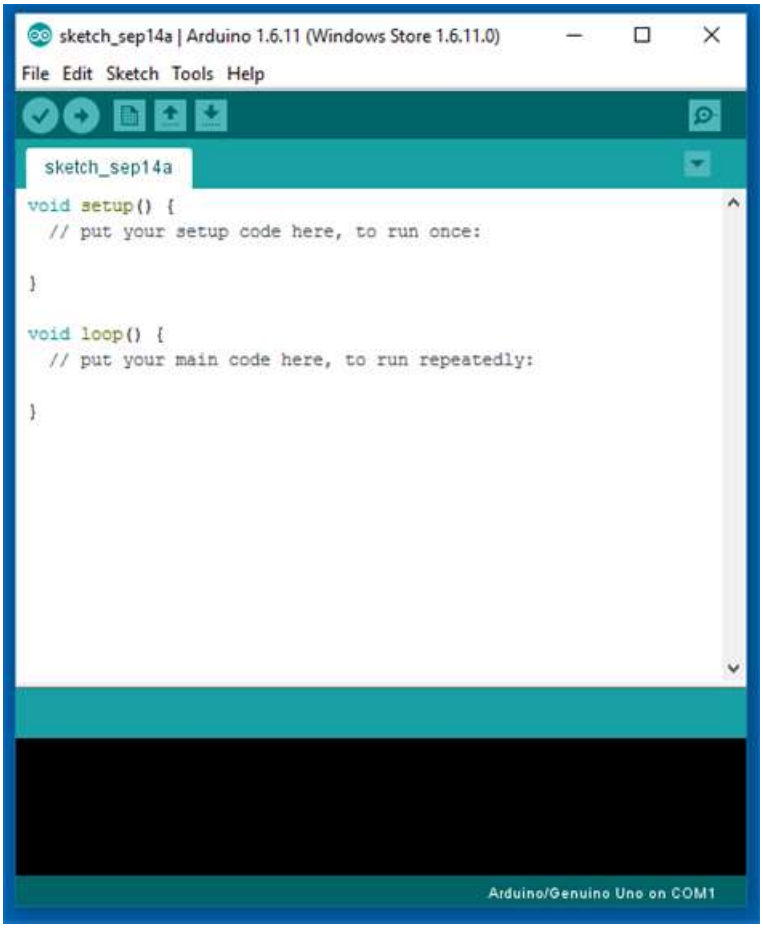
Pro realizaci jsem se rozhodl využít open source Arduino. Výkon jeho čipů od firmy Atmel je naprosto dostačující a veškeré periferie, které jsou potřeba pro realizaci je možné zapojit. Arduino má i základní sběrnice pro vhodné ukládání dat a pro vhodnou vizualizaci na displej. Typů desek Arduina existuje celá řada.

Tento projekt vyžaduje více digitálních pinů s možností přerušení běžícího programu.

Pro každý senzor je potřeba jeden takovýto pin. Jediné Arduino, které má takovýto počet pinů je Arduino Due. Tento microcontroler byl zvolen pro realizaci brankových senzorů.

4.2 Vývojové Prostředí

Arduino se neproslavilo jen díky veškerému výše popsanému hardwaru, pomohlo tomu také IDE (integrated Development EnvIRonment), což je jednoduché vývojové prostředí. Arduino IDE obsahuje mimo jiné textový editor. Několik tlačítek, která slouží pro kontrolu a překlad kódu. Dále je zde nástroj serial monitor, který komunikuje s Arduino deskou. Struktura programu je tvořena v jazyce C. Program se přeloží do strojového kódu a komunikuje s arduinem. Pro nahrání kódu nám tedy stačí Arduino, vhodný kabel, kterým Arduino připojíme k počítači a nainstalované IDE.4.1)



Obrázek 4.1: Arduino Ide

4.3 Základní struktura programu v IDE

Program se dělí na tři části. První slouží pro inicializaci, deklaraci proměnné, volbu pinu, vstupní hodnotu a datový typ. Druhou - funkci setup použiji pro nastavení pinů jako vstup, výstup a sériovou komunikaci. V kapitole 2, Řídící systém, použiji zapojení přes I2C pro základní nastavení parametrů disleje, podsvícení, nastavení kurzoru. Poslední, třetí, částí je nekonečná smyčka loop, která slouží jako ovládání Arduino desky.

```
1  dioda = 10;
```

V první fázi dochází k inicializaci globálních proměnných. Tyto proměnné se následně používají v celém programu. Vytvořili jsme proměnnou dioda, veškeré informace, které proběhnou na pinu 10 jsou uloženy v této proměnné.

```
1  void setup() {
2  pinMode(dioda,OUTPUT);
3  }
```

Funkce setup se vyvolá jen na začátku programu. Nastavíme piny na vstup/-výstup. Spustíme seriovou komunikaci. Zde jsem nastavil proměnnou dioda na výstup.

```
1 void loop() {
2 digitalWrite(dioda,HIGH);
3 }
```

Po vyvolání funkce setup se nastartuje funkce loop, která běží jako nekonečná smyčka. V této funkci vytvoříme kód, který bude následně řídit Arduino desku. Zde jen označujeme pin 10 jako digitální výstup a proměnnou dioda nastavujeme na High, což je logická jednička.

4.3.1 Přerušení běžící smyčky

Jedná se o asynchroní obsluhu události. Přerušení běžící smyčky jen vykoná obsluhu přerušení a následně pokračuje hlavní programu. Každé přerušení má svojí vlastní prioritu. Přerušení lze velmi jednoduše vysvětlit na klasickém počítači. Přerušení vyvolané aktivací tlačítka restart, použití kláves na klávesnici. Jen samotný pohyb myši, který má svojí prioritu. U Arduina lze tato přerušení nastavit na speciálních pinech. Jakákoliv elektrická součástka, připojená na tento pin, může vyvolat přerušení základní smyčky. Přerušení se vyvolá kdykoliv, když dojde k dodržení logiky přerušení. To znamená, v jakémkoliv bodě běžící smyčky. Tato metoda je velmi dobře využitelná při potřebě zachytit všechna data, která mohou být ztracena díky funkcím *milis*() a *delay*(). proměnné, sdílené mezi ISR a hlavním programem, musejí být správně aktualizovány, deklarují se jako volatilní.

U Arduina, toto přerušení lze vyvolat na speciálních digitálních pinech. K tomuto pinu lze připojit vstupní periferii. Vstupní periferie může být například tlačítko. Dobrým příkladem pro představu je tlačítko reset. Jinou vstupní periferií mohou být senzory. V mém případě je to celkem šestnást fototranzistorů. Na běžných arduinech, jako je Uno, či Mega je jen málo pinů, na kterých lze přerušení zavolat. Arduino Due má tuto možnost na všech svých digitálních pinech.

Tuto proceduru vykonává procesor. V tomto projektu je tato metoda použita pro získání časů prvního a posledního překročení brankové čáry. Tento projekt je specifický a je třeba každý ze senzorů ovládat vlastním přerušením. Pro výpočet všech našich parametrů je potřeba alespoň dvou senzorů.

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode); Základní funkce na aktivaci přerušení. Obslužná rutina přerušení je událost, která má v sobě malou sadu instrukcí. Když dojde k externímu přerušení, procesor nejprve provede tento kód, který je přítomen v ISR, a vrátí se zpět do stavu, kde opustil normální provedení. [13] [14]

Arduino umožnuje celkem tři možnosti aktivace funkce interrupt.

1. Rising: přerušení při aktivaci vzestupné hrany.
2. Falling: přerušení při aktivaci sestupné hrany
3. Change: přerušení při změně logické hodnoty.

Příklad na tlačítku bez aretace. Pokud tlačítko je zapojené jako pull-down, tedy jeho výchozí hodnota je logická nula, tak při stisknutí dochází ke změně logiky. Došlo by tedy k aktivaci modu: change a rising. Při puštění tlačítka by došlo k aktivaci: change a falling.

Jednoduchý program, který je napsán pro diodu ovládanou dvěma tlačítky. Na displeji se zobrazují sekundy od spuštění.

```
#include<LiquidCrystal.h>
LiquidCrystal lcd (7,8,9,10,11,12);
volatile int output = LOW;
int i = 0;    n
dioda = 13;
```

Zavolání knihovny pro ovládání displeje. Nastavení volatilní proměnné na logickou hodnotu 0. proměnná i je rovná nule. Je typu int, takže se k této proměnné budou v každém cyklu přičítat další celá čísla. Dioda je připojena na pin 13.

```
void setup()
{
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("start");
  lcd.setCursor(0,1);
  lcd.print("preruseni");
  delay(3000);
  lcd.clear();
  pinMode(dioda,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(2),
  buttonPressed1,RISING);
  attachInterrupt(digitalPinToInterrupt(3),
  buttonPressed2,RISING);
}
```

Ve funkci setup připraví displej. Nastaví pin, ke kterému je připojena dioda jako výstup.

```
void loop()
{
    lcd.clear();
    lcd.print("Pocitadlo:");
    lcd.print(i);
    ++i;
    delay(1000);
    digitalWrite(dioda,output);
}
```

Přičti k proměnné každou sekundu další celé číslo. Pošli logickou hodnotu output na pin, ke kterému je připojená dioda.

```
void buttonPressed1()
{
    output = LOW;
    lcd.setCursor(0,1);
    lcd.print("Interrupt 1");
}

void buttonPressed2()
{
```

```
10      output = HIGH;
11      lcd.setCursor(0,1);
12      lcd.print("Interrupt2");
```

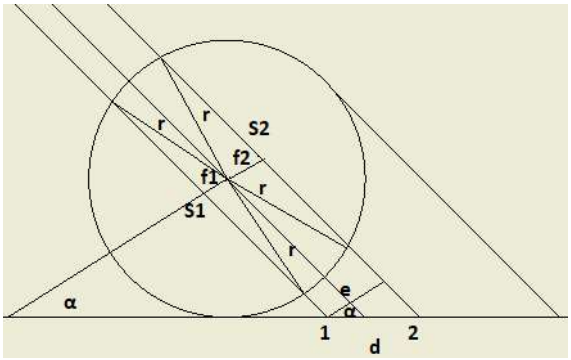
Funkce buttonPressed1 a buttonPressed2 mění logickou hodnotu proměné "výstup". Ovládají tak přímo diodu a na displeji stále běží odpočítávání. [13]

4.4 Kinematická část

Abych mohl vypočítat rychlost potřebuji zjistit celkem dvě polohy, protože mne zajímá za jak dlouho míč urazil vzdálenost AB. Také je možné zjistit celkový čas a ten vztáhnout k velikosti míčku. Nebo je možno stanovit fixní vzdálenot, kterou míč během změřeného času urazil.

4.4.1 Přesnější řešení

Pomocí hardwaru a softwaru jsou docíleny hlavní podmínky k realizaci brankového systému. Připojení všech periferii, senzorů a napsání obslužného softwaru. Pro zjištění rychlosti však potřebujeme mnohem víc. Samotné senzory je schopny registrovat průchod míčku přes brankovou čáru, ale nejsou schopny zjistit měřenou rychlost míče. Pro tuto část bude potřeba přerušit danou smyčku při změně logické hodnoty senzoru a zapnout tzv. přerušení(innterrupt). V přerušení zavolá jiná část kódu a ta se provede. Aby bylo možné zjistit rychlost míče tak je potřeba znát dvě polohy, tj. vzdálenost, na které zjistíme dobu průchodu a následně zjistíme rychlost průletu míče. Problém nastává v tu chvíli, když si uvědomíme, že střela neprochází vždy rovně, ale může mít sklon libovolného úhlu.



Obrázek 4.2: jednotlivé proměnné

Zde (Na obrázku4.2) je graficky zobrazen průlet míčku brankovou čarou.

$$e = d \cdot cos(\alpha) \tag{4.1}$$

$$e = f_1 + f_2 \tag{4.2}$$

$$s_1 = 2 \cdot \sqrt{r^2 - f_1^2} \tag{4.3}$$

$$s_2 = 2 \cdot \sqrt{r^2 - f_2^2} \tag{4.4}$$

$$v = s_1 \Delta t_1 \tag{4.5}$$

$$v = s_2 \Delta t_2 \tag{4.6}$$

$$\Delta t_1 2 = (s_1 - s_2/2 + d \cdot sin()) \cdot v \tag{4.7}$$

Z těchto vzorců, odvodíme vzorec pro výpočet rychlosti a následně úhel, pod kterým míček přešel brankovou čáru. Tento způsob výpočtu nebudu uvažovat v interpretaci programu. Zjištění rychlosti proběhne prvním způsobem.

4.4.2 Čas senzorů

Možnost druhého, jednoduššího řešení jsem otestoval na gravitačním zrychlení.

$$a = g = 9,81[m/s] \tag{4.8}$$

Gravitační zrychlení popsané ve vzorci (4.8) Pokud použiji tuto hodnotu a dosadím ji za zrychlení mohu počítat se vzorcem (4.9).

$$v = g \cdot t \tag{4.9}$$

Vzorec, kterým jsem schopný vypočítat výslednou rychlost. Je třeba však zjistit čas, jak dlouho padá míček z výšky jednoho metru. Ten odvodím ze vzorce (4.10).

$$v = 1/2 \cdot g \cdot t^2 \tag{4.10}$$

Vyjádření času tedy je popsáno v rovnici (4.11).

$$t = \sqrt{2h/g} \tag{4.11}$$

Po dosazení do vzorce (4.11) získám výsledek (4.12)

$$t = \sqrt{2 \cdot 1/9,81} = 0,451[s] \tag{4.12}$$

Mám vypočtený čas, je tedy možné dopočítat i výslednou rychlost, kterou by měl padat míček z jednoho metru. (4.13)

$$v = g \cdot t = 9.81 \cdot 0,451 = 4,429[m/s] \tag{4.13}$$

1	4,05
2	4,1
3	3,9
4	3,95
5	3,98
6	3,89
7	4,2
8	4
9	3,78
10	3,87
11	3,972

Obrázek 4.3: Průměr

Výsledek $4,429[m/s]$ je hodnota, ke které bych se měl přiblížit. Systém tvořený optickou závorou jsem vyndal z brány a připravil ho na testování. Na obrázku4.3 je zobrazena tabulka deseti pokusů. Důvodem nepřesností byla proměnlivá výška. Výsledná chyba je okolo 10 procent. Pro první verzi mé práce zkusím aplikovat tento matematický model.

Pro výpočet rychlosti použiji následujující kód. Budu uvažovat, za dráhu jen průměr balonku. Cekový čas je čas odstínění posledního senzoru mínus čas prvního zastínění prvního seznzoru.

rychlost = prumermicku/celkovyCas)·1000 (4.14)

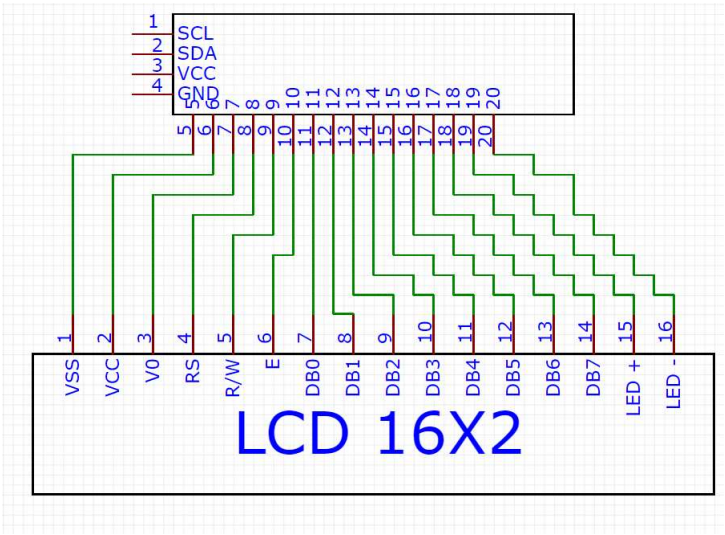
Toto primitivní řešení počítá vždy s rovnou střelou. Pokud tedy míček projde brankovou čarou pod jiným úhlem, tak dojde k nepřesným výsledkům. Pro první verzi je tato chyba zanedbatelná.

4.5 Potřebné komponenty

1) 16x fototranzistor 2) 16x IR Dioda 3) 2x LCD displej 16x2 4) 2x I2C sběrnice 4) ukládací modul SD card (přes SPI sběrnici) 5) RTC - modul pro generování reálného času. 6) tlačítka a diody pro registraci průchodu balonku a restart.

4.5.1 Elektrické zapojení.

Pro tvorbu schémat elektrického zapojení jsem použil program easy eda. Jedná se o jeden z nejvyužívanejších programů pro návrh PCB desek. V softwaru je možné si tuto desku navrhnout a následně si ji nechat na zakázku vyrobit v číně. V tomto softwaru jsem vytvořil veškerá elektrická spojení napojená přímo na čip Arduina. Na obrázku4.4 je zobrazen LCD displej, připojený k I2C převodníku.

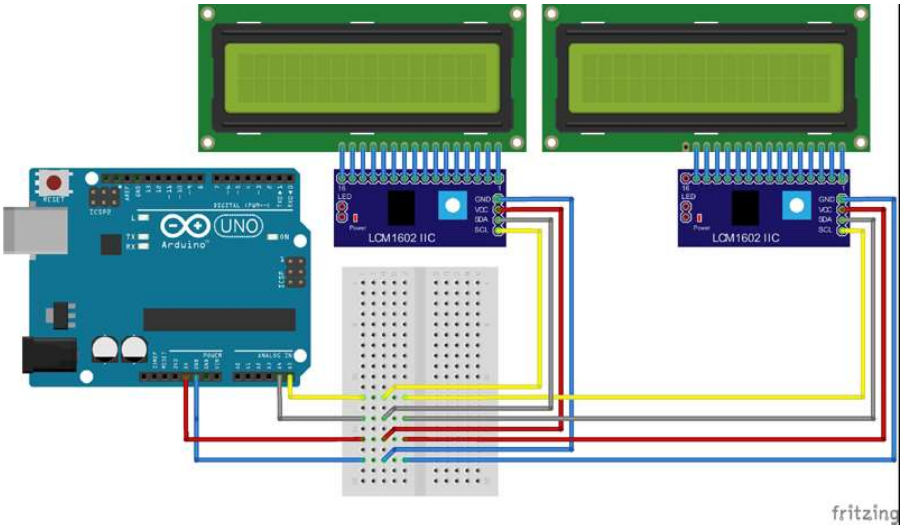


Obrázek 4.4: LCD a I2C

Zbylá zapojení jsou k nalezení v příloze. Jednotlivá zapojení budou zobrazena v programu fritziing. Jedná se o grafický program pro Arduino projekty. Lze v něm projekt skvěle odsimulovat.

4.5.2 Zapojení dvou LCD displejů

Nyní si ukážeme jednoduché zapojení dvou LCD displejů přes výše zmíněnou sběrnici I2C. Potřebujeme 2x displej, 2x I2C převodník, diodu a tlačitko. Každé zařízení (LCD displej připojený přes převodník) má vlastní hexadecimální adresu. Zjistíme ji skenováním I2C převodníků. Nahrajeme tento jednoduchý program do Arduina a zjistíme adresy všech připojených zařízení. Zde v programu můžeme vidět, že jeden displej má adresu 0x3F a druhý 0x3E. Komunikace obou displejů je takto dostatečně zajištěna. První displej nám oznámí stav diody a druhý, zda je splněna podmínka.



Obrázek 4.5: Zapojení 2x LCD displej [9]

```
1 #include <WIRe.h>
2 #include <LiquidCrystal_I2C.h>
```

Aby Arduino mohlo komunikovat s dislayem, tak je třeba nahrát knihovnu WIRe. Pro komunikaci pomocí převodníku přes I2C importuji knihovnu *LiquidCrystal_I2C.h*.

```
1 LiquidCrystal_I2C lcdChar1(0x3F, 16, 2);
2 LiquidCrystal_I2C lcdChar2(0x3E, 16, 2);
```

vytvořím dvě proměnné, každou pro vlastní displej. Tato proměnná potřebuje celkem tři hodnoty. První je hexadecimální adresa převodníku, který je připojený k displeji. Dalšími dvěmi hodnotami jsou počet sloupců a počet řádků.

```
1 const int diodPin = 10;
2 const int tlacitkoPin = 12;
3 boolean Tlacitka = 0;
```

Inicializace proměnných. Dioda je pripojena na pin 10, tlačítko je připojené na pin 12. Nastavení logické nuly pro výchozí polohu tlačítka.

```
1 void setup() {
2 lcdChar1.init();
3 lcdChar1.backlight();
```

```
 4  lcdChar1.setCursor (0, 0);
 5  lcdChar1.setCursor ( 0, 1);
 6  lcdChar1.print("Dioda HIGH");
 7  lcdChar2.init();
 8  lcdChar2.backlight();
 9  lcdChar2.setCursor (0, 0);
10  lcdChar2.setCursor ( 0, 1);
11  lcdChar2.print("Dioda HIGH");
12
13  //set up
14  pinMode(diodaPin,OUTPUT);
15  pinMode(tlacitkoPin,INPUT);
16
17   }
```

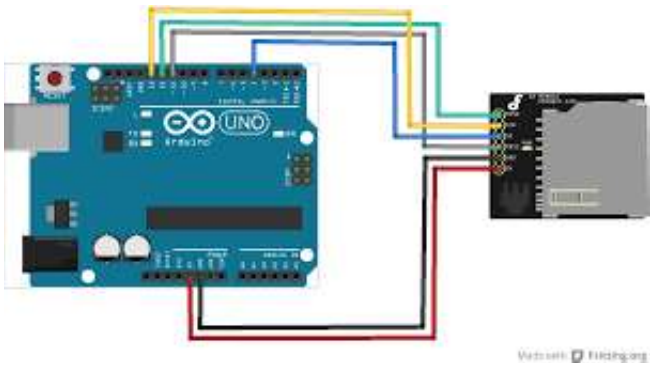
V setup funkci je třeba nastavit oba displeje. Zapnout podsvícení při použití funkce backlight(). Nastavení umístění kurzoru vykonává funkce setCursor. Veškeré číslování je od nuly, poslední znak je tedy (15, 2). Nastavení diody na výstup a tlačítka jako vstup. Při aktivování setup funkce se na displejích vypíše: "Dioda HIGH".

```
1  void loop() {
2  stavTlacitka = digitalRead(tlacitkopin);
3  //Logika
4  if (stavTlacitka == 1) {
5    digitalWrite(diodaPin,HIGH);
6    lcdChar1.print("Dioda sviti");
7    lcdChar2.print("Splneno");
8    }
9    else {
10     digitalWrite(diodaPin, LOW);
11     lcdChar1.print("Dioda nesviti");
12     lcdChar2.print("Nesplneno");
13       }
14 }
```

Hodnotu tlacítka uložím do proměnné stavTlacitka. Pokud má tlačítko logickou hodnotu 1. Zde velmi záleží na zapojení tlačítka. Pull up/ Pull down. Zde je tlačítko zapojeno jako pull up při využití externího odporu. Podmínková část tedy vypisuje při stisknutí tlačítka jeho hodnotu. Stav tlačítka == 1 bude v tomto případě jen tehdy, pokud budu tlačítko držet.

4.5.3 Zapojení SD karty

Na tomto programu si ukážeme jednoduché připojení SD karty k Arduinu přes SPI převodník, které najdeme na obrázku zapojení 4.6. Tento příklad je ilustrační. Měl by nastínit řešení a pozdější aplikaci v Bakalářské práci. Jedná se o vhodný způsob ukládání dat. V tomto programu ukládáme booleovu hodnotu led diody do vytvořeného souboru. Soubor může mít příponu .txt anebo formát .csv. Jedná se o formát, který se dá velice snadno otevřít přímo v Excelu. Je tedy možno hned pracovat s uloženou hodnotou.



Obrázek 4.6: Zapojení modulu SDcard [11]

```
1 #include <SD.h> //Load SD card library\\
2 #include<SPI.h> //Load SPI Library\\
```

Pro komunikaci přes SPI je třeba importovat knihovnu SPI.h. Druhá knihovna je pro práci s SD kartou.

```
1 const int dioda = 11;
2 boolean tlacitko = 12;
3 int hodnotaTlacitka = 0;
4 int chipSelect = 4;
5 File myData;
```

V této části jsem vytvořil proměnnou dioda na pinu 11. Dále jsem vytvořil proměnnou tlacitko, následně nastavil jeho hodnotu na 0. SPI komunikuje s přesně zadaným pinem. chipSelect = 4. Proměnná pro vytvoření souboru se jmenuje myData.

```
1 void setup(){
2 Serial.begin(9600);
3 pinMode(dioda, OUTPUT);
4 pinMode(tlacitko, INPUT);
5 pinMode(4, OUTPUT);
6 SD.begin(4);
7 }
```

Setup funkce obsahuje komunikaci po sériové lince, nastavení vstupů a výstupů. Pin 4, chipselect je nastaven také na výstup. Jako poslední se nastavuje přímá komunikace s SD kartou.

```
1 void loop() {
2 hodnotaTlacitka = digitalRead(tlacitko);
```

```
3  digitalWrite(dioda,hodnotaTlacitka);
4  myData = SD.open("DiodaData.txt", FILE_WRITE);
5  if (myData) {
6  Serial.print("Hodnota diody: ");
7  Serial.print(dioda);
8  Serial.println("");
9  delay(250);
10 myData.print(dioda);
11 myData.close();
12 }
13 }
```

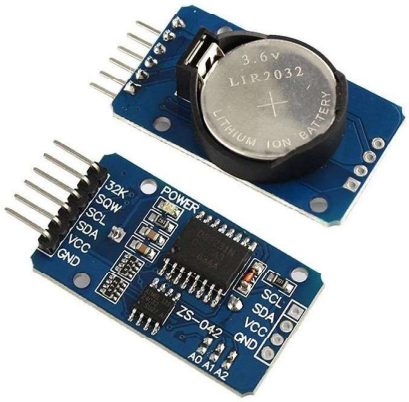
Hodnotu tlačítka ukládám do proměnné hodnotaTlacitka. Podle hodnoty proměnné hodnotaTlacitka určuji stav diody. Vytvořím soubor DiodaData.txt pro zápis a tuto informaci uložím do proměnné myData. Pokud se tento soubor vytvoří, tak na seriálovou linku bude vypsán výše uvedený text. Zpoždění jsem nastavil na 250 ms. Do souboru se ukládá hodnota diody. Soubor je na konci třeba uzavřít metodou close.

4.5.4 Arduino a reálný čas

V této sekci se dozvíte něco o reálném čase a jeho propojení s arduinem. Je důležité upozornit, že Arduino neběží v reálném čase. To znemožňuje zaznamenávat a ukládat informace s přesnou časovou identifikací. Jediný čas, který Arduino řeší, je čas od jeho zapnutí. Pro možnost zaznamenávat soubory v reálném čase je potřeba k desce Arduina připojit externí modul.[4] [5]

4.5.5 RTC

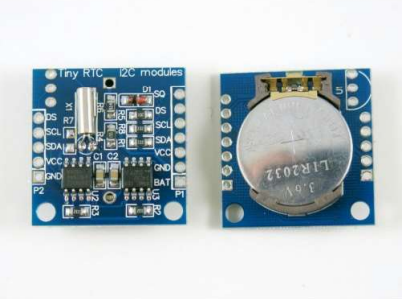
RTC (Real-Time Clock). Způsoby jak počítač získá reálný čas jsou dva. Prvním je získání reálného času z internetu, a to po každém spuštění. Pokud bude počítač vypnut nemá se jak připojit k reálnémů času. Druhým způsobem je obvod RTC hodin s malou baterií. Obvod je schopen bez napájení stále udržovat informaci o reálném čase. Tento způsob uchovávání času se v poslední době používá pro všechna elektrická zařízení, která pracují s časem. Budík, lednička, mikrovlnná trouba, automatické ovladače teploty. [4] [5]



Obrázek 4.7: DS3231[1]

Na obrázku 4.8 je zobrazen modul DS1307 používaný pro propojení s Arduinem. Jedná se o modul jehož funkčnost ovlivňuje teplota okolí. Teplota má špatný dopad na frekvenci oscilátoru. Časový posun je asi 5 minut za měsíc. To je hodinová roční nepřesnost. Aby tento modul mohl uchovávat informace v paměti, je na něm obsažen čip EEPROM paměti.

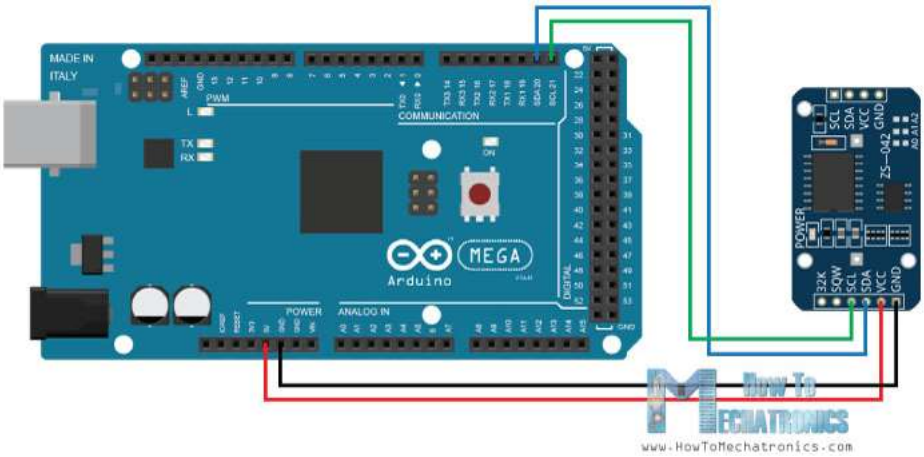
Na obrázku 4.7 je zobrazen druhý modul pro generování reálného času a to DS3231, který je o něco přesnější než jeho starší verze. Tento model má navíc jednoduší zapojení bez potřeby pájení. Oba moduly mají vestavěné pull-up rezistory.[4] [5]



Obrázek 4.8: DS1307[1]

4.5.6 Zapojení RTC

Modul se k Arduinu připojuje za použití čtyř vodičů. Znázorněné schéma zapojení je zobrazené na obrázku... Zprava připojíme zem, napájení, hodinový signál a poslední datový signál. Na obrázku 4.9 je zobrazené připojení k Arduino Mega. Kdybychom připojili tento modul k Arduinu Due, tak musíme napájecí kabel připojit na vstupní napětí 3,3 V.



Obrázek 4.9: Zapojeni RTC[1]

Pro zprovoznění reálného času je potřeba importovat knihovnu s3231.h. I2C propojení zajišťuje RTC (SDA, SCL); v nastavení je třeba inicializovat RTC.begin(). Po této inicializaci můžeme využívat veškeré metody knihovny. Pro demonstraci využijeme tři metody: 1) rtc.SetDow (její parametr nastavuje den v týdnu), 2) rtc.SetTime (má tři parametry: minuty, hodiny a sekundy), 3) rtc.SetTime(den, měsíc, rok). Tyto tři metody níže v programu můžete vidět jako zakomentované. Používají se totiž jen při prvním nahrání programu

do Arduina. V modulu je nastaveno datum 1.1.2000. Čas přenastavím a nahraji do Arduina. Následně zakomentuji příkazy a znovu soubor nahraji do Arduina. Důvodem dvojího nahrávání je uložení zadaného času. Čas je nyní nahrán do modulu. Když nyní Arduino odpojíme od napájení a znovu ho připojíme, tak zjistíme, že čas stále běží.[1] [2] [4] [5]

```
1 #include <DS3231.h>
2 DS3231  RTC(SDA, SCL);
```

Pro práci s RTC modulem využiji knihovnu DS3231. S arduinem komunikuje přes SDA a SCL.

```
1 void setup()
2 {
3 Serial.begin(9600);
4 RTC.begin();
5 RTC.setDOW(WEDNESDAY);
6 RTC.setTime(12, 0, 0);
7 RTC.setDate(28, 3, 2018);
8 }
```

Setup funkce zahájí komunikace po sériové lince na frekvenci 9600 baudů. Zahájení komunikace s RTC, setDOW pro nastavení dne v týdnu. setTime pro nastavení času. setDate pro nastavení přesného data.

```
1 void loop()
2 {
3 Serial.print(RTC.getDOWStr());
4 Serial.print(" ");
5 Serial.print(RTC.getDateStr());
6 Serial.print(" -- ");
7 Serial.println(RTC.getTimeStr());
8 delay (1000);
9 }
```

Ve smyčce dochází k záznamu přesného času na sériovou linku.Tento základní kód použiji pro ukládání dat v přesném čase. Na následujícím obrázku... je zobrazen sériový port zobrazující zápis dne, měsíce, roku a přesného času.

4.5.7 Shrnutí

V této práci jsme se zmínili o několika typech desek Arduino, o potřebných sběrnicích I2C a SPI. Probrali jsem způsoby jak ukládat data přes kabel, wifi, Bluetooth a paměťovou kartu. Největším problémem bylo zjištění, že Arduino nepracuje s reálným časem a pokud chceme ukládat hodnoty v reálném čase, tak potřebujeme připojit externí modul. Spojením tohoto modulu a modulu pro SD kartu se podařilo vytvořit základní šablonu uloženou ve formě csv. Formát, který je možno otevřít v tabulkových programech a následně naměřené hodnoty užívat v grafech. Vytvoření této dokumentace zajišťuje vhodné ukládání a vizualizaci dat. Pro vývoj systému snímání polohy a rychlosti míče, který prolétá brankovou čárou nám tato data pomohou k rychlejšímu zlepšení přesnosti měření.

Kapitola 5

Kompletní zpracování

V předchozích kapitolách jsem vysvětlil na jednoduchých kódech, jak fungují jednotlivé části celého projektu. V této čáti vysvětlím krok po kroku celý kód mé práce. Okomentuji jednotlivé části kódu. Inicializaci, setup, pomocné funkce, loop.

5.1 Inicializace

V první části kódu, je třeba nahrát důležité knihovny a správně inicializovat proměnné.

```
1 #include <WIRe.h>
2 #include <LiquidCrystal_I2C.h>
3 #include <DS3231.h>
4 #include <SPI.h>
5 #include <SD.h>
```

Knihovny, které jsou potřeba pro práci s jednotlivými externími moduly jako jsou SD, DS3231, LiquidCrystal I2C, SPI, I2C.

```
1 #define NUMBER_OF_SENSORS          16
2 #define SENSOR_PINS_DIFFERENCE     2
3 #define SENSOR_STARTING_PIN        22
4 #define TLACITKO_PIN               31
5 #define DIODA_ZLUTA_PIN            33
6 #define DIODA_ZELENA_PIN           35
7 #define PRUMER_MICKU               35.0
8 #define ROZTEC_SENZORU             15.0
```

Definuji počet senzorů, rozteč mezi sousedními senzory, pin, ke kterému je připojen první senzor. Tlačítko připojené na pin 31, Zelená dioda a žlutá dioda. Vzdálenost mezi senzory. Parametr míčku pro následné zjišťování rychlosti.

```
1 LiquidCrystal_I2C lcdRYCHLOST(0x3F, 16, 2);
2 LiquidCrystal_I2C lcdPOLOHA(0x3E, 16, 2);
3 uint8_t testChar[8] = {
4   B01110,
5   B11111,
6   B11111,
7   B11111,
8   B11111,
9   B11111,
```

```
10    B01110
11  };
```

V této části inicializuji displej pro zobrazení rychlosti, displej pro zoprazení polohy. Na displeji pro zobrazení polohy vytvořím speciální symbol, který by měl připomínat míček.

```
1  int chipSelect = 53;
2  File polohaFile;
3  String namePoloha;
4  String dIR;
5  Time  t;
6  DS3231  RTC(SDA, SCL);
7  boolean tlacitkoVal = 0;
8  const int sensorPin = 22;
9  unsigned long myPrichoziCasSpolecny;
10 unsigned long myPrichoziCas[NUMBER_OF_SENSORS];
11 boolean myMicek[NUMBER_OF_SENSORS];
12 unsigned long myOdchoziCasSpolecny;
13 unsigned long myOdchoziCas[NUMBER_OF_SENSORS];
14 boolean gol=0;
15 boolean micekPryc=0;
```

Následuje inicializace všech proměnných, důležitých k průběhu skriptu: Chipselect pro komunikaci s SD kartou, polohaFile jako proměnná, do které se bude zapisovat soubor, proměnná dIR, pro vytvoření nové složky, DS3231 pro komunanikaci s RTC modulem, proměnné pro příchozí čas společný a pro příchozí čas jednotlivých senzorů. Stejně tak i pro odchozí čas společný a odchozí časy jednotlivých senzorů. Logická hodnota gól nastavená na 0 a logická hodnota přítomnosti míčku, také nastavena na 0.

5.2 setup

Funkce setup se zavolá jen jednou a to při zapnutí Arduina. Skládá se z několika částí. Zavolání základních funkcí pro senzory, příprava displejů. věření pamětové karty, zapojení sériové komunikace(pro testování)

```
1  void setup() {
2    setupSensors();
3    resetSensors();
4    Serial.begin(9600);
5    Serial.println("Restart");
6    lcdRYCHLOST.init();
7    lcdRYCHLOST.backlight();
8    lcdRYCHLOST.setCursor (0, 0);
9    lcdRYCHLOST.print("Rychlost micku:");
10   lcdRYCHLOST.setCursor ( 0, 1 );
11   lcdRYCHLOST.print("____________________");
```

Externí funkce jako setupSensors a resetsensors vysvětlím v samostatné kapitole. Začátek komunikace po sériové lince, zobrazení "restart"po sériové lince. Zde sériovou linku používám pro testování jednotlivých komponent. Pokud vše projde korektně, zavolá se funkce loop. Inicializace displeje zobrazujícího rychlost. Aktivace podsvícení a aktivace kurzoru. Na první řádce se zobrazí:

"Rychlost micku:", v druhé řádce displeje se zobrazí ""na každé místo. Takto vypadá displej zobrazující rychlost před prvním výstřelem.

```
1   lcdPOLOHA.init();
2   lcdPOLOHA.backlight();
3   lcdPOLOHA.setCursor (0, 0);
4   lcdPOLOHA.print("Umisteni micku:");
5   lcdPOLOHA.setCursor ( 0, 1 );
6   lcdPOLOHA.print("____________________");
7   lcdPOLOHA.createChar(0, testChar);
```

Stejně tak inicializujeme displej na zobrazení polohy. TestChar je použit na zobrazení speciálnho znaku tam, kde míček projde brankovou čarou.

```
1   pinMode(TLACITKO_PIN,INPUT);
2   pinMode(DIODA_ZLUTA_PIN,OUTPUT);
3   digitalWrite(DIODA_ZLUTA_PIN,LOW);
4   pinMode(DIODA_ZELENA_PIN,OUTPUT);
5   digitalWrite(DIODA_ZELENA_PIN,HIGH);
6   pinMode(chipSelect,OUTPUT);
```

Nastavení diod jako výstupu z Arduina, tlačítko je nastaveno jako vstup. V první fázi je zelená dioda High a žlutá Low. ChipSelekt nastavím také na výstup.

```
1   if (!SD.begin(chipSelect)) {
2     Serial.println("initialization of the SD card failed!");
3     return;
4   }
5   Serial.println("initialization of the SDcard is done.");
6   RTC.begin();
7   t = RTC.getTime();
8   dIR = String(t.year,DEC) + String(t.mon,DEC) + String(t.date,DEC);
9   Serial.println(dIR);
10  namePoloha = dIR + "/" + String(t.hour,DEC) + String(t.min,DEC)
11  + String(t.sec,DEC) + ".csv";
12  Serial.println(namePoloha);
13
14    if (!SD.exists(dIR)){
15       SD.mkdIR(dIR);
16      }
17      polohaFile = SD.open(namePoloha, FILE_WRITE);
```

Nejdříve zjistíme, jetli je karta správně připojena. Pokud ne, tak se na sériový port vypíše: "initialization of the SD card failed!"Pokud je karta správně připojena, nejčastěji bývá problém s prohozením Mosi/Miso, či zapojení chipSelectu na jiný pin než je uveden, tak se na sériovou linku vypíše: "initialization of the SDcard is done.". Spuštění RTC. Funkce getTime ve které jsou uložené veškeré části reálného času. DIR vytvoří název pro složku do která bude me ukládat jméno dle roku, měsíce a dne. Zapíše v dekadické podobě. V předem vytvořené složce se při restartu Arduina vytvoří vždy nový soubor .V proměnné namePoloha se vytvoří jméno souboru, které se bude skládat z hodin,minut, sekund v dekadické podobě. Ověření existence Složky v adresáři. Pokud neexistuje tak vytvořím adresář na kartě. V této

složce vytvořím soubor s názvem namePoloha. Všechny kroky budou vypsány na sériovou linku.

```
    if (polohaFile) {
       Serial.println("uspech");
      polohaFile.println(rovnaSe);
      polohaFile.println(namePoloha);
      polohaFile.println(rovnaSe);
      polohaFile.println("Date: ");
      polohaFile.println("Time: ");
      polohaFile.println("Name of player: ");
      polohaFile.println(rovnaSe);
      polohaFile.println("");
      polohaFile.print("rychlost[m/s];prumer_micku[mm]
      ;casPtotal[us];casOtotal[us]");
      for (int i = 0; i < NUMBER_OF_SENSORS; i++){
        polohaFile.print(";S" + String(i));
      }
      for (int i = 0; i < NUMBER_OF_SENSORS; i++){
        polohaFile.print(";casP" + String(i));
      }
      for (int i = 0; i < NUMBER_OF_SENSORS; i++){
        polohaFile.print(";casO" + String(i));
      }
      polohaFile.println("");
      polohaFile.close();
    }
```

Pokud existuje Polohafile, tak ho otevřu a zapíši do něj následující šablonu. Na sériové lince si nechám zapsat "úspěch". V hlaviče je formulář pro jméno jednotlivých střelců. Do souboru se zapíše jméno sloupců. Rychlost, poloha jednotlivých senzorů. Časy jednotlivých senzorů. Následně se soubor zavře.

```
noInterrupts();
attachInterrupt(digitalPinToInterrupt(22),ISR22,CHANGE);
attachInterrupt(digitalPinToInterrupt(24),ISR24,CHANGE);
attachInterrupt(digitalPinToInterrupt(26),ISR26,CHANGE);
attachInterrupt(digitalPinToInterrupt(28),ISR28,CHANGE);
attachInterrupt(digitalPinToInterrupt(30),ISR30,CHANGE);
attachInterrupt(digitalPinToInterrupt(32),ISR32,CHANGE);
attachInterrupt(digitalPinToInterrupt(34),ISR34,CHANGE);
attachInterrupt(digitalPinToInterrupt(36),ISR36,CHANGE);
attachInterrupt(digitalPinToInterrupt(38),ISR38,CHANGE);
attachInterrupt(digitalPinToInterrupt(40),ISR40,CHANGE);
attachInterrupt(digitalPinToInterrupt(42),ISR42,CHANGE);
attachInterrupt(digitalPinToInterrupt(44),ISR44,CHANGE);
attachInterrupt(digitalPinToInterrupt(46),ISR46,CHANGE);
attachInterrupt(digitalPinToInterrupt(48),ISR48,CHANGE);
attachInterrupt(digitalPinToInterrupt(50),ISR50,CHANGE);
attachInterrupt(digitalPinToInterrupt(52),ISR52,CHANGE);
interrupts();
Serial.print("Serializace dokoncena");
  }
```

Na začátku programu deaktivuji veškeré interrupty, které byly zapnuty v předchozím spuštění. Připravím interrupty na všech pinech vyvoláním funkce unterrupts, která aktivuje všechny interrupty, končí funkce setup. Mód funkce interrupt je nastaven na CHANGE. Reakce na jakoukoliv logickou změnu na daném pinu. Na sériovou linku se vypíše: "Serializace dokončena".

5.3 Loop

Zde popíši chování loop funkce, tedy průběh nekonečné smyčky, která běží na procesoru. V této smyčce dochází k přerušení a zavolání podprogramu. V tomto podprogramu je zaznamenán čas prvního a posledního zastínění senzoru. Po této procedůře se vracíme do hlavní smyčky. Následně jsou tyto informace předány do pomocné funkce rychlost. V poslední řadě dochází k zobrazení výsledků a po zmáčknutí tlačítka k archivaci k restartování logické části programu.

```
1  void loop() {
2  for (int i = 0; i < NUMBER_OF_SENSORS; i++) {
3     lcdPOLOHA.setCursor(i, 1);
4     if (myMicek[i]) {
5        lcdPOLOHA.print((char)0);
6     }
7  }
8  if (gol==1 ){
9  digitalWrite(DIODA_ZELENA_PIN,LOW);
10 digitalWrite(DIODA_ZLUTA_PIN,HIGH);
11 }
```

Proměnná, která ověřuje přechod míče přes brankovou čáru, gól, má hodnotu logické nuly. Skenují se všechny senzory, na displej poloha zobrazím speciální znak, pokud dojde k přerušení optické závory. Logická hodnota diod se změní. Zelená dioda zhasne, rosvítí se dioda oranžová. Do proměnné myMicek[i] je pro každý senzor uložena hodnota logické jedničky při přítomnosti míče, hodnota logické nuly při jeho nepřítomnosti. V této části dochází k aktivaci přerušení.

```
1  zahajeni_testu = micros();
2  micekPryc = 1;
3  for (int i = 0; i < NUMBER_OF_SENSORS; i++) {
4        if (readSensor(i)==0){
5           micekPryc = 0;
6        }
7  }
8  Serial.println("Testovac doba pry : " +
9  String(micros()-zahajeni_testu)+ " us");
10
11
12 if ((gol==1 ) & (micekPryc==1)){
13
14    noInterrupts(); pot eba)
15    lcdRYCHLOST.setCursor ( 0, 1 );
16    lcdRYCHLOST.print("_____"+String(rychlost())+"m/s");
17    Serial.println(rychlost());
```

```
18 }
```

MicekPryc nastavuji na hodnotu logické jedničky. Smyčka forcyklu projde všechny senzory a pokud někdy není míček pryč nastaví ho na logickou 0. Pokud $zahajeni_t estu$ odečteme od doby spustění Arduina, tak se na sériovou linku vypisuje doba průletu míče. Pokud gol == 1 a micekPryc==1, tak deakivujeme funkci interruptů. Již nejsou potřeba. Naopak, nechceme stále zjišťovat změny na senzorech. Nyní je třeba pracovat se zjištěným výsledkem. V tuto chvíli zavolám funkci rychlost(), která vrací hodnotu dopočítané rychlosti v metrech za sekundu. Tuto hodnotu vypíši na displej a na sériovou linku.

```
1  tlacitkoVal = digitalRead(TLACITKO_PIN);
2  Serial.print(gol);
3  Serial.println(micekPryc);
4  Serial.println(tlacitkoVal);
5  if  ((tlacitkoVal) & (gol==1)& (micekPryc==1)) {
6     polohaFile = SD.open(namePoloha, FILE_WRITE);
```

Tlačítko, se zde používá, jako potvrzení pro reset. Jen po aktivaci tlačítka se soubor uloží. Pokud nastane situace ((tlacitkoVal) (gol==1) (micekPryc==1)) tak naměřené hodnoty lze zapsat do souboru. Problém zde může nastat, pokud si senzory stále myslí, že nedošlo k opuštění brankového sektoru.

```
1     if (polohaFile) {
2      polohaFile.print(rychlost());
3      polohaFile.print(PRUMER_MICKU);
4      polohaFile.print(";");
5      polohaFile.print(myPrichoziCasSpolecny);
6      polohaFile.print(";");
7      polohaFile.print(myOdchoziCasSpolecny);
8      polohaFile.print(";");
9      for (int i = 0; i < NUMBER_OF_SENSORS; i++){
10       polohaFile.print(";");
11       polohaFile.print(myMicek[i]);
12     }
13     for (int i = 0; i < NUMBER_OF_SENSORS; i++){
14       polohaFile.print(";");
15       polohaFile.print(myPrichoziCas[i]);
16     }
17     for (int i = 0; i < NUMBER_OF_SENSORS; i++){
18       polohaFile.print(";");
19       polohaFile.print(myOdchoziCas[i]);
20     }
21     polohaFile.println("");
22     polohaFile.close();
23   }
24   else
25   {Serial.println("error opening poloha.csv");}
```

Pokud je vytvořen soubor poloha file, tak zapiš následující informace: rychlost, průměr míčku, odchozí a příchozí čas všech senzorů, polohu všech senzorů a samostatné časy pro příchod i odchod všech senzorů. Pokud se tak nestane, tak vypiš chybu.

```
1     resetSensors();
2     lcdRYCHLOST.setCursor ( 0, 1 );
3     lcdRYCHLOST.print("____________________");
4     lcdPOLOHA.setCursor ( 0, 1 );
5     lcdPOLOHA.print("____________________");
6     digitalWrite(DIODA_ZLUTA_PIN,LOW);
7     digitalWrite(DIODA_ZELENA_PIN,HIGH);
8     gol =0;
9     interrupts();
10  }
11  }
```

Resetuj informace na všech senzorech, resetuj displeje. Logická hodnota oranžové diody je 0 a zelené 1. Hodnota proměnné gol = 0. Znovu se zapne přerušení. V této chvíli může hráč střílet znovu.

5.4 Pomocné funkce

Do pole myMicek ukládáme 1, když micek prošel a 0 když dosud neprošel. Sensory fungují obráceně vrací 0, když míček zrovna je a 1, když míček zrovna neni.

```
1  void backFuncionInterrupt(int cisloSenzoru,int cisloPin) {
2        if (digitalRead(cisloPin)==true){
3            myOdchoziCasSpolecny=micros();
4            myOdchoziCas[cisloSenzoru]=myOdchoziCasSpolecny;
5        }
6        else {
7            myPrichoziCas[cisloSenzoru]=micros();
8            if (gol == 0){
9              myPrichoziCasSpolecny=myPrichoziCas[cisloSenzoru];
10           }
11           myMicek[cisloSenzoru] = 1;
12           gol = 1;
13       }
14 }
15
16 void ISR22() {backFuncionInterrupt(0,22);}
17 void ISR24() {backFuncionInterrupt(1,24);}
18 void ISR26() {backFuncionInterrupt(2,26);}
19 void ISR28() {backFuncionInterrupt(3,28);}
20 void ISR30() {backFuncionInterrupt(4,30);}
21 void ISR32() {backFuncionInterrupt(5,32);}
22 void ISR34() {backFuncionInterrupt(6,34);}
23 void ISR36() {backFuncionInterrupt(7,36);}
24 void ISR38() {backFuncionInterrupt(8,38);}
25 void ISR40() {backFuncionInterrupt(9,40);}
26 void ISR42() {backFuncionInterrupt(10,42);}
27 void ISR44() {backFuncionInterrupt(11,44);}
28 void ISR46() {backFuncionInterrupt(12,46);}
29 void ISR48() {backFuncionInterrupt(13,48);}
30 void ISR50() {backFuncionInterrupt(14,50);}
```

```
31 void ISR52() {backFuncionInterrupt(15,52);}
```

Pro všechny senzory se volá vlastní funkce interrupt, která má parametry pozice senzoru a číslo pinu. Digitalread říká, že pokud je jeho logická hodnota 1, tak již míček na senzoru není. Zde zjistím odchozí čas společný i časy jednotlivých senzorů. Pokud padl gól, tedy proměnná gol == 0, tak zjišťuji čas příchozí, a na všechny senzory, které tuto změnu zaznamenali zapíši logickou jedničku. Změním logickou hodnotu proměnné gol.

```
1  void    setupSensors() {
2      for (int i = 0; i < NUMBER_OF_SENSORS; i++) {
3           pinMode(SENSOR_STARTING_PIN + i
4           * SENSOR_PINS_DIFFERENCE, INPUT);
5      }
6  }
7
8  void resetSensors(){
9       for (int i = 0; i < NUMBER_OF_SENSORS; i++){
10         myPrichoziCas[i]=~0;
11         myOdchoziCas[i]=0;
12         myMicek[i]=0;
13
14      }
15 }
16
17 int readSensor(int ind) {
18   int val = digitalRead(SENSOR_STARTING_PIN
19   + ind * SENSOR_PINS_DIFFERENCE);
20   return val;
21 }
22
23 float rychlost(){
24     float celkovyCas;
25     float rychlost;
26     pomocna =3600/1000000;
27     rychlost =  (PRUMER_MICKU/celkovyCas)*1000;
28     Serial.print(rychlost);
29     return rychlost;
30 }
```

Nastavení všech senzorů zajišťuje funkce setupSensors(). Funkce resetSensors() zresetuje veškeré hodnoty na senzorech, časy a logiku polohy. Funkce vytvořená na čtení ze všech senzorů najednou je readSensor. Poslední funkcí, je vypočtená rychlost.

5.5 Archivace výsledků

Archivace výsledků probíhá jako zápis csv souborů na SD kartu. Z Arduina se na kartu zapisují hodnoty rychlosti, logiky jednotlivých senzorů na zápis polohy. Na obrázku(). Senzory zaznamenávají společný čas příchozí, společný čas odchozí. Čas příchodu a odchodu je zaznamenán i na jednotlivých senzorech. Tyto časy později poslouží k přesnému vytvarování předmětu, který prošel přes optickou závoru. Tato modelace bude jen ve 2D souřadnicích.

Při použití této vizualizace ověřím správnost použití vzorce na výpočet úhlu. Míč se při průchodu rovnou střelou, zobrazí jako kružnice. Při průchodu pod úhlem se bude zobrazovat elipsa. Tyto garfické vizualizace zobrazím pomocí programovacího jazyka Python, který je velmi vhodný nástroj pro datovou analýzu. Jazyk má obrovskou uživatelskou základnu. Přímo pro datovou analýzu jsou předpřipraveny knihovny Pandas, Seaborn, Matplotlib, Numpy.

rychlost[m/s]	S0	S1	S2	S3	S4	S5	S6	S7
5.1335.00	0	0	0	0	0	0	0	0
4.5135.00	0	0	0	0	0	0	0	0
4.6635.00	0	0	0	0	0	0	0	0
5.6235.00	0	0	0	0	0	0	0	0
5.4335.00	0	0	0	0	0	0	0	0
3.5435.00	0	1	1	1	0	0	0	0
5.7635.00	0	0	0	0	0	0	0	0
4.3735.00	0	0	0	0	1	1	1	0
5.7735.00	0	0	0	0	0	0	0	0
5.5735.00	0	0	0	0	0	0	0	0
5.0035.00	0	0	0	0	0	0	0	0
6.2935.00	0	0	0	0	0	0	0	0
4.6035.00	0	0	0	0	0	0	1	1
5.4535.00	0	0	0	0	0	0	0	0
6.7735.00	0	0	0	0	0	0	1	1
5.1635.00	0	0	0	0	0	1	1	1
7.8835.00	0	0	0	0	0	0	0	0
6.3435.00	0	0	0	0	0	0	0	0
5.6535.00	0	0	0	0	0	1	1	1
6.5835.00	0	0	0	0	0	1	1	1
6.3135.00	0	0	0	0	0	0	0	0
6.4535.00	0	0	0	0	0	0	0	0
5.7635.00	0	0	0	0	0	0	1	1
6.2735.00	0	0	0	0	0	0	0	0
5.5535.00	0	0	0	0	0	0	0	0
5.3435.00	0	0	0	0	0	0	0	0
5.6135.00	0	0	0	0	0	0	0	0
5.8335.00	0	0	0	0	0	0	0	0
6.1235.00	0	0	0	0	0	1	1	1

Obrázek 5.1: Archivace výsledků

Python je jeden z nejpoužívanějších programovacích jazyků dnešní doby. Je to dynamický jazyk, objektově orientovaný, dá se však použít i procedurálně. Je vhodný na tvorbu webových aplikací, webscraping, datovou analýzu. Je to nejpoužívanější jazyk ve světě strojového učení. V dnešní době je to nejlepší jazyk pro vývoj, díky skvělé čitelnosti kódu a jeho rychlému napsání. Python nevyužívá preprocesoru. Jedná se jen o byte kód, použitelný na jakémkoliv operačním systému. Je velmi pomalý, což ho pro využití ve všech případech velmi limituje. Společnost Nvidia na tento jazyk vsadila a vytvořila API přímo do svého grafického rozhraní CUDA. Tato knihovna se jmenuje Numba.

Numba je open source Python kompilátor, který obsahuje nástroje pro kompilaci just-in-time pro cíle CPU i GPU. Nejenže kompiluje funkce Pythonu pro provádění na CPU, ale obsahuje i Python-nativní API pro programování grafických karet NVIDIA přes ovladač CUDA.[20]

Tímto kompilátorem společnost Nvidia zvětšila tržní hodnotu superpočítačů jako je Jetson.

V Pythonu je velmi jednoduché vytvořit objekt, je zde totiž objektem úplně vše.

```
1  a = 5
```

Reference na objekt a, který je typu int, má hodnotu 5. Typ a hodnota tohoto objektu je volně přepisovatelná.

Importování balíčku probádá, jako ve většině programovacích jazyků, pomocí příkazu Import.

```
1 Import Pandas
2 data=pd.read_csv("/home/jakub/Downloads/Stahovacka.csv")
```

Takto jednoduše se do Pythonu načte csv subor pro čtení dat. Pro následovnou datovou anylýzu použiji speciální rozhraní jupyter notebook. Jedná se o upravené GUI IPython knihovny, která funguje jako slovník vstupů a výstupů. Je tedy možné zavolat jakýkoliv vtup či výstup. Velká výhoda Jupyter notebooku oproti běžnému kompilátoru, je možnost psát kód po jednotlivých řádcích a hned si nechat vypisovat výsledky. Do tohoto souboru je možné vkládat poznámky, obrázky a přidávat videa z youtube. Soubor se dá vygenerovat v Latexu, pdf či html formě.

5.6 Shrnutí

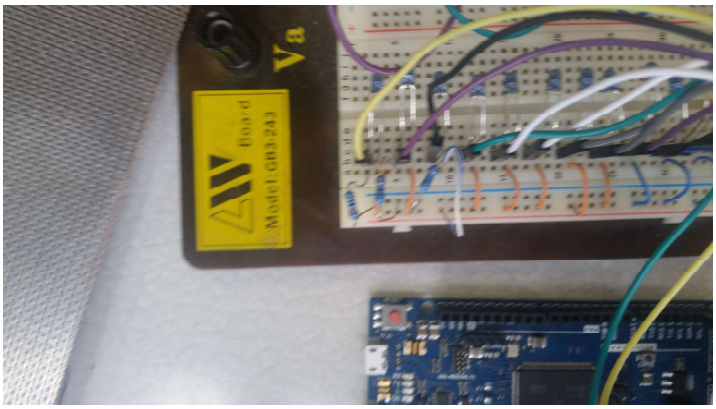
V této kapitole jsem popsal jak funguje kód. Importoval jsem důležité knihovny, připravil globální proměnné, vytvořil funkci setup, která se zavolá jen při spuštění programu. Podrobně jsem popsal logiku probíhající smyčky. Po inicializaci programu je tento stav: Na displeji poloha je připraveno pole šestnácti senzorů. Displej pro zobrazení rychlosti je také připraven. Hodnoty všech senzorů jsou na logické nule. Svítí zelená dioda a oranžová je zhasnutá. Po průchodu míčku, se logika diod změní, na displejích se zobrazí informace a pokud není nějaký ze senzorů stále zastíněn, jsou deaktivovány interrupty. Po zmáčknutí tlačítka se resetují senzory, znovu se změní logika diod, displeje zobrazují to stejné, co na začátku a funkce interrupt je znovu aktivní. Systém je připraven na novou střelu, data jsou uložena na SD kartu do csv souboru s přesně vygenerovaným jménem. Pro následující zpracování byl navržen jazyk Python, jednoduchý pro datovou analýzu. V příloze naleznete datové analýzy jednotlivých střel zpracované v Jupyter notebooku.

Kapitola 6

Průběh výroby zařízení

6.1 První fáze

V první fázi jsem se musel seznámit se samotnými deskami Arduino. Zjistit, jak takový microcontroler funguje a naučit se programovat na základních projektech. Naučil jsem se zapojit displej přes všech jeho šestnáct pinů. Následně jsem se dozvěděl o sběrnicích a pořídil si I2C převodník. Zde jsem se zastavil minimálně na týden, protože jsem nechápal, jak je možné mít na dvou displejích stejný výstup. Problém byl ve stejné hexadecimální adrese, i když výrobce uváděl adresu jinou. Díky skeneru I2C adres jsem se naučil automaticky zjišťovat, jakou adresu mají zapojená zařízení. Bylo třeba pájkou upravit adresu jednoho z převodníků.



Obrázek 6.1: Zapojení senzorů na breadboardu

Pro připojení jednotlivých senzorů jsem použil breadboard tzv. nepájivé pole (obrázek 6.1). Jednotlivé součástky jsem postupně zapojoval a zkoušel jsem odezvu ze softwaru. Prvním problémem byla část programovací. Následně chybné zapojení a častokrát se stalo, že jsem měl nefunkční diodu, tlačítko, či fototranzistor. Takto jsem se naučil proměřit jednotlivé součástky, což je velmi důležité pro tvorbu takovýchto projektů. Na konci této fáze jsem se pokusil přidělat senzory ke stolu. Senzory byly stále na nepájivém poli. V této části nebylo využito přerušení a polohu jsem byl schopný zjistit jen v určitých okamžicích, kdy se nevykonávala jiná část programu. Rychlé střely takto nebylo možné zachytit.

Na následujícím obrázku 6.2 je zdokumentovaná první chvíle, kdy se podařilo přemostit I2C převodník a vytvořit druhou hexadecimální adresu. Jsou zde zobrazeny výstupy, které se zobrazují i ve finální fázi.

Obrázek 6.2: zobrazení na displejích

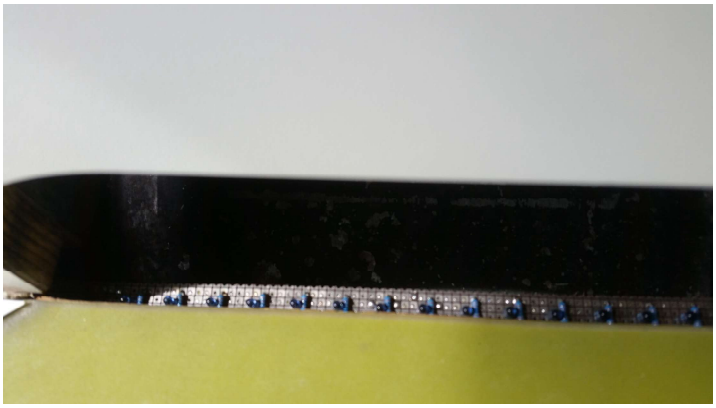
6.2 Druhá fáze

Použití interruptů pro zjištění polohy bylo nejdůležitější částí této fáze. Podařilo se zobrazit polohu i rychlejších střel, takže hardwarově bylo vše připraveno. Rychlost těchto dat již záležela jen na softwareovém vyhodnocení. V této fázi jsem začal řešit archivaci dat. Připojil jsem SD Card modul. Podařilo se mi vytvořit soubor, do kterého se zapsala poloha střely. Problém csv a stringu v Arduinu jsem řešil kvůli následnému pracování s výsledky. Proběhly první testy a bylo možné začít připravovat defaultní přidělání systému ke stolu. V této fázi jsem začal pájet a vytvářet jednotlivé shieldy pro ideální zapojení k Arduino Due desce.

Obrázek 6.3: Pás IR diod

Zde na obrázku 6.3 je zobrazen pás IR diod, který je přidělán v bráně oboustranou lepenkou. Jak můžeme vidět na fotografii, je výplň branky z kovu, takže zde docházelo ke zkratům. Bylo třeba pás přidělat několika vrstvami oboustraných pásek. Při připevnění druhého shieldu se seznorickou částí, nastala největší komplikace s neprůchozím míčkem. Bylo sice možné střelit míček do brány, ale míč nepropadl a zůstal uvnitř brány. Při každé střele došlo k odražení míče a nárazu do senzorové řady. Pomocí programu Inventor a technologie 3D tisku bylo třeba vytvořit ochraný shield.

Bohužel jsem zjistil, že tato ochraná pomůcka zastinuje senzory, takže měření nefungovalo správně. Nakonec byl tento problém vyřešen posunutím



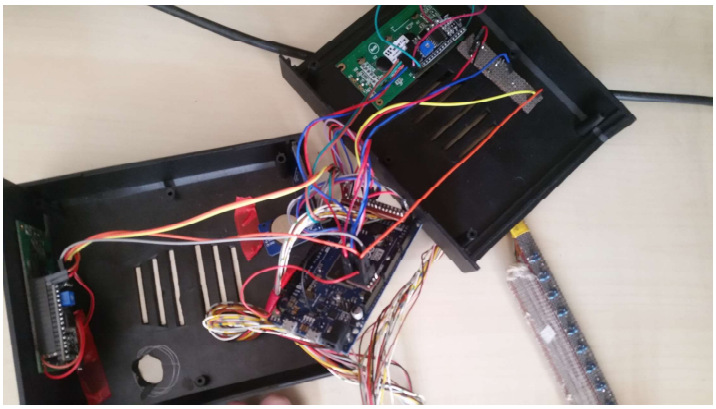
Obrázek 6.4: Pás fototranzistorů na brankové čáře

senzorů co nejvíce k brankové čáře. Diody byly natočeny pod úhlem 30 Po této úpravě se stala branka prostupnou.

K poslední části řešení bylo třeba připájet další shieldy a mít tak veškeré součásti lehce připojitelné. V průběhu pájení nastalo několik dost nepříjemných problémů, které oddálily důležité testování systému. Nejčastějším problémem bylo propojení vodičů a vyvolání zkratu. Jelikož se simultálně opravoval i software, bylo velmi problematické dostat se do fáze, kdy je vše správně zapojeno, připraveno k pájení a v programu má správně pojmenovanou proměnnou, připojenou ke správnému pinu.

6.3 Testovací fáze

Testovací fázi, zahájil test přesnosti měření rychlosti. Z výšky jednoho metru jsem pustil míč volným pádem. Na displeji se zobrazovaly hodnoty v mikrosekundách na milimetr. Převod jednotek na metry za sekundu zobrazil první výsledky v dobře porovnatelném formátu. Průměrnou hodnotu z těchto měření jsem porovnal s ideální vypočtenou hodnotou. Výsledná odchylka byla 10 procent. Zařízení jsem znovu nainstaloval do konstrukce branky.



Obrázek 6.5: Příprava na sestavení

Při instalaci nastal problém přetočení senzorů. Na displeji se výsledky zobrazovali na opačné straně. Pro lepší instalaci by bylo vhodné napsat

krátký program na ověření přítomnosti senzorů. Obecně je třeba pro zařízení napsat několik ověřovacích programů. Díky těmto programům je možné velmi efektivně zjistit, kde dochází k problému (obrázek 6.5). Jedním z těchto problémů byl např. přerušený kabel, náhodné odpojení. Proto bude lepší si takovéto problémy vypisovat na displej.

Obrázek 6.6: Uzavřená krabička

Další situací, která se musela řešit, po nainstalování systému, bylo lepidlo. Shield diod připojený do sítě, generuje spoustu tepla. Teplo má špatný dopad na adhezi lepidla, takže dochází k odlepení součásti a její možné poškození. Po svolení vlastníka stolu bylo možné tyto diody přivrtat. Nedošlo tak k žádné úpravě, která by omezila použití systému při zápase. Testování systému proběhlo úspěšně. Získal jsem validní naměřená data.

6.4 shrnutí

Samotná konstrukce systému byla časově velmi náročná. Naučení se práce s Arduinem, pájení, hledání zkratů a chyb v kódu. Dva roky jsem se věnoval tomuto projektu a jsem rád, že se podařilo vyvinout funkční systém. Práce s naměřenými výsledky je důležitou součástí datové analýzy.

Kapitola 7
Vyhodnocení výsledků

Systém byl testován na vzorku 100 střel rozdělených do třech datasetů. Každý z datasetů označuje jednu ze tří nejúžívanějších střel stolního fotbalu. Jedná se o střely: Pinshot, Snakeshot a Pullshot. Z těchto střel je údajně nejrychlejší střela Pinshot. Na vzorcích provedeme základní datovou analýzu a zjistíme pravdivost tohoto tvrzení. V první fázi vyčistím data od chybových a nepřesných výsledků. Za chybové výsledky označuji ty, které se načetly při odlepení diodové lišty a následném upevňování. V této kapitole budou zobrazena již data vyčištěná a jejich analýza pomocí Excelu a následně pomocí programovacího jazyka Python

Prvního testování systému se zůčastnili celkem tři hráči stolního fotbalu. Dva aktivně hrající druhou ligu a jeden hrající ligu třetí. Každý se pokusil o 15 střel každého typu. Nejedná se tedy o data jen od jednoho hráče. Hledáme obecně nejrychlejší střelu, rozdílné schopnosti každého hráče minimalizují ovlivnění celkových výsledků.

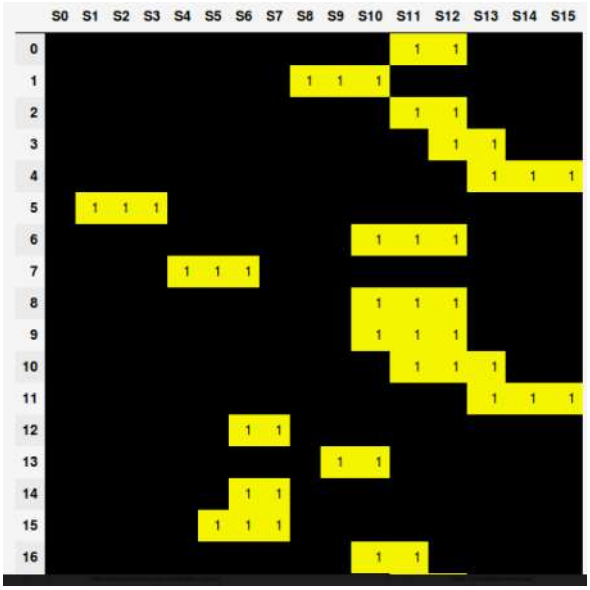
7.1 Vyčištění dat

Čistá data jsou nejdůležitější součástí každého vyhodnocení. V mém případě se musím potýkat s prostým faktem, že pokud zahřejeme lepidlo, tak jeho přilnavost klesá. K mechanickým úpravám stolu nejsem oprávněn, je tedy třeba odstranit data neukazující pravdivý průlet míče, ale pohyb diodového systému. Čištění dat ukáži na programovacím jazyku Python.

7.2 Python čištění dat

Objektově orientovaný programovací jazyk Python je ideálním nátrojem pro datové analytiky, strojové učení, tvorbu webových aplikací. Jeho velkou výhodou je uživatelská základna, která vyvinula nespočet knihoven. Knihovna pro datovou analýzu se jmenuje Pandas.

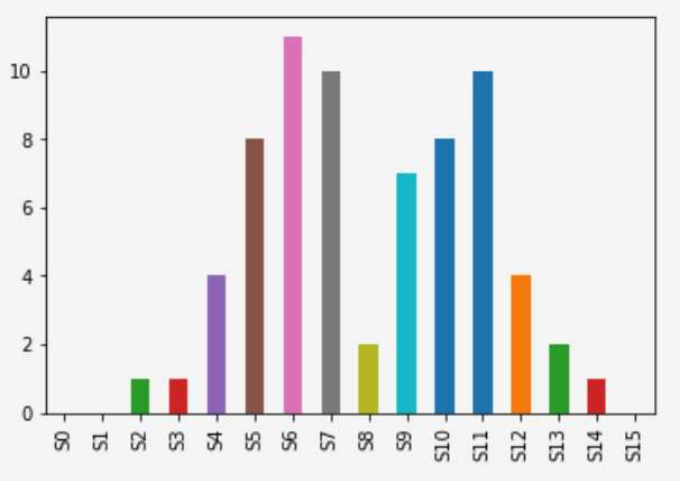
```
1 import pandas as pd
2 import numpy as np
3
4
5 data=pd.read_csv("Stahovacka.csv")
6 data_S = data.filter(regex='^S.*')
7 bin_filter = data_S.sum(axis=1) <= 3
8 data_S_fil = data_S[bin_filter]
9 data_S_fil.style.apply([lambda x: color])
```



Obrázek 7.1: Data

Na prvních dvou řádcích importuji knihovny důležité pro datovou analýzu, a to Pandas a Numpy. Tyto knihovny jsou napsány v jazyce C a Fortran. Na řádku 5 nahrávám soubor do proměnné Data. Řádek 6 se omezuje jen na sloupce začínající S, tedy na vechny zaznamenané polohy senzorů. Objekt bin_filter vymazává všechny řádky, na kterých je více než 3x zastíněný senzor. V poslední části používám anonymní funkci Lambda pro zlepšení vizualizace výsledků. Pole, kde budou nuly, budou černá a pole, která zaznamenají průchod míče, budou mít žlutou barvu.

```
1  total = data_S_fil.sum()
2  total.rename('total', inplace=True)
3  total.plot.bar()
```

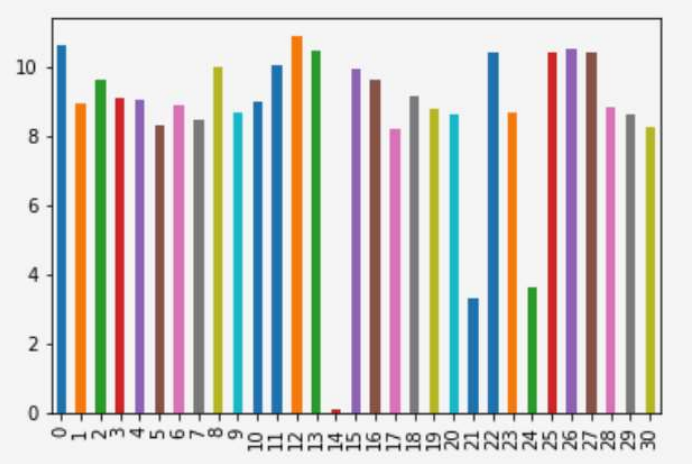


Obrázek 7.2: Zastínění senzorů

Objekt Total je suma všech jedniček jednoho senzoru. Pomocí metody interplace tento řádek vložím do dat, která jsem si načetl. Reálná data nijak nezměním. Tento řádek je velmi důležitý pro grafické zobracení polohy 4.1. V posledním řádku vytvoříme bar graf.

Takto zobrazená data zobrazují kam střelci střílí. Ve stolním fotbale má většina hráčů střely daleko rychlejší blíže k sobě, než-li do protipohybu. Velmi efektivní je přesná střela k tyčce. Tedy zastínení posledních dvou senzorů. Pokud bych takto analyzoval skutečná data ze zápasu, tak bych mohl najít slabinu a návyk hráčů. Pokud totiž hráč není schopen dostřelit až úplně ke kraji, tak je následná obrana daleko jednodušší.

```
1 rychlost = data['rychlost[m/s]'].apply(lambda x: x[:-3])
2 rychlost.astype(float).plot.bar()
```



Obrázek 7.3: Zastínění senzorů

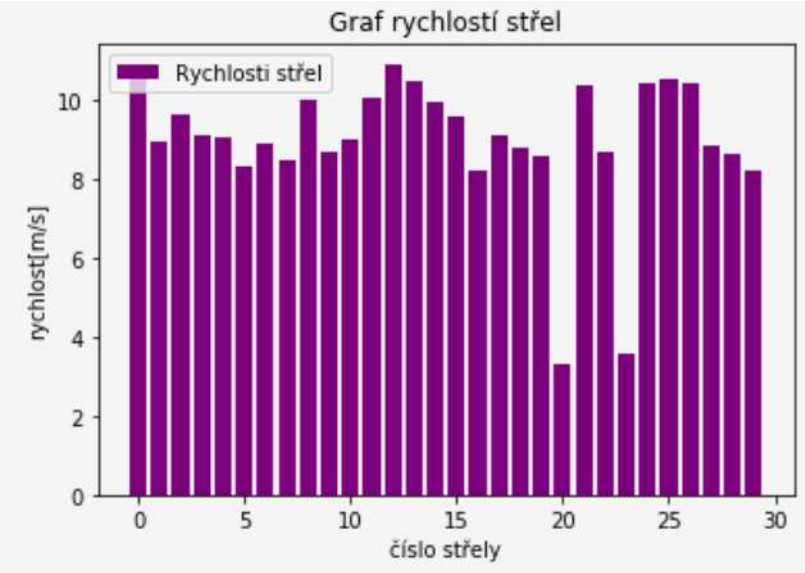
Na tomto grafu jsou zobrazeny rychlosti jednotlivých střel. Je zde vidět i chyba, která by měla být vymazána ze systému. Z tohoto grafického zobrazení jsem schopen zjistit, nejrychlejší střely. Data rychlostí, která jsem nasbíral jsou pro Excel a Python zbrazeny v nepdporovaném formátu, proto je třeba je nejdříve upravit odpovídající formát. Na data ze sloupce rychlost aplikujeme funkci Lambda, která vrací data již jen s jednou desetinou čárkou a převede je na float.

```
1 y = []
2 for d in data["rychlost[m/s]"]:
3     if float(d[:6]) >= 3:
4         y.append((float(d[:6])))
5     else:
6         continue
7 x = list(range(0,len(y)))
```

Zde je zobrazen jiný způsob, kterým můžemu převést datový typ a vyčistit data od chybových rychlostí. Vytvořím dva listy. První naplněný vyčištěnými hodnotami a druhý značící počet záznamů.

```
1 import matplotlib.pyplot as plt
2
3 plt.bar(x,y, label="Rychlosti strel", color = "purple")
```

```
4
5  plt.xlabel("cislo strely")
6  plt.ylabel("rychlost[m/s]")
7  plt.title("Graf rychlosti strel")
8  plt.legend()
9  plt.show()
```



Obrázek 7.4: Rychlost

Tento kód popisuje tvorbu vlastního grafu za použití výše vytvořených listů. Graf má popsané jednotlivé osy a má i svojí legendu. Tímto způsobem je graf lépe čitelný.

```
1  print("Maximalni rychlost: ", rychlost.max(), "m/s")
2  print("pocet pokusu: ", len(data))
```

V této části dochází k vyhodnocení dat. Vyhodnocuji nejrychlejší střelu z datasetu a počet pokusů. V souboru jsou dále zajímavé časy jednotlivých senzorů. Z těchto dat budu schopen zjistit jaký objekt prošel přes brankovou čáru.

7.2.1 vyhodnocení rychlosti

Z celkového porovnání třech hlavních střel stolního fotbalu vyšla nejlépe střela Pinshot. Její rychlost je okolo 40 km/h. Druhá nejoblíbenější střela Snakeshot dosahuje rychlosti podobné a to okolo 36 km/h. Nejpomalejší ze změřených dat vyšla střela Pullshot, její naměřená rychlost byla okolo 30 km/h. Pokud by hráč chtěl využít nejrychlejší střelu je nejlepší naučit se Pinshot.

7.2.2 Vyhodnocení dat během zápasu

Systém jsem otestoval během ligového zápasu jeden na jednoho. Na výsledcích, které jsem nasbíral bude podrobně vysvětlena taktika a odůvodněny jednotlivé střely. Hráč používá střelu Snakeshot.

Obrázek 7.5: Prvni střelba[1]

Na prvních třech střelách se brankář a útočník seznamují. Brankář se snaží krýt střed a tyč od sebe. Je to z důvodu aby zjistil, jak moc přesnou střelu útočník má. První gól šel přímo k tyči. V tuto chvíli uvolňuje brankř vzdálenější tyč. Tím se přiblíží ke straně blíže ke střelci. Druhý gól šel znovu přesně k tyči. Při třetí střele se brankář ještě o trochu posunul. Kryl si střed a bližší tyč. Gól prošel přímo ke vzdálenější tyči.

Z těchto výsledků plyne, že útočník je velmi nebezpečný, protože je schopen umístit střelu k oběma tyčím. Střela k sobě je vždy rychlejší než ta od sebe a to řádově o 3 m/s.

Při dalších třech střelách je zřetelné, že brankář zrychlil intenzitu otevírání a zavírání děr. Útočník zareagoval střelbou mezi rozpohybované hráče. U poslední střely je zřetelné, že hráč netrefil přesně na tyčku, ale zde vyhrála hlavně rychlost. Takto skončil první leg. Velmi důležité je zhodnocení těchto šesti střel, protože na turnaji, kdy se hraje na 4 vítězné legy, je třeba predikovat a navádět útočníka na místa, kam střílí, ale postavit mu zde hráče. Tyto zápasy jsou velmi náročné na psychiku hráčů. Ve světovém fotbálku není neobvyklé, že hráči dost dobře chytají, takže během legu na branku letí 6-12 střel. Všechny střely byly vystřeleny z utočné řady.

7.2.3 Rychlost střely amatérské hráčky

I v tomto sportu hrají dámy. Na turnajích existuje spousta ženských kategorií. Dámy mají střelu pomalejší než muži. Srovnáváme jen na malém vzorku jedné amatérská hráčky. Beru v potaz, že střela Pinshot měla v průměru něco okolo 10 m/s. S touto hodnotou budu porovnávat 10 naměřených výsledků.

7.2.4 Shrnutí

Z výsledků je zřetelné, že střely testovací hráčky jsou v průměru o 4 m/s pomalejší než u mužů. Pomalejší střely se lépe chytají na reflex a lépe se dají vykmitnout. Pokud má hráčka střelu přesnou a perfektně naučenou, stejně tak jako zvládání stresu, tak jí rychlost nijak neomezuje v poražení jakéhokoliv protihráče.

Kapitola 8

Závěr

V této práci jsem se věnoval automatizaci za použití microcontorleru Arduino (kapitola Řídící systém). Srovnal jsem zde jednotlivé typy desek Arduino, ESP a Microprocesor Raspberry Pi. Ve stejné kapitole jsem odůvodnil výběr desky. Popsal jsem základy stolního fotbalu a koncepci utkání (kapitola Úvod). Nastínil jsem možnosti řešení, výběr senzoru kapitola Koncepce. Popsal jsem realizaci samotného systému, zapojení, užití řešení a výběru kinematických vzorců pro zjištění rychlosti. Dále informace o systému přerušení, návrh archivace výsledků a jejich zobrazení na displeji (kapitola Realizace). Je zde i okomentovaný kód. Praktická část, která byla časově nejnáročnější částí této práce je popsána v kapitole Průběh výroby zařízení, jíž jsem rozdělil na celkem tři fáze problémů, které jsem musel řešit při tvorbě tohoto systému. Výsledky, které jsem naměřil, jsem podrobil datové analýze za použití jazyka Python v (kapitole Vyhodnocení výsledků). Je zde i odpověď na nejryhlejší střelu.

Úkolem této práce bylo vyvinout funkční systém na snímání polohy a rychlosti míče na brankové čáře stolního fotbálku. Během dlouhé konstrukce jsem se potýkal se spoustou problémů spojených se špatným zapojením, neznalostí hardwaru a softwaru. Výstupem této práce je funkční systém, který může být instalován i během zápasu a tak odhalit taktiky hráčů - nejoblíbenější umístění střely a její rychlost. S rychlou a přesnou střelou hráč zvyšuje pravděpodobnost úspěchu. Systém byl sestaven z hardwaru Arduino Due, Infra LED diod, fototranzistorů, tlačítek, modulů pro SD kartu, RTC modulu, displejů. V této práci byly využity tyto softwary: "Overleaf, Easy Eda, Arduino IDE, Microsoft Excel,Python, Inventor Naměřená data se archivují a v další fázi projektu, by z nich mělo být patrné jaká střela byla použita. Toto je klasifikační úloha a je tedy třeba sestavit vhodný matematický model. Stejně tak přidělat vhodný tréninkový systém. Jedním z návrhů je řada 16 diod rosvícených nad brankovou čárou s možností automatického a manuálního systému. Použití nového open source programovacího jazyka. Pro tento projekt bych použil Python a napsal jednoduchou webovou aplikaci pro vizualizaci dat. Změna hardwaru je možná při jiné volbě metodiky řešení. Hardware, který by měl tolik pinů s možností přerušení je velmi složité a nákladné najít. V případu použití laserové brankové čáry by bylo vhodné použít ESP microcontroler.

Seznam obrázků

Literatura

[1] : SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 978-80-251-4840-2

[2] VODA, Zbyšek. Průvodce světem Arduina. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.

[3] Arduino DUE - HW Kitchen. Ochutnejte s námi Arduino! | HWKitchen.cz [online]. Copyright © HW Kitchen, všechna práva vyhrazena [cit. 06.05.2018]. Dostupné z: https://www.hwkitchen.cz/Arduino-due/

[4] RTC Arduino.cz - Webový magazín o Arduinu a elektronice [online]. Dostupné z: https://Arduino.cz/

[5] RTC Arduino and DS3231 Real Time Clock Tutorial - HowToMechatronics. How To Mechatronics [online]. Copyright © 2018 HowToMechatronics.com. All rights reserved. [cit. 06.05.2018]. Dostupné z: https://howtomechatronics.com/tutorials/Arduino/Arduino-ds3231-real-time-clock-tutorial/

[6] Arduino - SD . Arduino - Home [online]. Dostupné z: https://www.Arduino.cc/en/Reference/SD

[7] Attention RequIRed! | Cloudflare. Attention RequIRed! | Cloudflare [online]. Dostupné z: https://computers.tutsplus.com/tutorials/how-to-add-an-sd-card-data-logger-to-an-Arduino-project–cms-21713

[8] Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877 | Vývoj.HW.cz. Vývoj.HW.cz | Vše o elektronice a programování [online]. Copyright © 1997 [cit. 07.05.2018]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-I2C-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256

[9] SPI rozhraní : Tajned - .NET and Embedded Development. Tajned - .NET and Embedded Development [online]. Copyright © http [cit. 07.05.2018]. Dostupné z: http://www.tajned.cz/2016/12/spi-rozhrani/

[10] Arduino - WiFi . Arduino - Home [online]. Dostupné z: https://www.Arduino.cc/en/Reference/WiFi

[11] Arduino Bluetooth modul HC-05 | Arduino návody. Webový magazín o Arduinu | Arduino návody [online]. Dostupné z: http://navody.Arduino-shop.cz/navody-k-produktum/Arduino-bluetooth-modul-hc-05.html

[12] Martinek,Radislav. Senzory v průmyslové praxi. 1.vyd. Praha: BEN-TECHNICKÁ LITERATURA,2004 ISBN 978-80-73001-14-8;80-73001-14-4

[13] Arduino Reference. Arduino - Home [online]. Dostupné z: https://www.Arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

[14] Arduino a přerušení – uArt.cz. uArt.cz – Electronics, programming and stuff... [online]. Dostupné z: https://uart.cz/271/Arduino-a-preruseni/

[15] [online]. Dostupné z: $https://embedded.fel.cvut.cz/sites/default/files/kurzy/lpe/photodiode_photoresistor/Photoresistor.pdf$

[16] $301MovedPermanently.301MovedPermanently[online].Dostupnéz : http://www.elektross.gjn.cz/soucastky/jeden_prechod/fotodioda.html$

[17] [online]. Dostupné z: .(https://eluc.kr-olomoucky.cz/verejne/lekce/612

[18] $LED\check{}Wikipedie.[online].Dostupnéz : https://cs.wikipedia.org/wiki/LED$

[19] $[online].Dostupnéz : https://www.kof.zcu.cz/st/dp/horsky/html/2fotoel.htmlhttp://fyzika.jreichl.com/main.article/view/723-fyzikalni-podstata$

[20] $[cit.01.06.2019].Dostupnéz : http://numba.pydata.org/numba-doc/latest/index.html$

[21] ESP32-DevKitC Ver. D | ESPRESSIF | SOS electronic. Elektronické součástky | SOS electronic [online]. Copyright © SOS electronic s.r.o. 1991 [cit. 01.06.2019]. Dostupné z: https://www.soselectronic.cz/products/espressif/esp32-devkitc-ver-d-305403

[22] [online]. Dostupné z: https://www.mouser.co.uk/new/espressif/espressif-esp32-devkitc-boards/

[23] Srovnání-desek. Jakub Valenta, FS ČVUT. 2018

Příloha A

Přílohy

A.1 Obsah CD

JakubValentaBCprace.pdf	PDF verze Bc práce
data	csv naměřené výsledky
kód	ino kód pro Arduino
analýza	py příkazy v pythonu pro analýzu
součástka	ipt ochranná součástka

Snejk

May 26, 2019

In [1]:
```
import pandas as pd
import numpy as np
import os
```

In [2]:
```
#data = pd.read_csv(os.path.join('.', 'Pinshot.csv'))
data=pd.read_csv("/home/jakub/Downloads/Snejk.csv")
data
```

Out[2]:

	rychlost[m/s]	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	\
0	10.6235.00	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
1	8.9635.00	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
2	9.6435.00	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
3	9.1235.00	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
4	9.0635.00	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
5	8.3135.00	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
6	8.8935.00	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
7	8.4835.00	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
8	10.0135.00	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
9	8.7035.00	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
10	9.0235.00	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
11	10.0535.00	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
12	10.9035.00	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
13	10.4935.00	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
14	0.0935.00	0	0	0	0	0	0	0	1	1	1	1	0	0	0	
15	9.9435.00	0	0	0	0	0	0	0	1	1	1	0	0	0	0	
16	9.6135.00	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
17	8.2035.00	0	0	1	1	1	0	0	0	0	0	0	0	0	0	
18	9.1435.00	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
19	8.8135.00	0	0	0	0	0	0	0	0	0	1	1	1	0	0	
20	8.6135.00	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
21	3.3235.00	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
22	10.4135.00	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
23	8.6935.00	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
24	3.6135.00	0	0	0	0	0	1	1	1	0	0	0	0	0	0	
25	10.4335.00	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
26	10.5235.00	0	0	0	0	1	1	1	0	0	0	0	0	0	0	
27	10.4435.00	0	0	0	0	0	0	0	0	0	0	1	1	0	0	

```
28     8.8535.00   0   0   0   0   0   1   1   1   0   0    0    0    0    0
29     8.6435.00   0   0   0   0   0   1   1   1   0   0    0    0    0    0
30     8.2435.00   0   0   0   0   0   1   1   1   0   0    0    0    0    0
```

```
    S14  S15
0     0    0
1     0    0
2     0    0
3     0    0
4     0    0
5     0    0
6     1    0
7     0    0
8     0    0
9     0    0
10    0    0
11    0    0
12    0    0
13    0    0
14    0    0
15    0    0
16    0    0
17    0    0
18    0    0
19    0    0
20    0    0
21    0    0
22    0    0
23    0    0
24    0    0
25    0    0
26    0    0
27    0    0
28    0    0
29    0    0
30    0    0
```

In [13]:
```
data_S = data.filter(regex='^S.*')
data_S
```

Out[13]:

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

7	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
9	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
15	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
17	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
19	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
21	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
23	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
24	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
26	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
28	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
29	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
30	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

In [17]:
```
bin_filter = data_S.sum(axis=1) <= 3
```

In [18]:
```
data_S_fil = data_S[bin_filter]
data_S_fil.style.apply([lambda x: 'background-color: yellow' if x == 1 else 'background
```

Out[18]: <pandas.io.formats.style.Styler at 0x7f605b49c080>

In [18]:
```
total = data_S_fil.sum()
total
```

Out[18]:
```
S0      0
S1      0
S2      1
S3      1
S4      4
S5      8
S6     11
S7     10
S8      2
S9      7
S10     8
S11    10
S12     4
S13     2
```

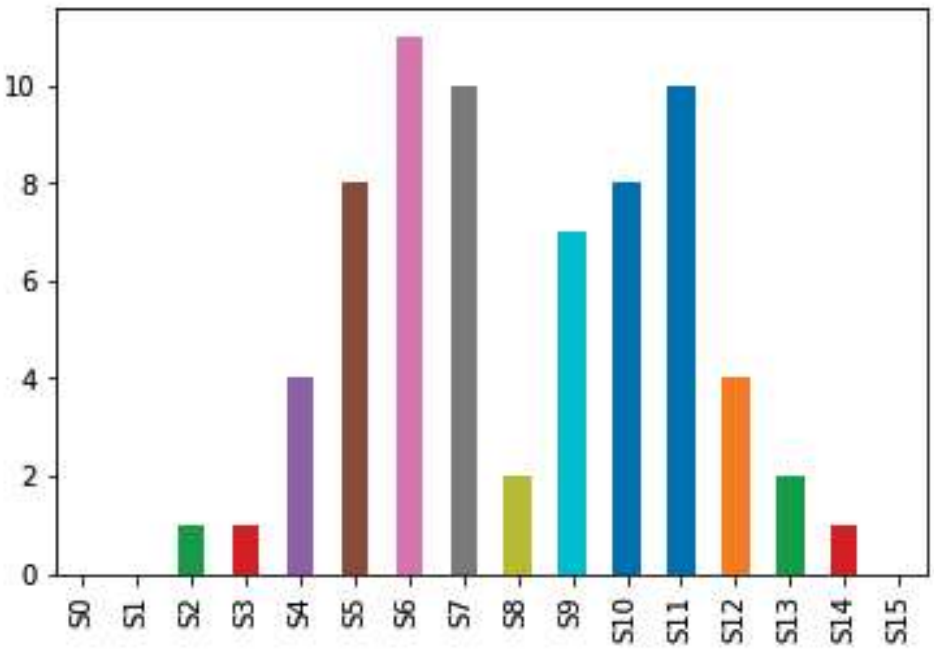
```
S14     1
S15     0
dtype: int64
```

```
In [20]: total.rename('total', inplace=True)
```

```
Out[20]: S0      0
         S1      0
         S2      1
         S3      1
         S4      4
         S5      8
         S6     11
         S7     10
         S8      2
         S9      7
         S10     8
         S11    10
         S12     4
         S13     2
         S14     1
         S15     0
         Name: total, dtype: int64
```

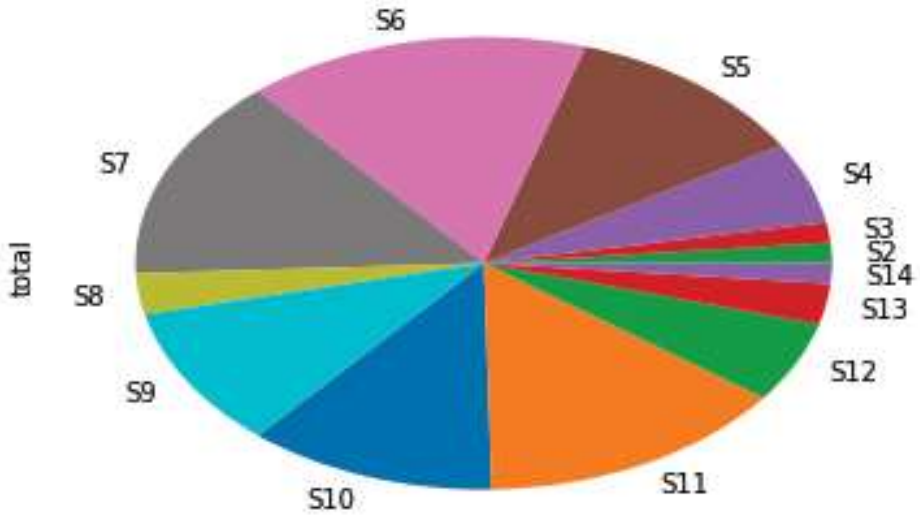
```
In [35]: total.plot.bar()
```

```
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2207f7c668>
```



```
In [22]: total.plot.pie()
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2220245b70>
```




```
In [21]: y = []
         for d in data["rychlost[m/s]"]:
             if float(d[:6]) >= 3:
                 y.append((float(d[:6])))
             else:
                 continue

         x = list(range(0,len(y)))
         print(x[:15])
         print(x[16:29])
         print(y[:10])
         print(y[11:20])
         print(y[21:29])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]
[10.623, 8.9635, 9.6435, 9.1235, 9.0635, 8.3135, 8.8935, 8.4835, 10.013, 8.7035]
[10.053, 10.903, 10.493, 9.9435, 9.6135, 8.2035, 9.1435, 8.8135, 8.6135]
[10.413, 8.6935, 3.6135, 10.433, 10.523, 10.443, 8.8535, 8.6435]
```

```
In [10]: import matplotlib.pyplot as plt #--> plt je takové dogma v python komunit
```

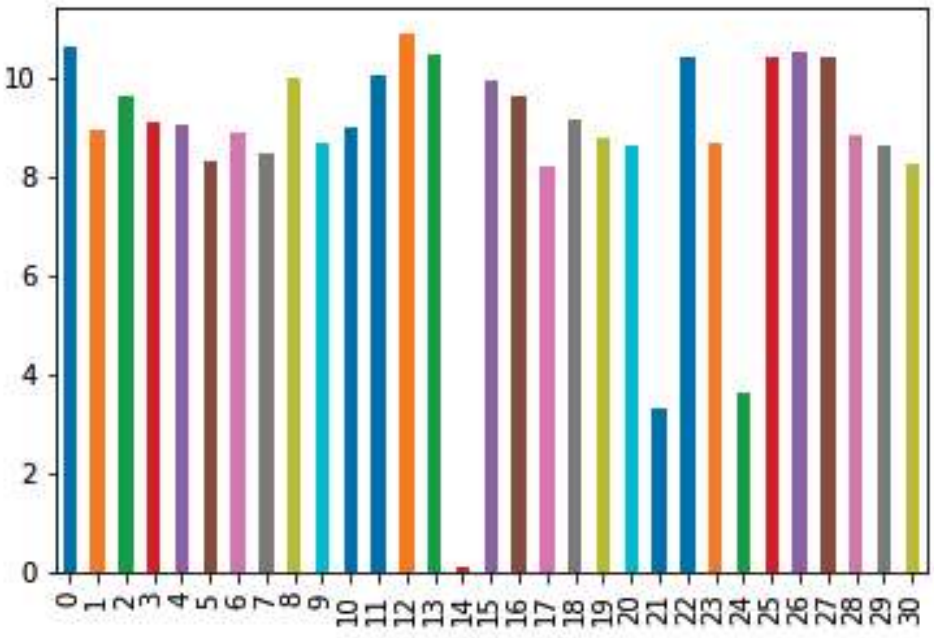
```
        plt.bar(x,y, label= "Rychlosti stel", color = "purple") #zmna nazvu grafu - >  plot/bar

        plt.xlabel("íslo stely")
        plt.ylabel("rychlost[m/s]")
        plt.title("Graf rychlostí stel")
        plt.legend() #--> pro zobrazení legendy
        plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

```
In [11]: data['rychlost[m/s]'].apply(lambda x: x[:-3]).astype(float).plot.bar()
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f605b555208>
```




```
In [12]: print("Maximální rychlost stely snejk je: ", max(y))
```

```
Maximální rychlost stely snejk je:  10.903
```

Pinshot

May 26, 2019

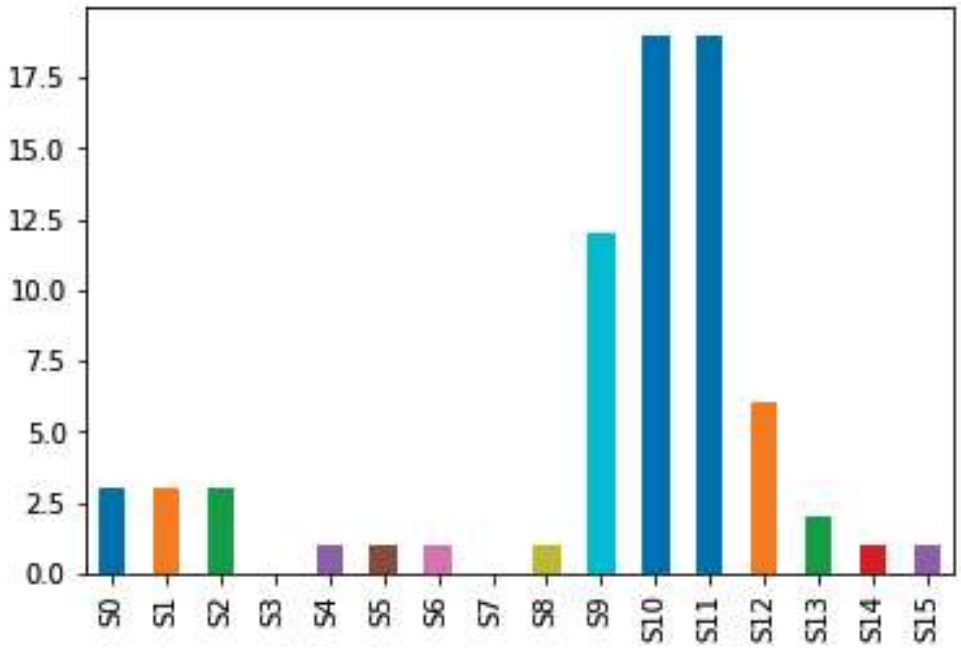
In [2]:
```
import pandas as pd
import numpy as np
import os

data=pd.read_csv("/home/jakub/Downloads/Pinshot.csv")
data_S = data.filter(regex='^S.*')
bin_filter = data_S.sum(axis=1) <= 3
data_S_fil = data_S[bin_filter]
data_S_fil.style.apply([lambda x: 'background-color: yellow'
                        if x == 1 else 'background-color: black'])
```

Out[2]: <pandas.io.formats.style.Styler at 0x7fc3b14673c8>

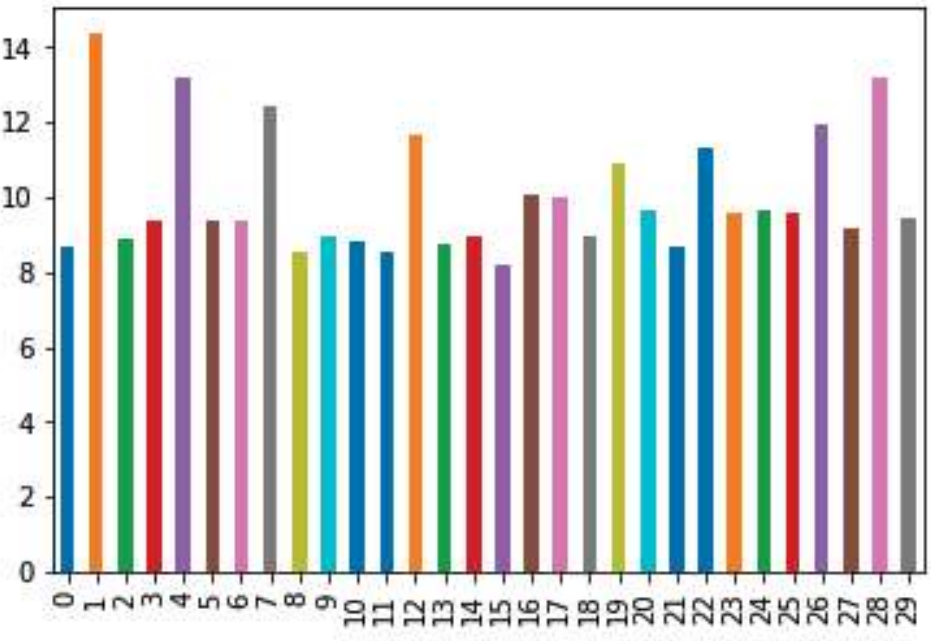
In [3]:
```
total = data_S_fil.sum()
total.rename('total', inplace=True)
total.plot.bar()
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc3aeaac710>



```
In [4]: data['rychlost[m/s]'].apply(lambda x: x[:-3])
        .astype(float).plot.bar()
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc3ad2436d8>
```




```
In [6]: print("Maximální rychlost stely je: ", data['rychlost[m/s]']
              .apply(lambda x: x[:-3]).astype(float).max(), "m/s", "poet pokus: ", len(data))
```

```
Maximální rychlost stely je:  14.3835 m/s poet pokus:  30
```

Stahovacka

May 26, 2019

In [2]:
```
import pandas as pd
import numpy as np
import os

data=pd.read_csv("/home/jakub/Downloads/Stahovacka.csv")
data_S = data.filter(regex='^S.*')
bin_filter = data_S.sum(axis=1) <= 3
data_S_fil = data_S[bin_filter]
data_S_fil.style.apply([lambda x: 'background-color: yellow'
                        if x == 1 else 'background-color: black'])
```

Out[2]: <pandas.io.formats.style.Styler at 0x7f2950b91208>

In [3]:
```
data_S
```

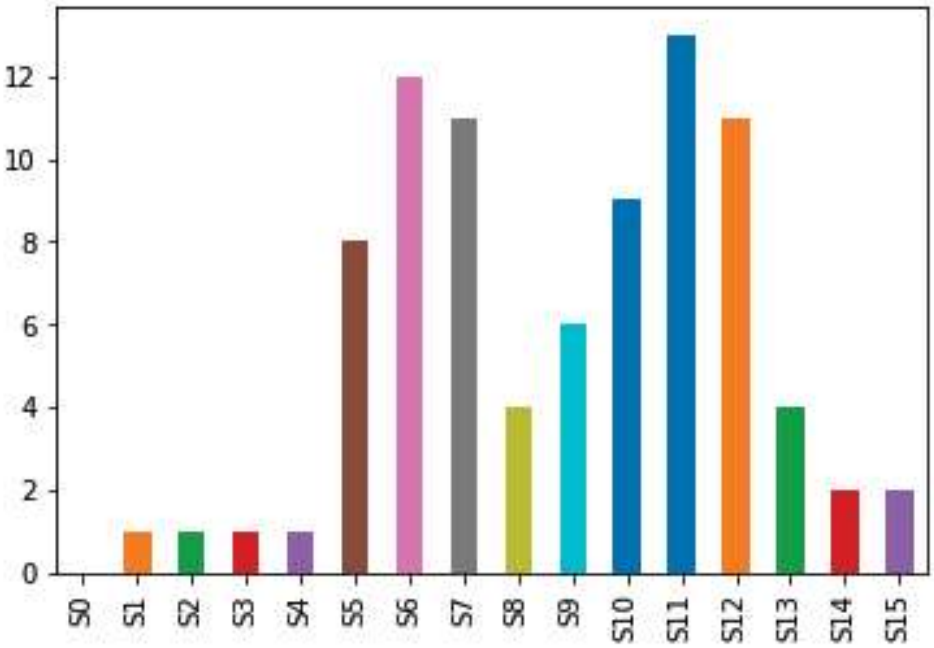
Out[3]:

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
5	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
12	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
14	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
15	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
18	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
19	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

22	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
24	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
27	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
28	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
31	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
32	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

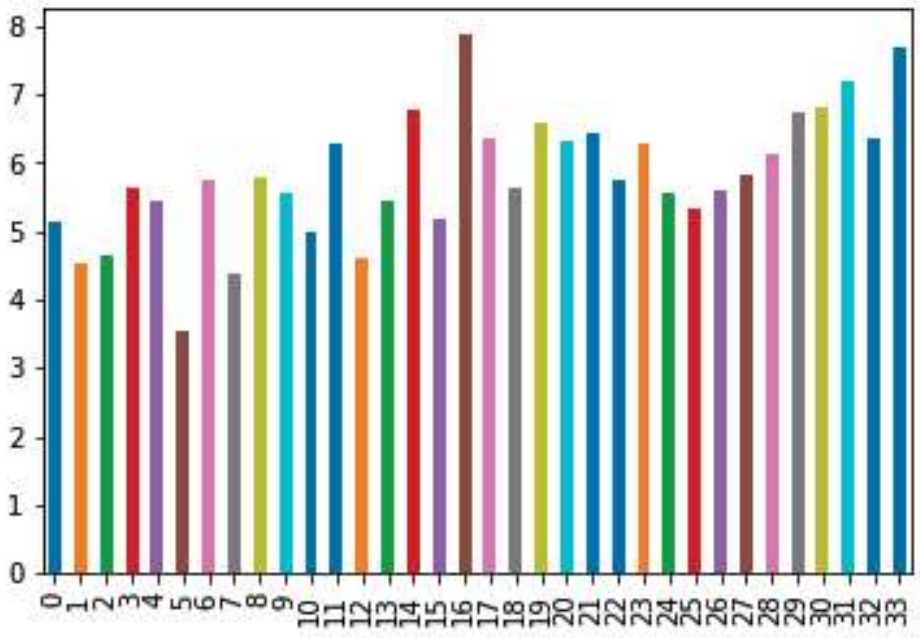
In [6]:
```
total = data_S_fil.sum()
total.rename('total', inplace=True)
total.plot.bar()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5e0d42bd30>



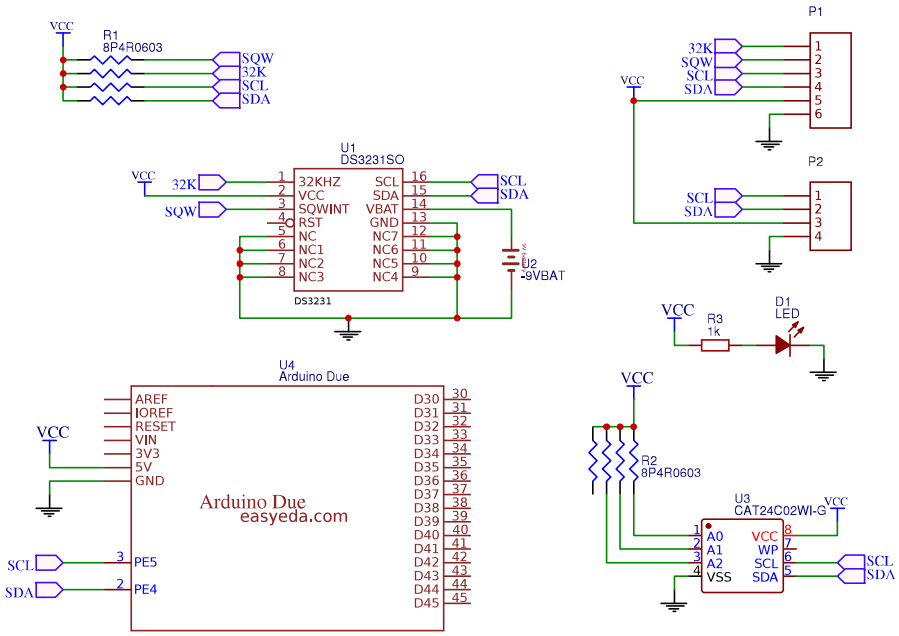
In [30]:
```
data['rychlost[m/s]'].apply(lambda x: x[:-3]).astype(float).plot.bar()
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5e0c0cd630>

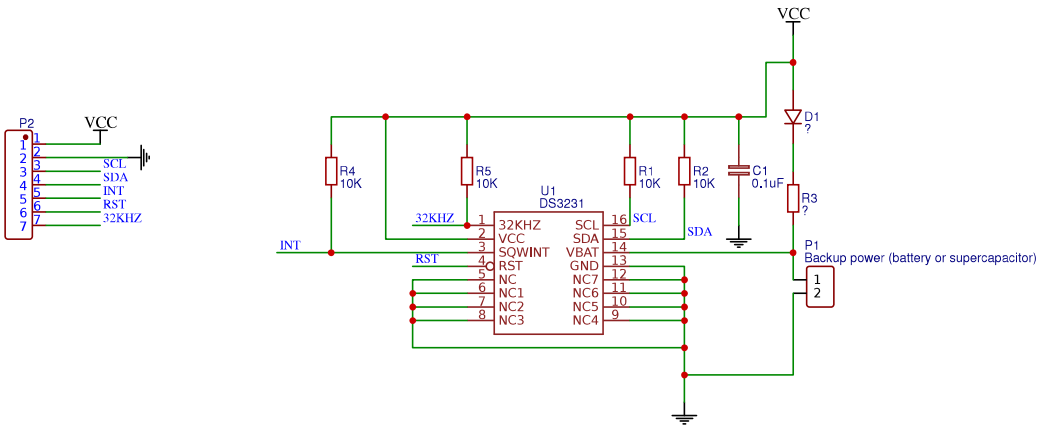



```
In [18]: print("Maximální rychlost stely  stahovaka je: ", data['rychlost[m/s]']
                .apply(lambda x: x[:-3]).astype(float).max(), "m/s ,", "poet pokus: ", len(data))
```

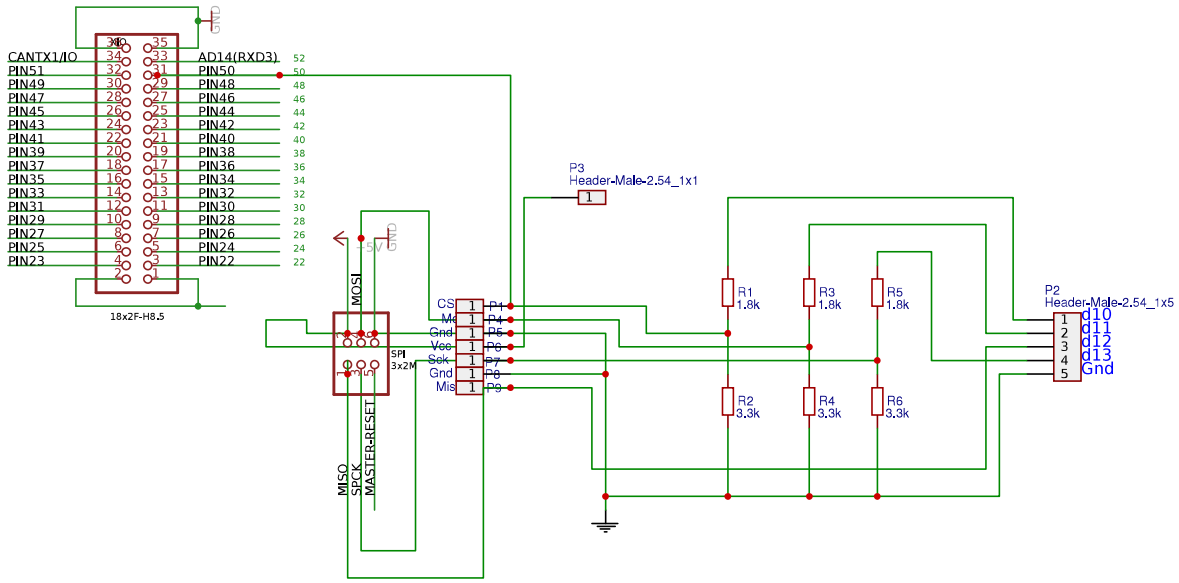
```
Maximální rychlost stely  stahovaka je:  7.8835 m/s , poet pokus:  34
```



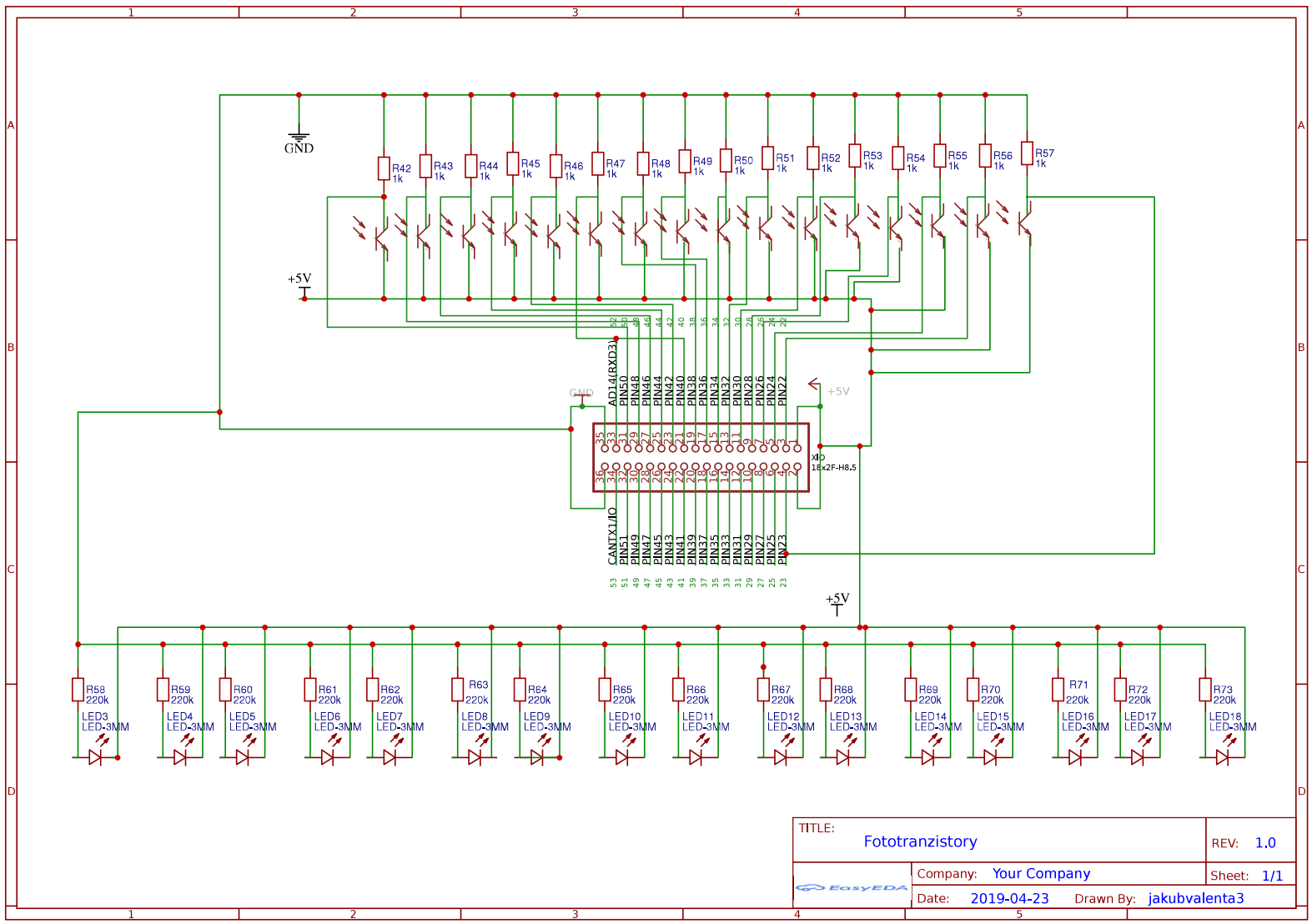
TITLE: RTC		REV: 1.0
EasyEDA	Company: Your Company	Sheet: 1/1
	Date: 2019-04-23 Drawn By: jakubvalenta3	



TITLE: DS3231		REV: 1.0
EasyEDA	Company: Your Company	Sheet: 1/1
	Date: 2019-04-22 Drawn By: jakubvalenta3	



TITLE: Card module		REV: 1.0
EasyEDA	Company: Your Company	Sheet: 1/1
	Date: 2019-04-22 Drawn By: jakubvalenta3	



TITLE: Fototranzistory		REV: 1.0
EasyEDA	Company: Your Company	Sheet: 1/1
	Date: 2019-04-23 Drawn By: jakubvalenta3	