VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

OPTICKÝ SNÍMAČ RYCHLOSTI POVRCHU
OPTICAL SPEED SENSOR OF SURFACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Ondřej Vaněk**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Sedlák, Ph.D**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Biomedicínské a ekologické inženýrství**
Ústav biomedicínského inženýrství

Student: Bc. Ondřej Vaněk ***ID:*** 155616

Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Optický snímač rychlosti povrchu

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se s problematikou rychlostních snímačů. 2) Sestavte experimentální pracoviště s valníkem a vybraným detektorem. 3) Vytvořte program v prostředí MATLAB pro komunikaci a stanovení rychlosti pohybu povrchu. 4) Experimentálně porovnejte měření rychlosti na povrchu s různou odrazivostí. 5) Proveďte průzkum trhu a vyberte rychlý detektor. 6) Implementujte vytvořený program na vybranou mikroprocesorovou platformu. 7) Navrhněte a vyrobte zařízení. 8) Ověřte funkčnost zařízení v průmyslovém prostředí.

DOPORUČENÁ LITERATURA:

[1] SALEH, B., E., TEICH, M., C.: Fundamentals of photonics. New York: Wiley, 2007. ISBN 978-0471839651.

[2] HALLIDAY, David, Robert RESNICK a Jearl WALKER, DUB, Petr (ed.). Fyzika. 2., přeprac. vyd. Přeložil Miroslav ČERNÝ. Brno: VUTIUM, c2013. Překlady vysokoškolských učebnic. ISBN 9788021441231.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 19.5.2017

Vedoucí práce: doc. Ing. Petr Sedlák, Ph.D.

Konzultant:

prof. Ing. Ivo Provazník, Ph.D.
předseda oborové rady

Abstrakt

Cílem této práce je vytvoření optického snímače pro měření rychlosti povrchu za použití senzorů používaných v počítačových myších. Takovýto optický rychloměr by měl najít uplatnění ve výrobním průmyslu pro levné bezkontaktní měření rychlosti pásových dopravníků či posunu polotovarů, aby bylo možné dalšími přístroji kontrolovat kvalitu produktů. Teoretická část práce zahrnuje prostudování tématiky, seznámení s optickými senzory a jejich použitím. Do praktické části patří vytvoření programu pro komunikaci a stanovení rychlosti pohybu povrchu v MATLABu a pro platformu Arduino©, vytvoření jednoduchého zařízení na platformě Arduino©, sestavení experimentálního měření a provedení testovacích měření.

Klíčová slova

optický senzor, rychlost povrchu, Arduino

Abstract

The aim of this work is development of an optical sensor for surface-velocity measurement based on ordinary optical mouse sensor. The proposed sensor should have find its usage in industrial applications for low-cost contactless measurement of conveyor belts or semi-finished products, thus some other gauges could check quality of products. Theoretical part of this thesis includes introduction to optical sensors and their usage. Rest of the work describes practical part consisting of programming of communication between sensor and PC in MATLAB and Arduino environments, development of simple device based on Arduino hardware, the set-up of experimental workplace and providing speed tests.

Key words

optical sensor, surface velocity, Arduino

Bibliografická citace:

VANĚK, O. *Optický snímač rychlosti povrchu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 74 s. Vedoucí diplomové práce doc. Ing. Petr Sedlák, Ph.D

Prohlášení

Prohlašuji, že svou závěrečnou práci na téma Optický snímač rychlosti povrchu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009Sb.

V Brně dne 19. května 2017 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Petru Sedlákovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc při zpracování mé diplomové práce. Dále děkuji Ing. Jiřímu Majznerovi, Ph.D. za cenné rady při programování platformy Arduino.

V Brně dne 19. května 2017 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

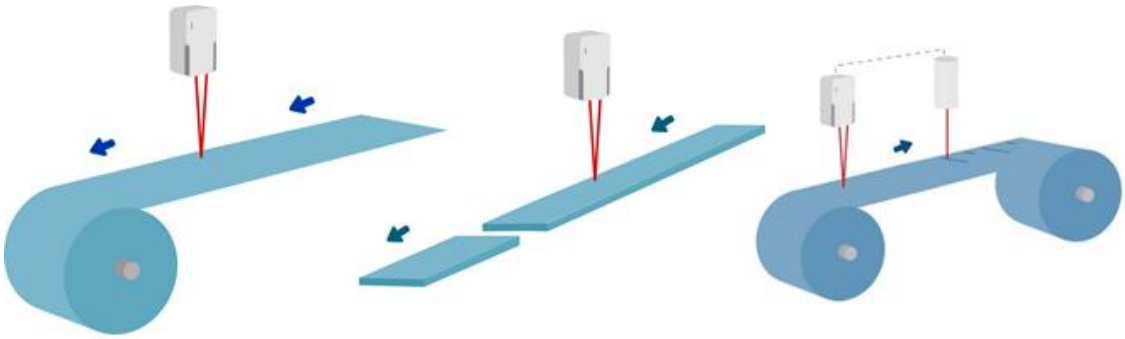
Využití optických snímačů rychlosti nachází spoustu uplatnění v průmyslu, jelikož mají oproti kontaktním snímačům rychlosti mnoho výhod. Patří mezi ně především nízké požadavky na údržbu a neovlivňování měřeného objektu, například zanecháváním stop na objektu, nebo opačně například ulpívání nového nátěru na měřidle. Dále díky bezkontaktnosti měření nedochází k žádnému ovlivňování rychlosti objektu. Ačkoliv na trhu existují přístroje pro bezkontaktní měření rychlosti, jejich cena je poměrně vysoká, a vznikl tedy podnět na vytvoření levnější varianty. Ta je založena na využití senzorů z optických počítačových myší.

První část práce popisuje problematiku rychlostních snímačů, jejich principy a uplatnění. V práci jsou dále porovnány senzory z optických myší a jsou z nich vybrány senzory hodící se pro ověření, zda je možné vytvořit rychloměr založený na tomto principu. Senzory jsou programovány podle doporučení výrobců, k čemuž je využito elektronické platformy Arduino. Ověřování dosažených vlastností zařízení bylo v průběhu práce pravidelně prováděno měřením na lineárním dopravníku. Pro umožnění měření rychlosti pohybujících se povrchů na vzdálenost decimetrů byla před senzor instalována optická soustava a externí zdroj osvětlení měřeného povrchu. Jednotlivé komponenty jsou ve výsledném zařízení vloženy do plastové konstrukce vytvořené pro zaručení mechanické odolnosti zařízení. Výstupem zařízení je měřená rychlost povrchu, která je posílána přes sériovou sběrnici do dalších zařízení. V krytu zařízení je instalován displej sloužící pro zobrazení aktuální rychlosti.

V současném stavu zařízení nedosahuje takových kvalit jako jiná, v průmyslu využívaná zařízení. Je však dokázáno, že zařízení tohoto druhu je možné pro měření rychlosti využít. Jsou zde také prezentovány návrhy pro další zlepšování tohoto zařízení.

1. Optické senzory pro měření rychlosti

Měření rychlosti pohybu povrchu je v současné době velmi důležitým prvkem při samostatně řízené produkci a nalézá užití při mnoha aplikacích. Liší se jak ve způsobu měření, tak v jeho přesnosti. Využití takovýchto snímačů najdeme například v tiskárnách pro měření rychlosti papíru během tisku, nebo při kontrole rychlosti produktu, jenž má být nastříhán na pravidelné kusy.



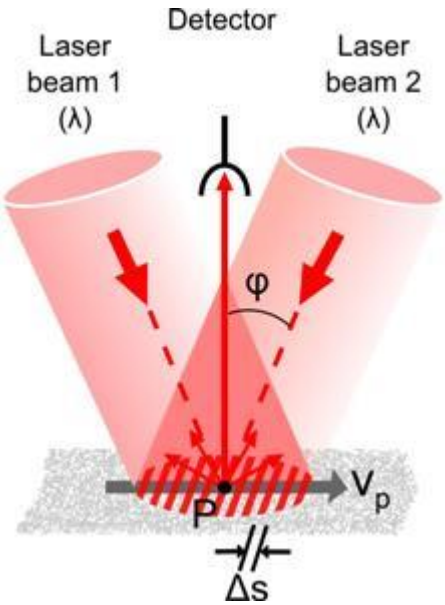
Obrázek 1-1 – Ukázky použití optických přístrojů pro měření rychlosti, [21]

Hlavní výhoda měření pomocí optických senzorů spočívá ve faktu, že monitorovaný objekt je měřen bezkontaktně. Na měřených objektech tak nevznikají žádné stopy, jako například po kolečku při měření kontaktním, ani není ovlivňována rychlost objektu. Další výhodou je absence pohyblivých částí, což má pozitivní vliv na životnost. V prašném prostředí však může docházet ke znečištění optických částí senzorů a je nutná jejich častá údržba. V praxi bývá tento problém řešen ofoukáváním senzoru. Laserové zařízení mohou mít také problémy s teplotní nestabilitou.

1.1. Měření rychlosti povrchu pomocí Dopplerova jevu

V praxi existují optické senzory pro měření rychlosti povrchu založené na principu Dopplerova jevu, popisujícího změnu frekvence dopadající vlny na detektor v závislosti na rychlosti vzájemného pohybu zdroje vlnění a detektoru. Dva laserové svazky dopadají z určité vzdálenosti a pod určitým úhlem na měřený povrch, na kterém vzniká interferenční obrazec ve formě rovnoměrných světlých a tmavých pruhů, viz Obrázek 1-2. Rozmístění světlých pruhů Δs, tj. interferenčních maxim, je definováno použitou vlnovou délkou λ a úhlem φ, pod kterým záření na podložku dopadá, podle vzorce

$$\Delta s = \frac{\lambda}{2 \sin \varphi}. \tag{1}$$

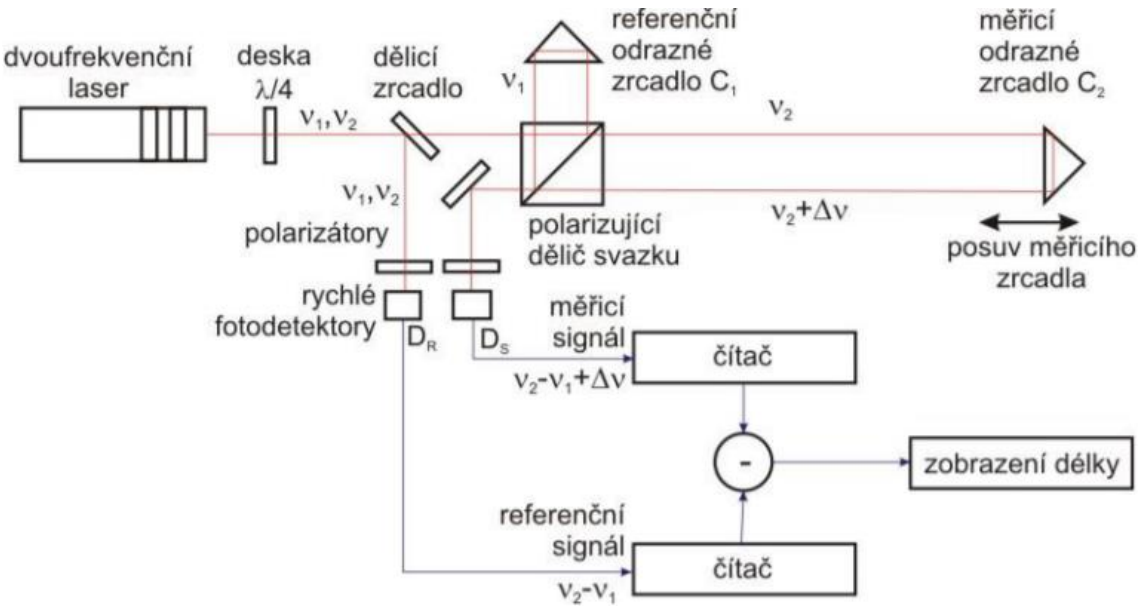


Obrázek 1-2 – Princip laserového měření rychlosti povrchu, [23]

Při pohybu objektu, na kterém vzniká interferenční obrazec, se frekvenčně moduluje intenzita světla odraženého do detektoru. Výsledná frekvence f_D, kterou detektor snímá, je přímo úměrná rychlosti daného objektu v_p,

$$f_D = \frac{v_p}{\Delta s}. \tag{2}$$

Tyto laserové měřiče rychlosti povrchu mohou pracovat také v režimu heterodynní detekce, který spočívá v detekci záznějové frekvence mezi dvěma optickými frekvencemi, které ve formě samostatných vln procházejí laserovým interferometrem. Laserový zdroj generuje dvě optické frekvence (f_1 a f_2) se vzájemně kolmou polarizací. Od tohoto záření se na polopropustném zrcátku oddělí referenční signál, který je snímán fotodetektorem. Na polarizujícím děliči svazku se následně oddělí optická vlna s frekvencí f_1 do referenční větve interferometru a poté se opět spojí se signálem měřícím (f_2 + Δf), který je již ovlivněn pohybem objektu. K interferenci měřeného a referenčního svazku nedochází vlivem vzájemného natočení jejich polarizačních rovin o 90°. Následným průchodem polarizátorem natočením polarizační roviny o 45° vůči oběma směrům dojde ke sloučení vln a je možné detekovat záznějový signál na detektoru. Záznějová frekvence je v klidu rovna rozdílu frekvencí f_1 a f_2. V případě pohybu měřeného objektu dojde k fázovému posunu měřící vlny. Schéma heterodynní detekce znázorňuje Obrázek 1-3. [14]



Obrázek 1-3 – Schéma heterodynní detekce

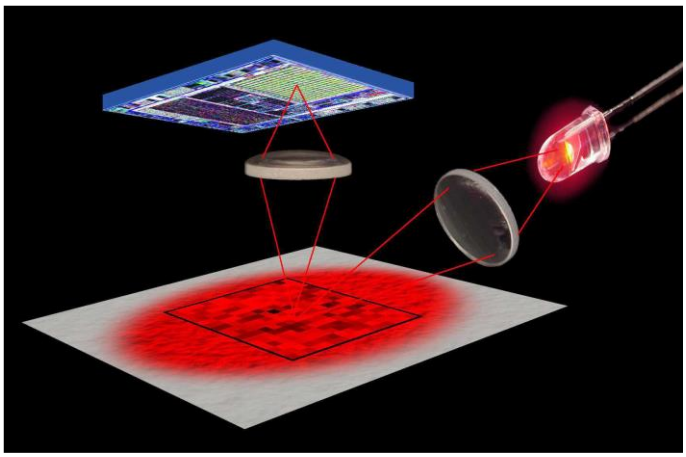
Jedním z typických využití těchto přístrojů je stanovení délky měřeného objektu na základě měřené rychlosti. Délku dostaneme integrací měřené rychlosti. Informace o délce objektu je nutná například pro stříhání kusů o stejné délce. Mezi nejznámější výrobce patří firma Polytec nebo firma Elovis, která uvádí přesnost metody až 0,01 %. [23], [19]

1.2. Optické počítačové myši

Nejčastější použití optických senzorů pro snímání povrchu můžeme najít v počítačových myších – zařízení HID (human interface devices). Měřící systém v myších původně využíval mechanický přenos pohybu přes gumovou kuličku na dva na sebe vzájemně kolmé válečky nebo enkodéry (osa x, y), které mechanicky nebo častěji opticky zaznamenávaly, zda se v daném směru myš pohybuje. Dnes se stanovují parametry pohybu na základě snímání odraženého světelného záření od povrchu, po kterém se myš pohybuje. Rozestavení optické soustavy a detektoru znázorňuje Obrázek 1-4 na následující straně.

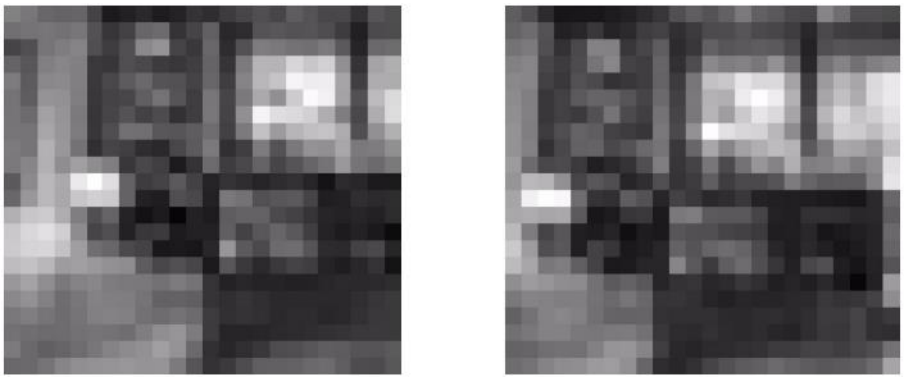
Senzor měří změny pozice optickým snímáním sekvence obrázků se vzorem povrchu a matematicky určí směr a rozsah pohybu. Přitom se využívá nerovností povrchu tak, že se povrch nasvítí ze strany pod určitým úhlem, což zvýší rozlišitelnost charakteristických „nerovností“ povrchu. Výsledné posunutí optické myši je stanoveno na základě vzájemného porovnání snímků. Tento měřící systém se skládá ze tří hlavních

částí: zdroj světla, optická soustava čoček, detektor světla. Spolu s mikroprocesorem a obslužnými elektrickými součástkami jsou umístěny na desce plošných spojů (DPS), která je zasazena do konstrukce myši. Akviziční systém založený na CMOS získané obrázky posílá do DSP (digitální signálový procesor) senzoru, který výpočtem stanoví velikost a směr pohybu v ose x a y, tedy hodnoty Δx a Δy, v závislosti na odlišnosti obrázků přijatých po sobě. Rozdíl takových obrázků znázorňuje Obrázek 1-5. Externí mikrokontrolér informaci o pohybu přečte a přeloží data do PS/2, USB nebo na radiofrekvenční signál pro odeslání do PC. Zde jsou data převedena na pohyb kurzoru. [17]



Obrázek 1-4 – Ilustračně znázorněné rozložení optické soustavy a senzoru, [24]

Druhý integrovaný obvod v myši obsahuje mikrokontrolér, který obsahuje firmware a program pro optický senzor, RAM paměť, časovač a USB rozhraní. Některé senzory (např. ADNS-9500 a ADNS-9800) obsahují oscilátor a není tedy třeba externího časovače. [17]

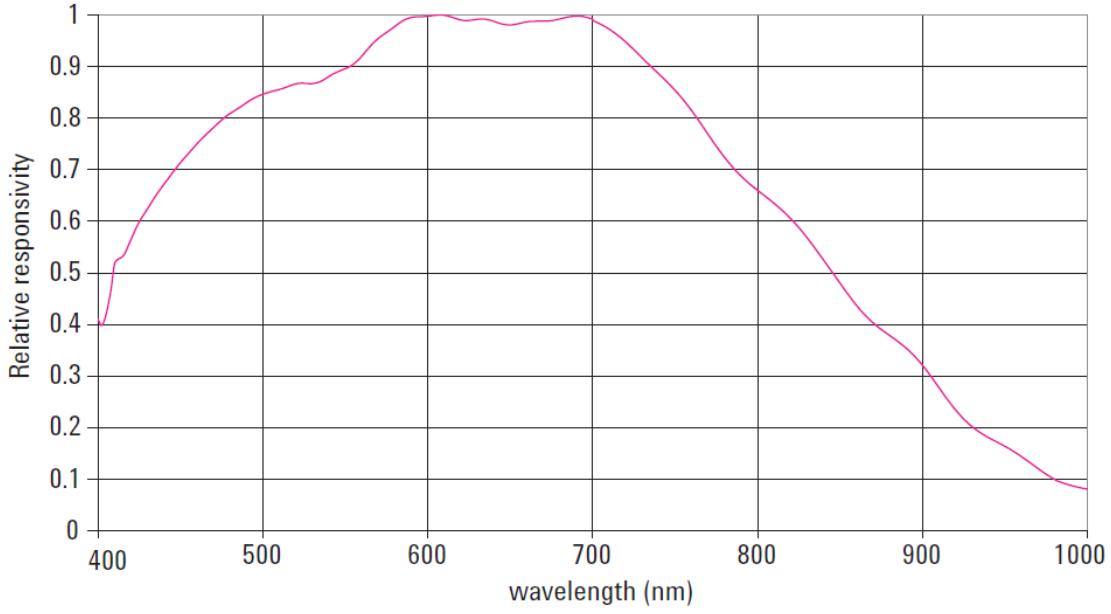


Obrázek 1-5 – Dva za sebou získané snímky povrchu s odstupem 0,67 ms, [24]

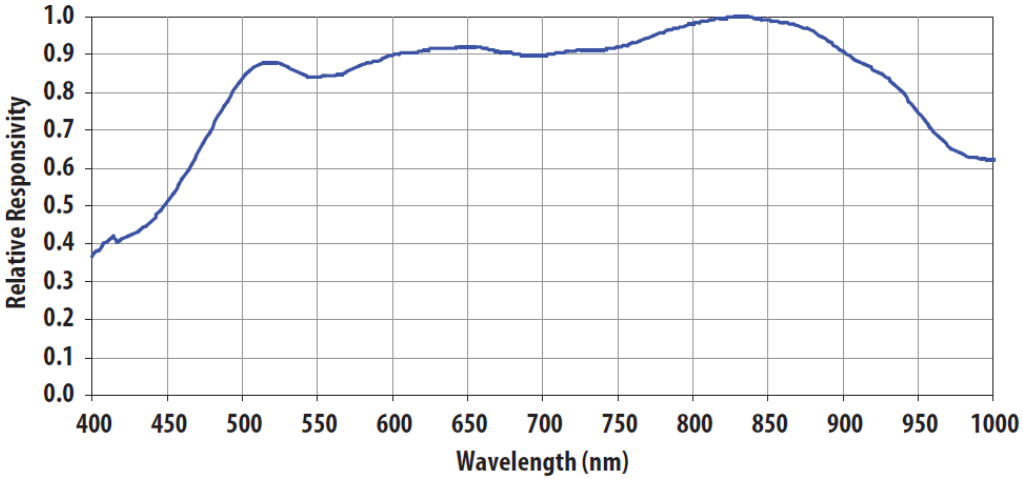
1.2.1. Detektor

Detektorem neboli senzorem se rozumí integrovaný obvod, který obsahuje systém pro akvizici dat (IAS – Image Acquisition System), procesor (DSP – Digital Signal Processor) a sériový port pro komunikaci. Samotný senzor reprezentuje matrice detektorů na bázi COMS technologie. Je uložen zpravidla rovnoběžně se snímaným povrchem. Předním výrobcem optických senzorů do počítačových myší je společnost Broadcom Limited (dříve Avago Technologies). Dalším výrobcem je například PixArt Imaging Inc (například optický senzor PMW3310).

Senzor snímá obrázky povrchu ve škále šedi. Velikost těchto obrázků je od 16x16 pixelů do 30x30 pixelů, podle použitého snímače. V závislosti na velikosti obrázku, kvalitě čoček a vlnové délce použitého záření se mění detaily obrazu. Relativní odezva senzoru ADNS–3080 při různých vlnových délkách je na následujícím grafu (Obrázek 1-6). Je patrné, že nejlépe senzor reaguje na vlnovou délku okolo 600 nm, proto se používá LED s vlnovou délkou 630 nm. U laserového senzoru ADNS-9800 je nejlepší odezva mezi 800 a 850 nm (Obrázek 1-7). Dle uvedených informací od výrobce je pro tento senzor použit VCSEL o vlnové délce 832 až 865 nm. [17], [11]



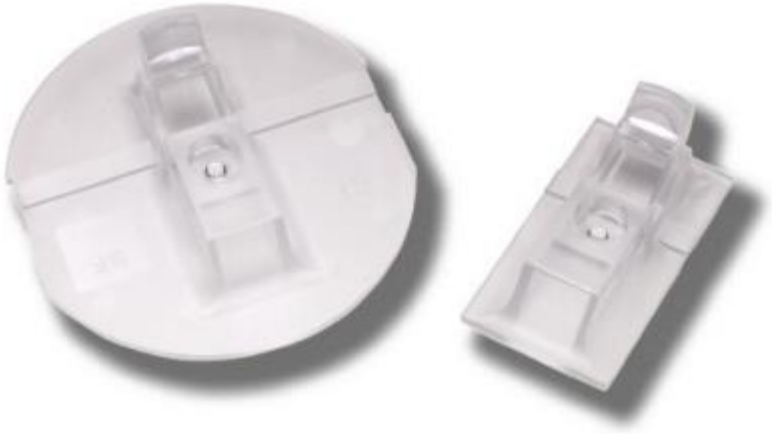
Obrázek 1-6 – Graf závislosti relativní odezvy senzoru ADNS-3080 na vlnové délce, [8]



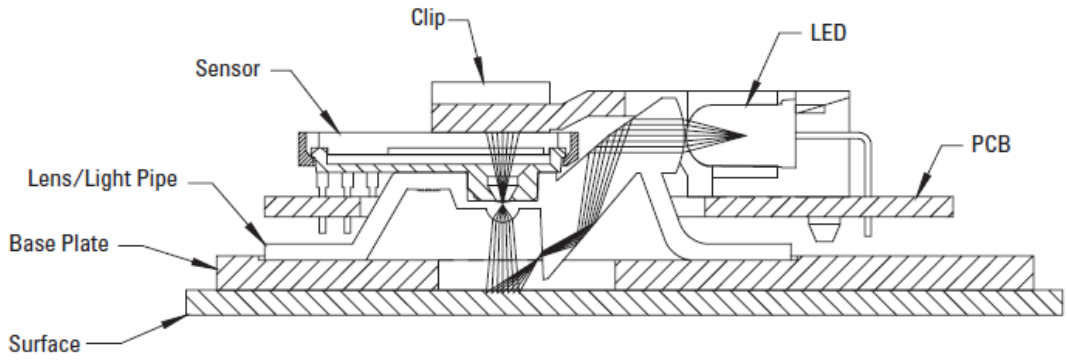
Obrázek 1-7 – Graf závislosti relativní odezvy senzoru ADNS-9800 na vlnové délce, [11]

1.2.2. Optická soustava u počítačových myší

Obrázek 1-8 ukazuje optickou soustavu typickou pro počítačové myši. Tato soustava čoček směruje světlo vycházející ze zdroje na povrch měřeného předmětu a po odrazu jej zaostřuje na rovinu detektoru, viz Obrázek 1-9.



Obrázek 1-8 – Čočka ADNS-2120 pro senzor ADNS-3080, [7]



Obrázek 1-9 – Řez optickou soustavou, [1]

Vzdálenost soustavy čoček k povrchu měřeného objektu je dána použitou soustavou čoček a senzorem. Pro přesnost měření je nutné, aby se povrch nacházel v konkrétní vzdálenosti, protože to ovlivní, jak se obraz promítá do senzoru. Nepatrnou výhodu v této oblasti mají laserové optické myši, které mají větší rozsah vzdálenosti mezi optickou soustavou a snímaným povrchem. Například pro senzory ADNS-9500 a ADNS-9800, tedy typy s VCSEL jako zdrojem světla, je rozsah vzdáleností od 2,18 mm do 2,62 mm. Pro optické senzory se zdrojem světla z LED, kde patří senzor ADNS-3080 a senzor ADNS-2051, je tento rozsah jen od 2,3 mm do 2,5 mm. Všechny zmíněné typy pracují v optimální vzdálenosti 2,4 mm nad povrchem. [1], [8], [10], [11]

1.2.3. Zdroj záření

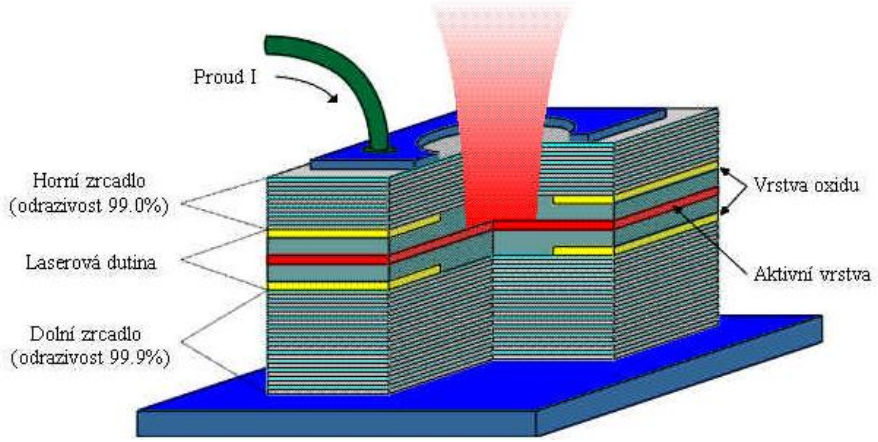
Jako zdroj světla se používá buď luminiscenční dioda (LED) pracující ve viditelné části elektromagnetického záření (u optických myší), nebo laserové světlo VCSEL využívající převážně infračervenou oblast elektromagnetického záření (u optických laserových myší).

- LED

Použití LED jako zdroje osvětlení povrchu je využíváno déle než laserové technologie. Počítačové myši vybavené LED zdrojem na trhu převládají, laserových myší je méně, vzhledem k jejich vyšším pořizovacím nákladům. Vlnová délka použitého světla ovlivňuje kontrast obrazu povrchu, proto se používají ve většině optických myší převážně luminiscenční diody s vlnovou délkou v rozsahu 630 - 650 nm, odpovídající červené barvě. Při použití červeného světla jsou totiž lépe rozpoznatelné detaily než při použití například modrého světla.

- VCSEL

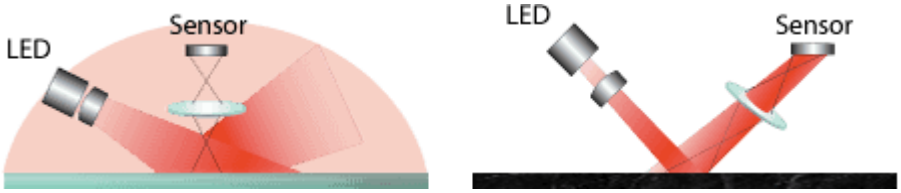
Druhým typem zdroje světla v počítačových myších je laser. Používá se VCSEL (Vertical-Cavity Surface-Emitting Laser), což je technologie plošně vyzařujících laserů, které emitují světlo kruhového průřezu z plochy součástky rovnoběžné s rovinou přechodu. VCSEL lasery jsou účinnější, spotřebovávají méně energie a mají menší úhel divergence svazku než hranově vyzařující lasery (EEL – Edge-Emitting Laser). [4]



Obrázek 1-10 – VCSEL, [20]

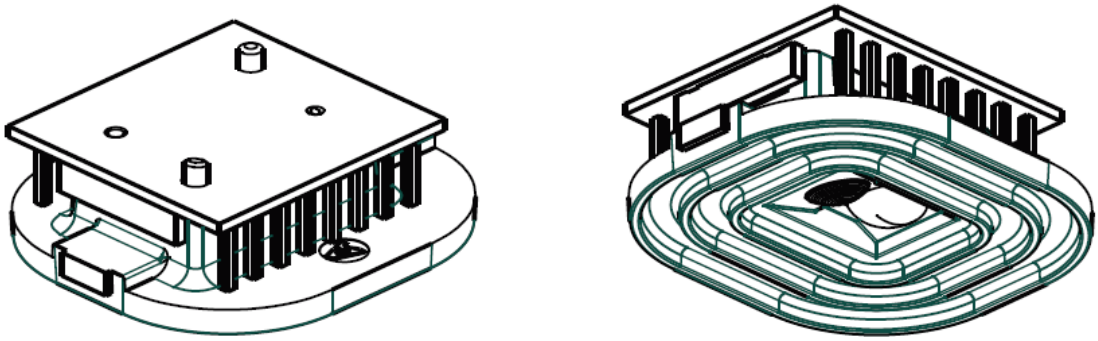
Světlo z laseru používá delší vlnové délky v infračervené oblasti, například 830 nm. Obrázek 1-7 znázorňuje odezvu senzoru ADNS-9800 využívající VCSEL technologii a použitou vlnovou délku 832 – 865 nm. Optické laserové myši soustředí stabilní infračervený paprsek na menší plochu než LED, a jsou tedy schopny pořizovat detailnější obrázky. Lepší ostrosti snímku osvětleného pomocí laseru je dosaženo díky menšímu optickému ohnisku zdroje, nevznikají tedy polostíny. [1], [11], [20], [12]

Firma Logitech uvádí dva způsoby rozestavení emitoru a detektoru u optických myší. V prvním případě senzor snímá rozptýlené záření, ve druhém případě je senzor vzhledem k povrchu měřeného objektu pod stejným úhlem jako zdroj záření, a snímá tedy odražené záření, viz Obrázek 1-11. Právě díky druhému typu dokáže senzor zpracovat větší kontrast a přibližuje se tak laserovým myším. [9]



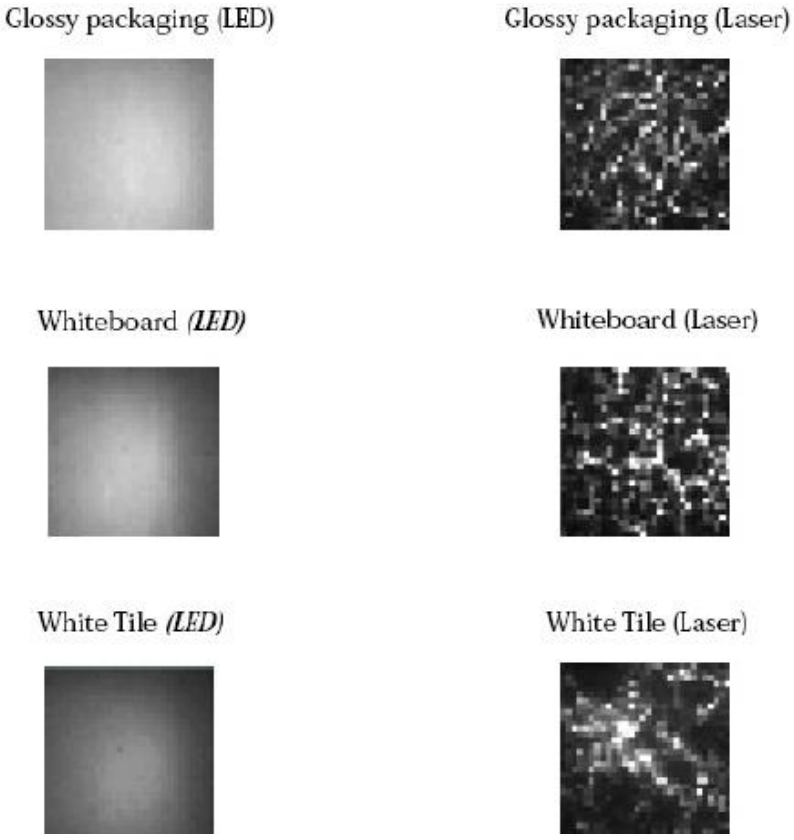
Obrázek 1-11 – Dva způsoby rozmístění detektoru záření, [22]

U laserových myší sensor přímo obsahuje VCSEL a celá součástka je tedy menší (viz Obrázek 1-12), zatímco u optických myší není zdroj světla (LED) součástí detektoru.



Obrázek 1-12 – ADNS-9500 s zaměřovacími čočkami, [10]

Obrázek 1-13 zobrazuje porovnání snímků povrchu pořízených senzorem, a to jak při osvětlení LED, tak při použití VCSEL. Z obrázků je patrné, že při použití laseru je vidět více detailů, než při použití červené LED na stejném povrchu. Díky tomu laserové senzory mohou pracovat na více površích.



Obrázek 1-13 – Snímky povrchu při osvětlení pomocí LED a laseru, [17]

1.2.4. Parametry senzoru

U senzorů se rozlišují různé parametry, díky kterým lze poznat, jak dobře bude senzor reagovat na pohyb měřeného objektu. Mezi základní parametry se řadí rozlišení, frekvence snímání, maximální možná rychlost a zrychlení měřeného povrchu.

- Rozlišení

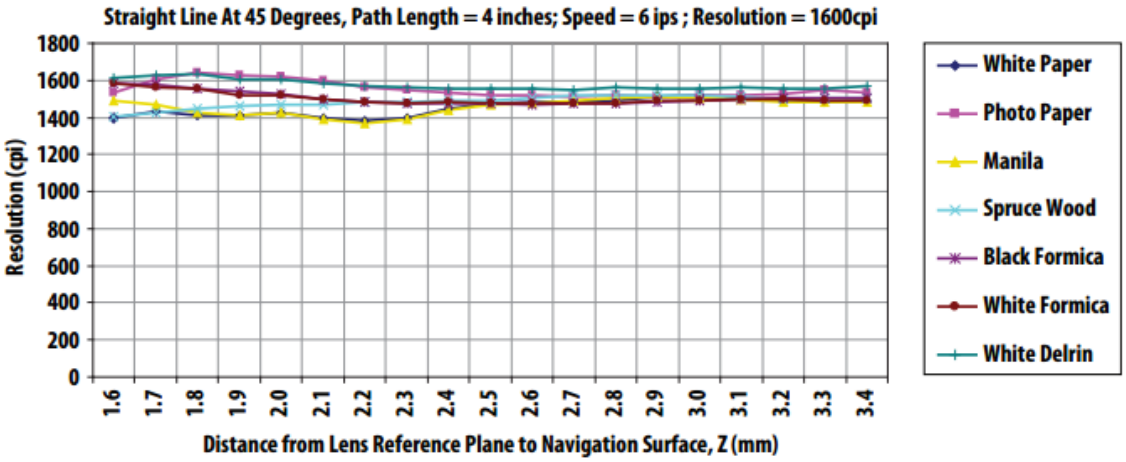
Nejvíce sledovaným parametrem je rozlišení senzoru. Uvádí se v tzv. „CPI“ (Counts Per Inch), neboli počet kroků, který senzor provede při posunutí o jeden palec. Jestliže je převod na kurzor myši bez modifikace, posune se právě o takový počet bodů (pixelů), jako udělá kroků. Pak by se CPI = DPI, což je zkratka “dots per inch“. Kvůli nárokům na přesnost či rychlost pohybu bývá u herních myší možnost nastavení DPI například od 200 do 12000. Někdy se právě termín DPI nesprávně zaměňuje za termín CPI. Čím vyšší CPI senzor má, tím více se pohne kurzor po obrazovce při stejném pohybu, má tedy větší senzitivitu. To ale může být řízeno softwarem, takže se kurzor může pohybovat víc nebo míň než je CPI. Dále bývá softwarově řízena tzv. akcelerace. To znamená, že po určitém čase se při konstantní rychlosti myši zvyšuje rychlost kurzoru.

Rozlišení může být z jednotek CPI přepočteno na vzdálenost, které odpovídá jeden krok, podle následujícího vzorce.

$$r = \frac{25{,}4}{CPI}\ \left[\frac{mm}{krok}\right]. \tag{3}$$

Konstanta 25,4 odpovídá převodu palců na milimetry (jeden palec je 25,4 mm). Například pro rozlišení 1600 CPI to odpovídá hodnotě 0,0159 mm/krok.

Rozlišení se u běžných myší pohybuje od 800 do 1600 CPI, u herních myší je to až 8200, např. senzor ADNS-9800 (nastavitelné s krokem 200). Obrázek 1-14 ukazuje závislosti rozlišení na vzdálenosti optické soustavy od povrchu měřeného systému pro různé typy povrchu. [17], [22]



Obrázek 1-14 – Graf závislosti rozlišení na vzdálenosti od povrchu, [10]

- Frekvence snímání (Frame rate)

Frekvence snímání je rychlost, s jakou senzor snímá obrázky povrchu. Udává se v počtech obrázků za sekundu. Například laserový senzor ADNS-9800 může snímat rychlostí až 12000 fps (frame per second). Pokud má optická myš nastavitelný parametr frame rate, může být zvolena různá hodnota uživatelem. U některých myší je tento parametr automaticky stanovovaný pro optimální výkon. [17], [11]

- Maximální rychlost povrchu

Dalším parametrem je maximální rychlost povrchu, jakou ještě dokáže senzor správně zaznamenat. Například senzor ADNS-2051 garantuje přesnost při 14 ips (inch per second), což je 355,6 mm/s. Herní senzor ADNS-9800 pracuje až do rychlosti 150 ips (3810 mm/s). [11], [1]

- Maximální zrychlení

U optických senzorů se uvádí maximální zrychlení, které je senzor ještě schopen správně zaznamenat. U herních senzorů bývá až 30 g, u senzoru ADNS-6000 je to jen 8 g. [11], [9]

2. Výběr optického senzoru

2.1. Požadavky

Pro vytvoření spolehlivého a přesného přístroje na měření rychlosti bylo v prvé řadě potřeba vybrat vhodný optický senzor. V našem případě byl přístroj navrhován do prostředí, ve kterém již nacházejí uplatnění i jiné přístroje pracující s optickými senzory. Vlnová délka elektromagnetického záření používaného pro osvětlení měřeného povrchu z tohoto důvodu musela být zvolena nejméně 600 nm, neboť při nesplnění této podmínky by mohlo dojít k ovlivňování měření.

Dále bylo potřeba vybrat senzor s dostatečně velkou maximální možnou rychlostí měřeného povrchu, abychom zaručili přesnost měření pro vyšší rychlosti. Vybírali jsme tedy senzory, které jsou schopny měřit rychlost povrchu až 2 m/s.

2.2. Vytipování vhodných senzorů

Protože povrch se může pohybovat až rychlostí 2 m/s, senzor musí být schopný registrovat minimálně tuto hodnotu. To odpovídá hodnotě 78,7 ips. Byl tedy vybírán senzor s minimální hodnotou 80 ips.

Požadavkům odpovídaly 4 nalezené optické senzory s LED a 2 optické senzory pracující s laserovým zdrojem světla VCSEL. U laserových senzorů ale bývá zdroj světla přímo na pouzdře, což by mohlo vést k problémům se zacílením světla na požadovanou plochu, která bude ve vzdálenosti mnohem větší, než k jaké je senzor původně určen. Jednou z možností vyřešení tohoto problému je použití náhradního zdroje záření a deaktivace původního laseru. Externí zdroj záření musí být vybrán s ohledem na citlivost detektoru na různé vlnové délky. Závislost citlivosti na vlnové délce senzoru ADNS-3080 a ADNS-9800 je znázorněna v kapitole 1.2.1z, viz Obrázek 1-6 a Obrázek 1-7. Následující tabulka slouží ke srovnání šesti odpovídajících senzorů.

Tabulka 2-1 – Srovnání vhodných senzorů, [11], [10], [28], [27], [26], [25]

Senzor	Maximální rychlost [ips]	Maximální zrychlení [g]	Frame rate [fps]	Rozlišení [cpi]	Zdroj záření
SDNS-3988	200	50	12500	6400	IR LED (HSDL-4261) – 870 nm
PMW-3360	250	50	12000	12000	IR LED
PMW-3320	80	20	5300	3500	IR LED (HSDL-4261) – 870 nm
PMW-3310	130	30	6500	5000	IR LED (HSDL-4261) – 870 nm
ADNS-9500	150	30	11750	5000	VCSEL 832-865 nm
ADNS-9800	150	30	12000	8200	VCSEL 832-865 nm

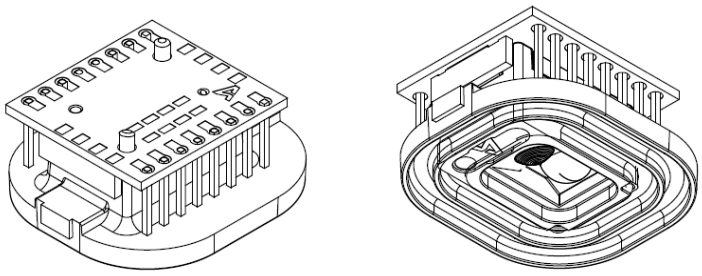
K vybranému senzoru bude dále potřeba instalovat vhodnou optickou soustavu, aby bylo možné senzor využít pro měření z větší vzdálenosti. Například senzor ADNS-3080 je dostupný s již instalovanou optickou soustavou o ohniskové vzdálenosti 4,2 mm. Přestože má senzor podstatně horší parametry, viz Tabulka 2-2, byl z důvodu testování tohoto sestavení do výběru také zahrnut.

Tabulka 2-2 – Parametry senzoru ADNS-3080, [2], [8]

Senzor	Maximální rychlost [ips]	Maximální zrychlení [g]	Frame rate [fps]	Rozlišení [cpi]	Zdroj záření
ADNS-3080	40	15	6400	1600	HLMP-ED80-XX000 – 630 nm

2.3. Výběr optické myši se vhodným senzorem

Pro první fázi testování byl nejprve vybrán senzor ADNS-9800 vzhledem k jeho dostačujícím parametrům a s přihlédnutím k ceně. Senzor byl zakoupen jako celek integrovaný do myši A4tech Bloody Terminator TL8, díky čemuž mohlo být využito propojení senzoru s DPS a správné vzdálenosti senzoru od měřeného objektu, protože z počátku byla využívána stejná optická soustava.



Obrázek 2-1 – ADNS-9800 s původní optickou soustavou

V další fázi práce, pro testování optických soustav, byl vybrán také senzor ADNS-3080. Jeho parametry jsou uvedeny v předchozí podkapitole 2.2. Byl zakoupen jako celek s již instalovanou optickou soustavou, umožňující zaostření na vzdálenější objekty. Tento senzor s instalovanou optickou soustavou je na obrázku 2-2.

Obrázek 2-2 – ADNS-3080 s optickou soustavou

3. Prototypový rychloměr využívající stávající konstrukci počítačové optické myši

Jeden z možných přístupů pro odhad rychlosti povrchu pomocí počítačové myši je výpočet pomocí kurzoru myši v PC. Přitom software v počítači převádí informace o pohybu na pohyb kurzoru a navržený program pak tento pohyb převede zpátky na data Δx a Δy. Tento postup je uveden v kapitole 3.1.

Vyhnout se zbytečnému převádění na pohyb kurzoru je možné při použití například elektronické platformy pro samotné řízení senzoru. Díky tomu není potřeba PC a je možné vytvořit autonomní zařízení z několika základních komponent. Tato práce je založena právě na tomto postupu, princip popisuje kapitola 3.2.

3.1. Rychloměr povrchu založený na vyčítání dat z počítačové myši

Počítačové myši posílají po sériové lince, nejčastěji přes USB, informace o změně pohybu. Počítačový software tato data transformuje na pohyb kurzoru a přímo k surovým datům uživatelé přístup nemají. Proto první myšlenkou, jak získat rychlost využitím počítačové myši, bylo přepočítat pozici kurzoru a jeho pohyb zpátky na pohyb myši.

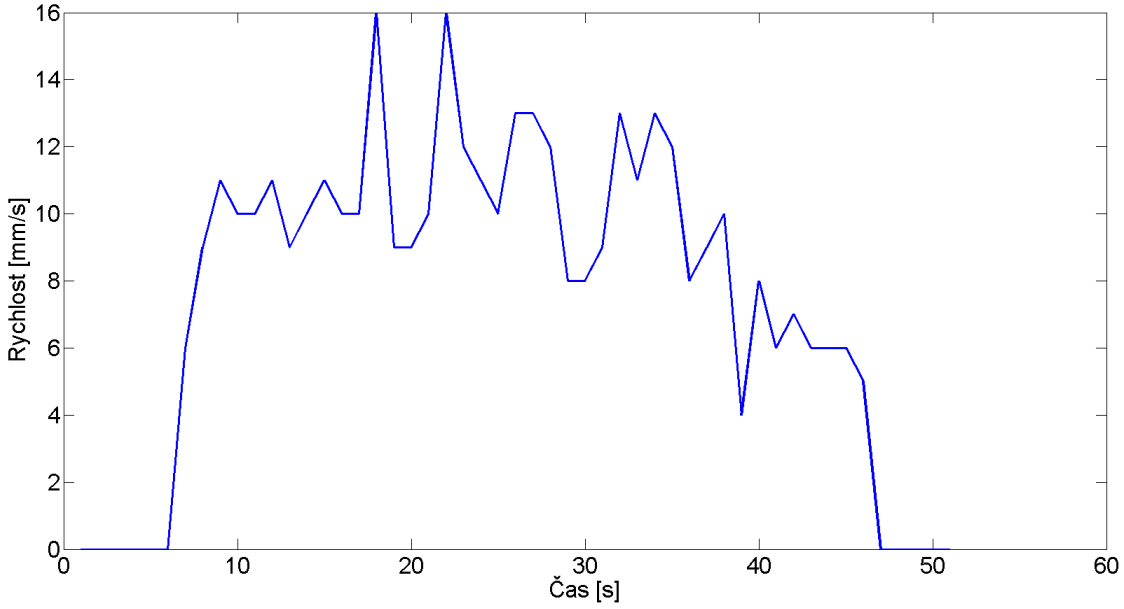
K tomuto účelu byl vytvořen jednoduchý program v prostředí MATLAB. Ten zjišťuje pozici kurzoru ve dvou okamžicích od sebe vzdálených přibližně 60 milisekund. Je vypočtena vzdálenost mezi těmito body a vydělena časem, který za tu dobu uběhl. Takto je získána rychlost kurzoru v pixelech za sekundu. Zde nastává hlavní problém tohoto způsobu měření rychlosti, neboť k přepočtu na milimetry za sekundu je nutné znát velikost monitoru v ose x a y. Pomocí této informace a získané informace o rozlišení monitoru je programem vypočítána velikost jednoho pixelu a vynásobením rychlosti touto velikostí získáme rychlost kurzoru v mm/s. Abychom získali rychlost myši, je nutné znát její rozlišení. Dosazením do vzorce

$$k = \frac{CPI}{PPI}, \tag{4}$$

kde *CPI* je rozlišení myši a *PPI* počet pixelů monitoru na jeden palec, získáme konstantu *k* potřebnou k přepočtu rychlosti kurzoru na rychlost myši:

$$v_m = \frac{v_k}{k}. \tag{5}$$

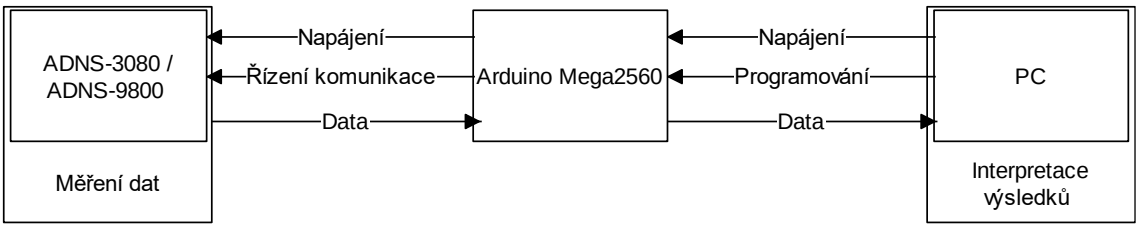
v_m značí rychlost myši, v_k rychlost kurzoru. Získali jsme tedy rychlost myši v ose x a y, případně můžeme vypočítat celkovou rychlost. Omezení tohoto způsobu měření rychlosti myši je v omezení pohybu kurzoru velikostí obrazovky, takže po dosažení kraje obrazovky pohyb kurzoru ustane, přestože myš se dále pohybuje. Prioritní funkce v daném operačním systému je pohyb kurzoru, a až poté se z této informace počítá odhad vzájemného pohybu myši a povrchu pod ní. Další nevýhodou je potřeba počítače, což znamená sníženou mobilitu a vyšší nákladnost. Potřeba příslušných programů, v tomto případě MATLAB, přináší další nemalé zvýšení nákladů. Obrázek 3-1 ukazuje záznam měření rychlosti vypočtením z pozice kurzoru při pohybu po uhlopříčce monitoru. Myš byla tažena rukou, což vysvětluje kolísání hodnot aktuální rychlosti. Skript k tomuto měření rychlosti je uveden v příloze A.



Obrázek 3-1 – Záznam měření rychlosti ze změny pozice kurzoru

3.2. Rychloměr povrchu využívající stávající konstrukci počítačové myši

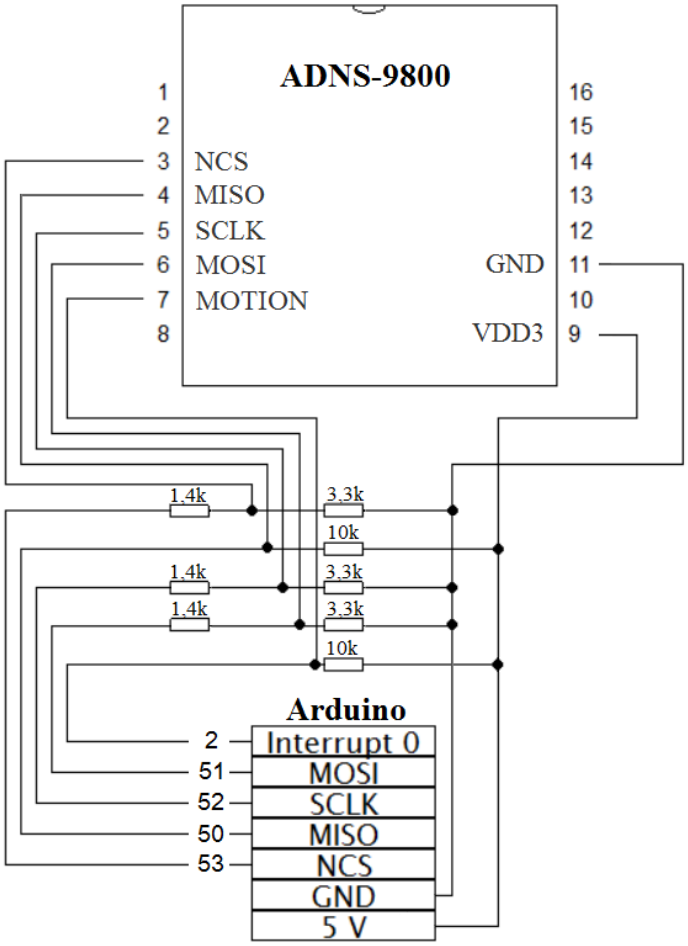
Hlavní procesor počítačové myši založené na senzoru ADNS-9800 je nutné zastoupit elektronickou platformou, programovatelnou tak, aby sloužila ke komunikaci se senzorem, a bylo z ní možné získat přímo surová data nesoucí informaci o pohybu. Podobně u senzoru ADNS-3080 bylo nutné použít platformu pro napájení a programování tohoto senzoru, popis se nachází v kapitole 5.2. Pro tento účel byla vybrána elektronická platforma Arduino Mega2560. Jde o open-source projekt, díky čemuž je cena tohoto zařízení poměrně nízká. Tato platforma byla programována pomocí vývojového prostředí Arduino IDE. K zobrazení výsledků sloužil program MATLAB. Experimentální pracoviště sestávalo z valníku, který se pod senzorem posouval konstantními rychlostmi. Tomuto způsobu měření rychlosti jsou věnovány následující dvě podkapitoly.



Obrázek 3-2 – Blokové schéma funkcí jednotlivých prvků

3.2.1. Hardware

Vybraný senzor ADNS-9800 byl připojen k platformě Arduino Mega2560, která zajišťovala napájení senzoru napětím 5 V a sloužila jako master při komunikaci se senzorem. Hlavní částí desky Arduino Mega2560 je procesor ATmega2560 od firmy Atmel Corporation. Deska obsahuje převodník mezi USB a komunikační linkou Arduina umožňující komunikaci mezi PC a čipem a disponuje 54 digitálními piny, sloužícími jako vstup nebo výstup, a 16 analogovými vstupy. Frekvence vnitřního oscilátoru procesoru je 16 MHz. Deska je napájena pomocí USB nebo je možné připojit externí zdroj napájení. Pracovní napětí desky je 5 V. Senzor ADNS-9800 může pracovat ve 3 V režimu, nebo 5 V režimu. Napájení senzoru bylo nastaveno na 5 V. SPI komunikace však probíhala na 3 V, proto byly při propojování desky a senzoru použity děliče napětí, které napětí snížily na požadovanou hodnotu. Obrázek 3-3 znázorňuje propojení senzoru ADNS-9800 a desky Arduino Mega2560. [31], [3]



Obrázek 3-3 – Schéma zapojení ADNS-9800 do desky Arduino Mega2560

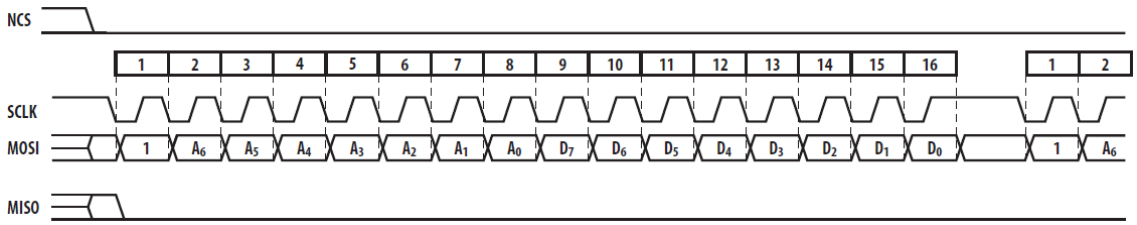
Některé piny na desce Arduino Mega2560 mají specializované funkce. Například pin 2, využitý dále v programu ve funkci *UpdatePointer*, má funkci externího interruptu 0. Na tento pin byl připojen pin MOTION senzoru ADNS-9800. Pokud senzor zaznamená pohyb, je tento pin sepnut na 0, což znamená, že je aktivní. Dalšími specifickými piny na desce Arduino jsou piny 50 (MISO), 51 (MOSI), 52 (SCLK) a 53 (NCS), které slouží ke komunikaci mezi senzorem a deskou. MISO (master in, slave out) slouží k přenosu dat, kde řídící zařízení (master) je vždy vstup a podřízené zařízení (slave) je výstup. MOSI (master out, slave in) funguje právě naopak. SCLK značí hodinový signál, podle kterého je komunikace řízena. NCS (chip select) neboli SS (slave select) definuje zařízení, u kterého má komunikace probíhat. Obsahuje nulovou hodnotu, pokud je aktivní sériový port, tedy připojené zařízení může komunikovat. Tyto piny byly využity pro komunikaci a připojeny na odpovídající piny na senzoru. Na desce Arduino jsou dále piny sloužící jako napájení pro připojená zařízení, použité byly piny 5 V a GND. Na senzoru jsou při práci v 3 V režimu analogová a digitální zem spojeny, proto je připojena k desce pouze analogová zem. [3], [11]

3.2.2. Software pro stanovení rychlosti povrchu

V programovacím prostředí Arduino IDE byl vytvořen program, sloužící k nastavení komunikace mezi senzorem ADNS-9800 a deskou Arduino MEGA 2560 a k posílání informací po sériové lince. Pro zpracování informací ze senzoru a zobrazení výsledků byl použit program MATLAB.

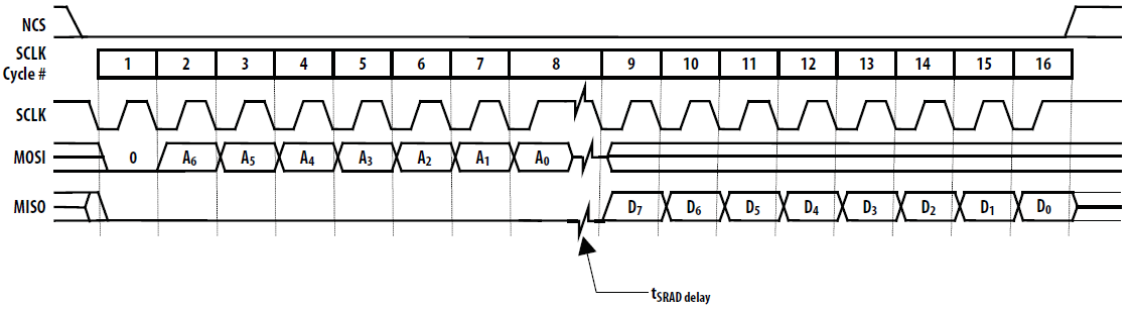
Komunikace se senzorem probíhá pod kontrolou mikroprocesoru Arduina, který ji řídí pomocí NCS a SCLK. Informace je ze senzoru možné přečíst z registrů. Pro správnou funkci senzoru je potřeba do registrů také zapisovat, následující dva odstavce proto budou věnovány těmto operacím.

Operace zápisu je definována jako poslání dat z mikroprocesoru do senzoru, je řízena mikroprocesorem a tvoří ji dva bajty. Sedm bitů z prvního bajtu tvoří adresa registru, do kterého se mají data zapsat, a MSB (Most Significant Bit, nejvýznamnější bit), první bit prvního bajtu, má hodnotu 1, která indikuje směr toku dat. Druhý bajt obsahuje data. Senzor ADNS-9800 čte příchozí data (MOSI) na nástupných hranách SCLK. [11]



Obrázek 3-4 – Schéma SPI při zápisu, [11]

Operace čtení je definována jako posílání dat ze senzoru do mikroprocesoru. Je tvořena dvěma bajty a je vždy zahájena mikroprocesorem, který pošle adresu registru do senzoru přes MOSI spolu s MSB nastaveným na 0. Druhý bajt obsahuje požadovaná data a je řízen senzorem pomocí MISO, který posílá bity se sestupnou hranou SCLK. Po odeslání posledního bitu adresy a před přečtením prvního bitu dat je třeba dodržet zpoždění t_{SRAD}, které je v případě ADNS-9800 100 μs, aby senzor připravil data. [11]



Obrázek 3-5 – Schéma SPI při čtení, [11]

Mezi operacemi čtení a operacemi zápisu je také nutné dodržovat zdržení. U zmíněného senzoru po operaci zápisu 120 µs (t_{SWR}, t_{SWW}) a po operaci čtení 20 µs (t_{SRR}, t_{SRW}). [11]

Arduino

Pro získání informace o pohybu jsou možné dva přístupy. První přístup spočívá v postupné komunikaci s optickým senzorem přes SPI, kdy jsou jednotlivé parametry pohybu vyčítány na vyžádání. Druhý přístup spočívá ve využití tzv. „Burst mode“, kdy na jedno vyžádání jsou předány všechny parametry pohybu. Na prvním přístupu je možné lépe objasnit, jak senzor funguje. Je však složitější, a proto není tak rychlý jako druhý přístup, na jehož principu výsledné zařízení funguje.

Program nahraný do desky Arduino byl navržen tak, aby při zaznamenání pohybu senzorem došlo k přečtení informace o změně pohybu z dané adresy. Kód musí vždy obsahovat dvě základní funkce, *setup()* a *loop()*. Mimo ně byly definovány další funkce, které slouží například ke čtení z registrů nebo naopak k zápisu do nich, nebo funkce, která je závislá na stavu pinu 2 a spouští se vždy se sestupnou hranou. Jde tedy o interrupt. Je zde použit ještě časový interrupt, což je časovač, který běží nezávisle na hlavním programu a udává přesné časové impulzy. Tento časovač je využit k integraci a vypisování pohybu. Výsledek je poslán sériovou linkou do PC. V následující části je kód popsán, celý kód je přiložen v příloze této práce: B.

- Funkce *setup()*

Tato funkce je základní funkce kódu, která proběhne vždy po zapnutí či restartu desky pouze jednou a je zde definováno nastavení. V našem případě je první částí této funkce časovač (TIMER3), s požadavkem na periodu 0,1 s, který je využíván pro integraci dat pohybu po zvolenou periodu. Nastavení periody je zde pomocí OCR3A podle vzorce

$$OCR3A = \frac{t}{r} - 1, \tag{6}$$

kde *t* je požadovaný čas, tedy 100 ms, a *r* je rozlišení časovače, které vypočítáme pomocí vzorce

$$r = \left(\frac{c}{n}\right)^{-1}, \tag{7}$$

kde *c* reprezentuje frekvenci procesoru (clock) a *n* nastavení dělení časovače pomocí bitů CS30, CS31 a CS32 na adrese TCCR3B, což je při našem nastavení 100b, tedy dělíme číslem 256 (viz Tabulka 2-1). Frekvence *c* je u procesoru ATmega2560 16 MHz. Odečítání jedničky v rovnici (6) je zde kvůli toho, že po dosažení požadovaného počtu se časovač vynuluje, což zabere právě jeden krok. Tedy abychom dostali požadovaný čas, tento krok musíme vzít v úvahu a odečíst jeden krok, který je tímto krokem potřebným pro reset nahrazen. Po dosazení do vzorců zjistíme požadovanou hodnotu OCR3A = 6249. [6], [31]

Tabulka 3-1 – Nastavení bitů 2,1,0 registru TCCR3B, ATmega2560, [6]

CS32	CS31	CS30	Popis
0	0	0	Časovač vypnut
0	0	1	$clk_{I/O}/1$
0	1	0	$clk_{I/O}/8$
0	1	1	$clk_{I/O}/64$
1	0	0	$clk_{I/O}/256$
1	0	1	$clk_{I/O}/1024$
1	1	0	Externí zdroj na pinu Tn, s padající hranou
1	1	1	Externí zdroj na pinu Tn, s rostoucí hranou

Dalšími kroky jsou nastavení komunikace, nastavení vstupních pinů a připojení funkce *UpdatePointer*(), která se spouští vždy, když na pinu 2 klesne stav z 1 na 0. Tento přechod značí, že byl senzorem detekován pohyb a funkce pak informace o pohybu načte do proměnné *xydat*.

Posledním krokem funkce *setup()* je zavolání funkce *performStartup()*. Tato funkce zajišťuje správný start senzoru, nahrání firmware, nastavení laseru do normálního módu a nastavení rozlišení senzoru. Rozlišení lze měnit zápisem do registru Configuration_I. Přednastavená hodnota je 1800 cpi, kdy je v registru zapsána hodnota 0x09. Nejnižší rozlišení 200 cpi je dosaženo při nastavení registru na 0x01, kdy můžeme snímat i velmi rychlé pohyby, zatímco při nejvyšším rozlišení 8200 cpi (0x29) je senzor citlivý i na velmi malý pohyb.

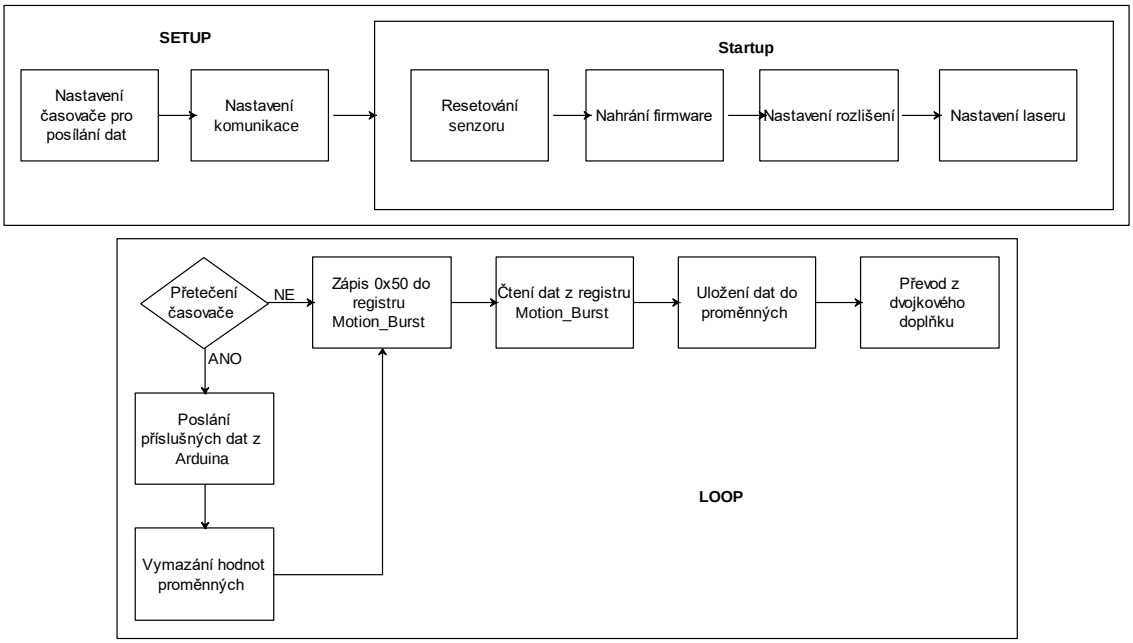
Na konci této funkce se do proměnné *initComplete* zapíše hodnota 1, což je vstupní podmínka funkce *UpdatePionter()*, aby tato funkce neprobíhala před nastavením senzoru.

- Funkce *loop()*

Funkce *loop()* má dvě části, dvě podmínky. První probíhá každých 100 ms, v závislosti na výše zmíněném časovači TIMER3. Každých 100 ms je vypsána informace o pohybu v ose x a y za právě tuto dobu.

Druhá podmínka testuje, zda senzor detekoval pohyb. Pokud ano, načte jej do proměnné *dX* a *dY* a tyto hodnoty přičítá do proměnné *deltaX* a *deltaY* po dobu, než dojde k proběhnutí první podmínky, kde jsou hodnoty poslány na výstup a následně vymazány.

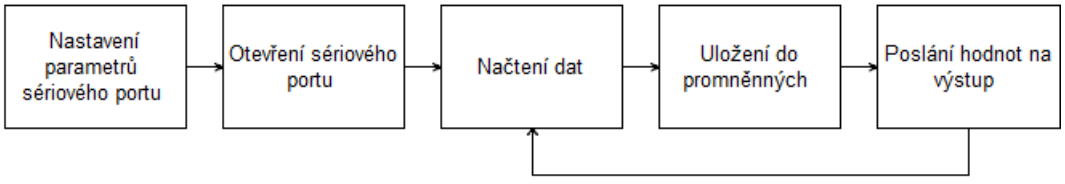
Jak bylo zmíněno na začátku této podkapitoly, existuje ještě druhý přístup pro získání informace o pohybu, který je jednodušší a rychlejší. Při tomto způsobu není potřeba připojení pinu Motion, neboť data jsou senzorem poslána kdykoliv, je-li aktivován Burst mód. Tento mód je aktivován posláním 0x50 na adresu registru Motion_Burst. Následně je dostupných čtrnáct bajtů dat pro přečtení, z nichž jsou pro nás nejdůležitější Delta_X_L, Delta_X_H, Delta_Y_L, Delta_Y_H a SQUAL. Po přečtení těchto dat je nutné zvýšit napětí na NCS alespoň na t_{BEXIT} (500 ns) a poté je možné proces opakovat. Vývojový diagram dvou hlavních funkcí je na následujícím obrázku. Funkce *setup()* je velmi podobná jako v minulém případě. Jsou zde definovány parametry komunikace a časovače a nahrán firmware. Ve funkci *loop()*, která probíhá v neustálé smyčce, je nejdříve testována podmínka, zda přetekl nastavený časovač. Pokud ano, jsou z Arduina poslána data nesoucí informaci o pohybu a kvalitě povrchu. Následně jsou hodnoty vymazány a pokračuje funkce *loop()* stejně, jako by podmínka neplatila. Je aktivován Burst mód a načtena požadovaná data, která se ukládají do proměnných, dokud opět nepřeteče časovač. Ten je v našem případě nastaven na 100 ms, je však možné jej nastavit na jinou hodnotu. Současně s posláním dat po sériové lince je také vypočtená aktuální rychlost posílána na připojený OLED displej. Na začátku programu musí být připojena knihovna umožňující komunikovat s displejem. Ve funkci *loop(),* konkrétně pokud platí podmínka pro splnění časovače, je zvolen font písma a pozice, na které se mají data zobrazit. Pomocí funkce *OLED.print* jsou data na displej poslána. Zdrojový kód tohoto programu je v příloze G.



Obrázek 3-6 – Vývojový diagram dvou hlavních funkcí programu

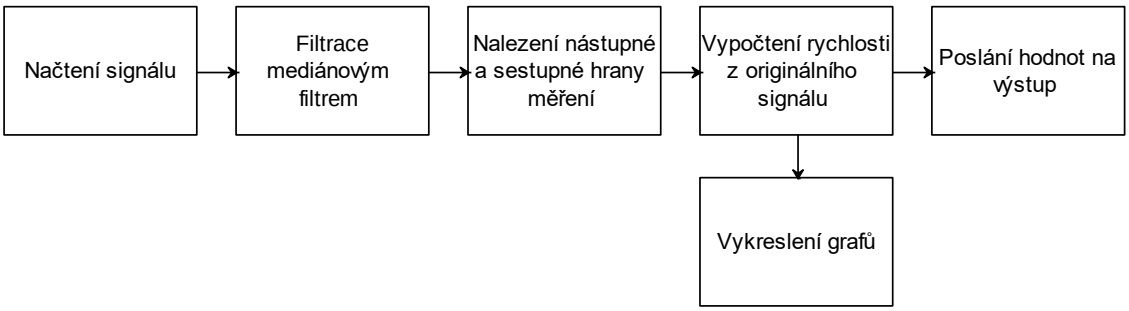
Matlab

Skript *nacitani.m* vytvořený v programu MATLAB slouží k zobrazení informací o pohybu a k uložení dat. Z desky Arduino jsou informace posílány sériovou komunikací. Ta je otevřena programem MATLAB a data jsou funkcí *fscanf* načítána. Z desky Arduino jsou posílána jako pohyb v ose x a pohyb v ose y vždy s oddělením čárkou. Načtená data je tedy nejdříve potřeba oddělit. K tomu slouží funkce *findstr*, která čárku najde a do proměnné *x* uloží data před čárkou a do proměnné *y* data za ní. Pokud při posílání dat z desky dojde k chybě a informace o pohybu senzoru v MATLABu není načtena, uloží do výsledného vektoru hodnotu 1111. Jinak jsou do výsledného vektoru uložena pohybová data. Aktuální data jsou funkcí *fprintf* vypsána v příkazovém řádku. Celková informace o pohybu je uložena ve výsledných vektorech *x1* a *y1*, které mohou být z pracovního prostoru uloženy a zpracovány pomocí skriptu *zpracovani.m*.



Obrázek 3-7 – Blokové schéma skriptu nacitani.m

Skript *zpracovani.m* načte uložená data *x1* a *y1*, ze kterých vypočte průměrnou rychlost povrchu pod senzorem. Pro zpracování dat je nezbytné je nejdříve filtrovat. Možné chybné vzorky, kterým je přiřazena hodnota 1111, jsou odstraněny a pro experimentální měření nepotřebné záporné hodnoty jsou potlačeny. Dále je nutná filtrace signálu mediánovým filtrem, aby bylo možné správně určit, kdy byl započat a ukončen pohyb senzoru. V signálu jsou poté nalezeny celistvé úseky a jejich pozice je zapsána do vektoru *ii*. Pomocí funkce *diff*, která počítá rozdíly mezi sousedními prvky, jsou pak nalezeny nástupné a sestupné hrany, tedy začátky a konce pohybu. Pro eliminaci chybných měření na začátku a na konci každého měření nejsou tyto vzorky pro výpočet používány. Přímo pro výpočet rychlosti je však použit nefiltrovaný signál. Součtem velikostí impulzů za jedno měření v ose x nebo y je získána celková vzdálenost uražená za dané měření v dané ose v jednotkách bodů závislých na rozlišení senzoru. Použitím Pythagorova zákona je z jednotlivých složek pohybu vypočtena celková vzdálenost v rovině xy. Pro přepočet vzdálenosti na jednotky mm je vzdálenost násobena 25,4 (přepočet z palců) a dělena rozlišením senzoru. Průměrná rychlost povrchu pod senzorem je získána vydělením vzdálenosti časem, po který měření probíhalo. Čas v sekundách je roven počtu impulsů děleným deseti, jelikož jeden impuls trval 0,1 s. Výpočet průměrné rychlosti je prováděn v cyklu opakujícím se pro počet proběhlých měření a výsledkem je tedy průměrná rychlost za všechna jednotlivá měření. Kód pro načítání dat ze sériové linky i kód pro zpracování jsou uvedeny v příloze C, D.



Obrázek 3-8 – Blokové schéma skriptu zpracovani.m

3.2.3. Software pro získání obrazu ze senzoru

Jak je vysvětleno v první části této práce, použitý optický senzor využívá ke zjištění pohybu snímky povrchu sejmuté velmi krátce po sobě, které mezi sebou porovná. Obrazy snímané senzorem ADNS-9800 mají velikost 1x1 mm, 30x30 pixelů a každý pixel znázorňuje stupeň šedi výsledného obrazu. V zařízení je možné tento snímek ze senzoru získat a použít pro zaostření optické soustavy detektoru na monitorovaný pohybující se povrch.

Arduino

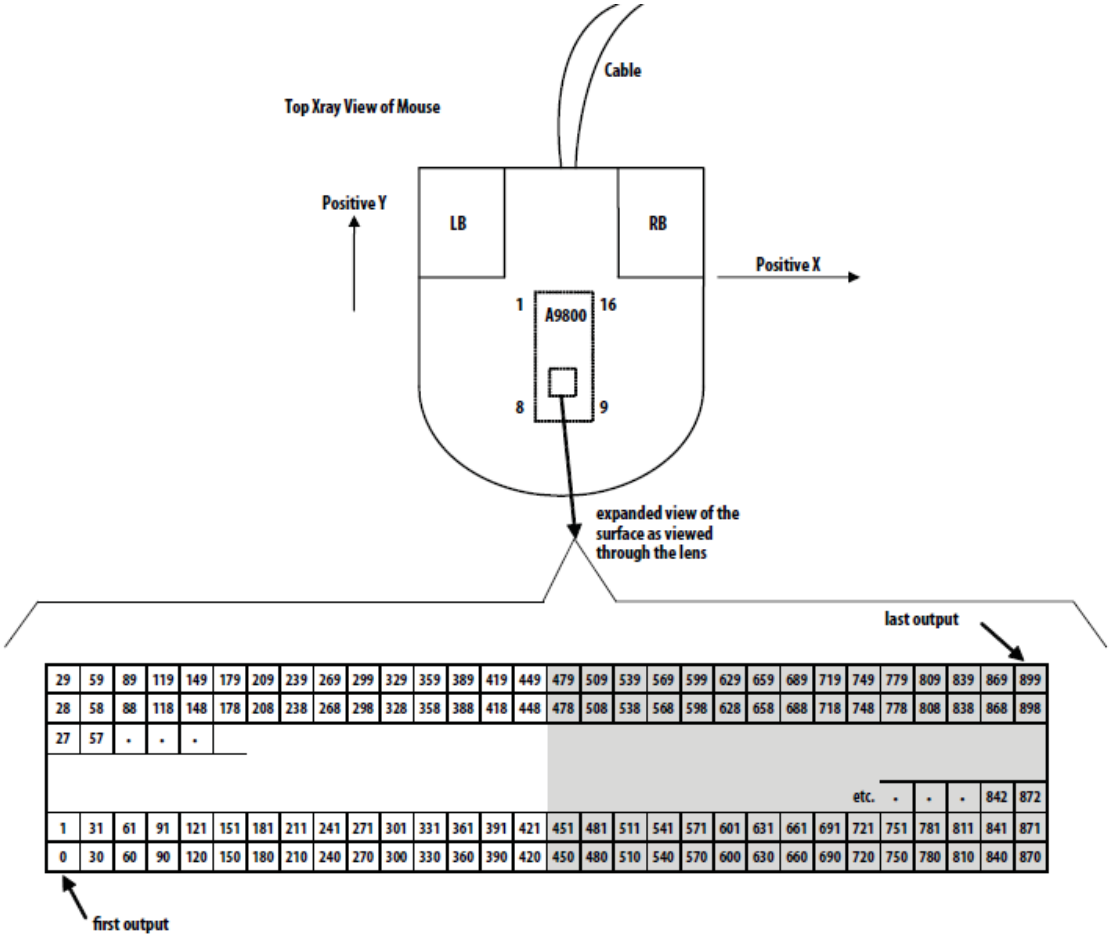
Kód pro získání devíti set pixelů požadovaného obrazu snímaného senzorem ADNS-9800 je z části stejný, jako kód pro měření rychlosti popsaný výše. Ve funkci *setup()* jsou nastaveny parametry komunikace a restartován senzor pro správnou funkci. Funkce loop(), která probíhá neustále znovu, obsahuje kód pro získání obrazu zahrnující tyto kroky 3 až 7:

1. Resetovat čip zapsáním 0x5a do registru Power_Up_Reset
2. Povolit laser nastavením bitu Force_Disabled v registru LASER_CTRL na 0
3. Zapsat 0x93 do registru Frame_Capture
4. Zapsat 0xc5 do registru Frame_Capture
5. Počkat dva rámce
6. Zkontrolovat, zda je dostupný první pixel přečtením bitu 0 registru Motion, pokud je 1, pak je první pixel dostupný
7. Číst registr Pixel_Burst, dokud není načteno všech 900 pixelů

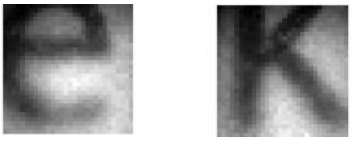
První a druhý krok je proveden ve funkci *startup()*. Před krokem 3 je změněn stav na pinu NCS na logickou 0 pro aktivaci komunikace a po celý průběh této funkce se nemění. Přečtené hodnoty pixelů z registru Pixel_Burst jsou pomocí *Serial.print()* poslány do PC. Pro rozlišení prvního pixelu obrazu je však nejprve odeslána hodnota 10000, která je v MATLABu detekována. Kód je uveden v příloze E.

Matlab

Pro získání zpracování informací o obraze a pro jeho vykreslení je použit program MATLAB. Nejdříve jsou nastaveny parametry sériového portu a následně je otevřen. Opakovaně jsou poté načítána data příkazem *fscanf* a je hledána hodnota 10000. Pokud je nalezena, je načteno následujících 900 pixelů obrazu. Protože číslování pixelů neodpovídá standartnímu zobrazení obrázku v MATLABu (viz Obrázek 3-9), je matice obrazu rotována o 270° proti směru hodinových ručiček. Ukázka získaných snímků je na obrázku na konci strany. (Obrázek 3-10). Skript je uveden v příloze F.



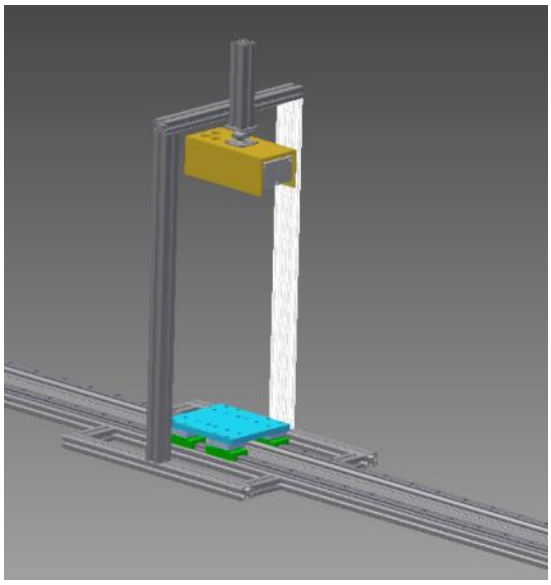
Obrázek 3-9 – Pixelová mapa snímaného obrazu senzorem ADNS-9800, [11]



Obrázek 3-10 – Obrázky získané senzorem ADNS-9800

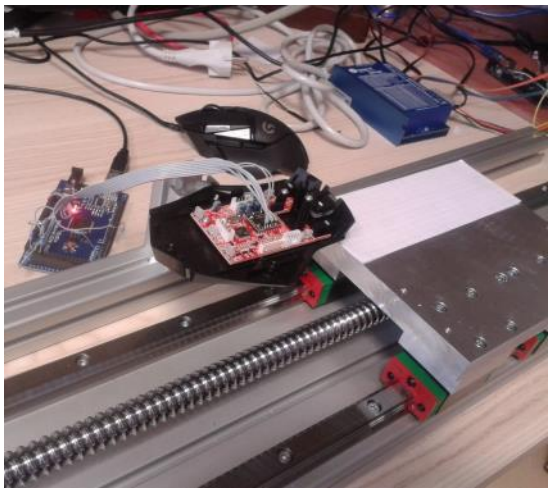
4. Předběžné experimentální měření rychlosti

Pro experimentální měření s vybraným senzorem byl použit lineární posuv, jenž je zobrazen na následujících obrázcích (Obrázek 4-1, Obrázek 4-2**)**. Pohyb zajišťoval krokový dvoufázový motor 57HS09 s točivým momentem 1,3 Nm, který otáčel kuličkovým šroubem o délce 1,5 m, průměru 16 mm a stoupání 5 mm. Na matku kuličkového šroubu byl umístěn vozík, který byl veden pomocí dvou rovnoběžných kolejnic. Nad vozík byly postupně upevněny senzory pro stanovení rychlosti objektu umístěného na vozíku.



Obrázek 4-1 – Model valníku s instalovaným rychloměrem

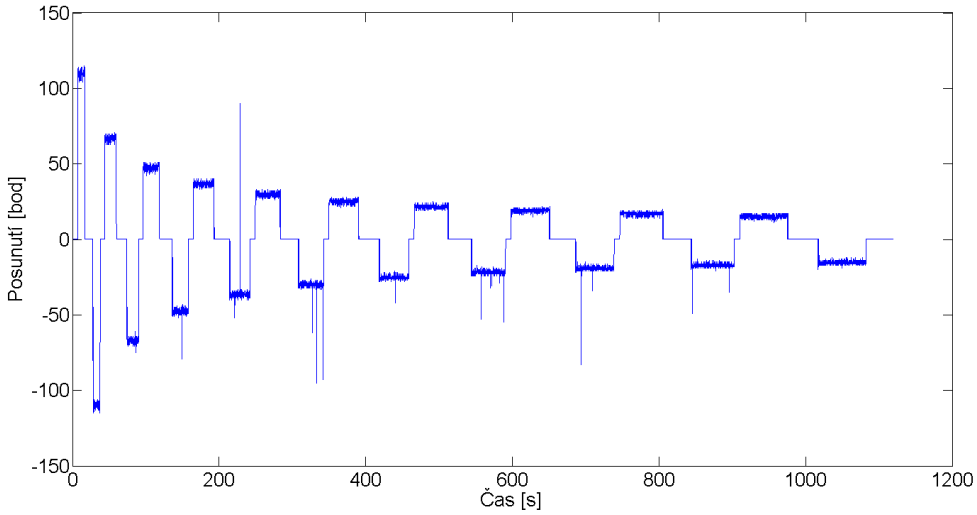
Vybraný senzor ADNS-9800 byl připojen k platformě Arduino MEGA 2560, která byla naprogramována tak, aby komunikoval s deskou Arduino. Požadované informace o pohybu byly posílány pomocí sériové komunikace do PC, kde byly zobrazeny a uloženy pomocí programu MATLAB. Podrobný popis byl zmíněn výše v kapitole 3.



Obrázek 4-2 – Valník s vybraným detektorem

4.1. Předběžné experimentální měření

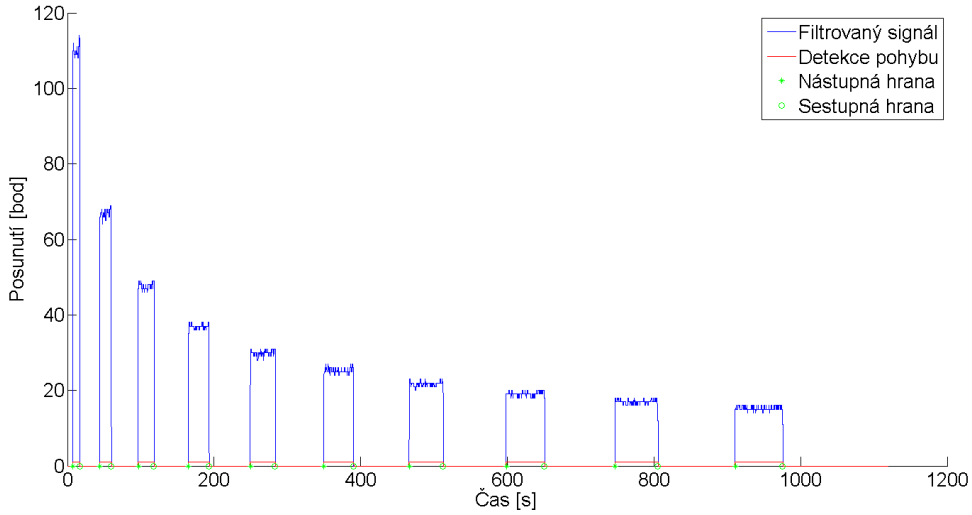
Bylo provedeno deset měření s různými rychlostmi valníku. Rychlost valníku v jednom měření (cesta tam a zpět) byla vždy stejná. Každé měření reprezentovalo pohyb valníku o identickou vzdálenost s rozdílnou rychlostí a směrem. Tento pohyb byl snímán senzorem, graficky znázorněný záznam měření v ose *y* poskytuje Obrázek 4-3. Při snímání z jednoho směru se však objevovaly chyby měření a pro výpočet rychlosti byl využíván vždy jen jeden směr. Snímaná data znamenají počet bodů, o které se povrch posunul za jeden krok. Jeden krok byl v našem případě 100 ms, dá se ovšem měnit v kódu nahrávaném do desky Arduino. Velikost bodu závisí na použitém rozlišení senzoru, v tomto případě bylo použito standardní rozlišení 1800 cpi. Tato data byla použita pro výpočet rychlosti. Valník se pohyboval převážně v ose *y* senzoru ADNS-9800, avšak z části také v ose *x*, a proto bylo potřeba brát v úvahu také tyto hodnoty.



Obrázek 4-3 – Záznam měření v ose y

4.2. Zpracování naměřených dat

Skript *zpracovani.m* byl použit k načtení naměřených dat x_1 a y_1, která byla uložena pomocí skriptu *nacitani.m.* Při měření mohl nastat případ, kdy se místo správné hodnoty uložila hodnota 1111, která indikuje chybu, a proto bylo nutné tyto možné chyby eliminovat. Pro určení přesného začátku a konce každého úseku bylo nejdříve potřeba surová data filtrovat. K tomu byl použit mediánový filtr, který má pro tento případ důležitou vlastnost zachovávání hran. Také byly potlačeny pro toto měření nepotřebné záporné hodnoty a z takto upraveného záznamu byly získány pozice začátků a konců měření. Sečtením impulzů v jednotlivých měřeních jsme získali informaci o pohybu v osách x a y. Za použití Pythagorovy věty byl získán celkový pohyb v rovině. Po vydělení tohoto pohybu dobou jeho trvání byla získána průměrná rychlost, která byla porovnána s rychlostí valníku.



Obrázek 4-4 – Nalezení počátků a konců jednotlivých měření

Rychlost valníku byla spočítána pomocí údajů o počtu kroků motoru, stoupání závitu šroubu a pomocí času, po který se valník posouval. Tabulka 4-1 uvádí tyto údaje a rychlost valníku spolu s naměřenými rychlostmi.

4.3. Vyhodnocení

Tabulka 4-1 – Vypočtená rychlost valníku a rychlost valníku naměřená senzorem

Měření č.	Krok	Čas [µs]	Počet kroků	Rychlost valníku [mm/s]	Naměřená rychlost [mm/s]	Naměřená rychlost 2 [mm/s]	Relativní chyba měření [%]
1	99	12452	78020	15,664	15,674	13,997	0,06
2	98	20307	78040	9,608	9,487	9,283	1,26
3	97	28166	78060	6,929	6,750	6,849	2,58
4	96	36030	78080	5,418	5,223	5,272	3,60
5	95	43897	78100	4,448	4,216	4,293	5,22
6	94	51767	78120	3,773	3,564	3,638	5,54
7	93	59644	78140	3,275	3,072	3,101	6,20
8	92	67522	78160	2,894	2,675	2,707	7,57
9	91	75405	78180	2,592	2,376	2,421	8,33
10	90	83293	78200	2,347	2,137	2,148	8,95

Vypočtená rychlost valníku a naměřená rychlost (viz Tabulka 4-1, 6. sloupec) se liší pouze mírně. Relativní chyba měření se nachází v posledním sloupci tabulky. Tato odchylka mohla být způsobena špatnou polohou senzoru. Senzor musí být v přesně

definované vzdálenosti nad měřeným objektem, a to 2,4 mm. Při špatném nastavení vzdálenosti mohla vzniknout chyba měření.

Z experimentálních důvodů bylo měření opakováno s posunutím senzoru o 45°, takže byl pohyb snímán v osách x a y se stejnou vahou. Takto naměřená rychlost je uvedena v předposledním sloupci tabulky (Tabulka 4-1). Toto měření potvrdilo správnost výpočtu rychlosti v rovině. Odchylky mohly být způsobeny nežádoucím pohybem senzoru v průběhu měření, jelikož nebylo dosaženo stejné kvality upevnění jako v prvním měření.

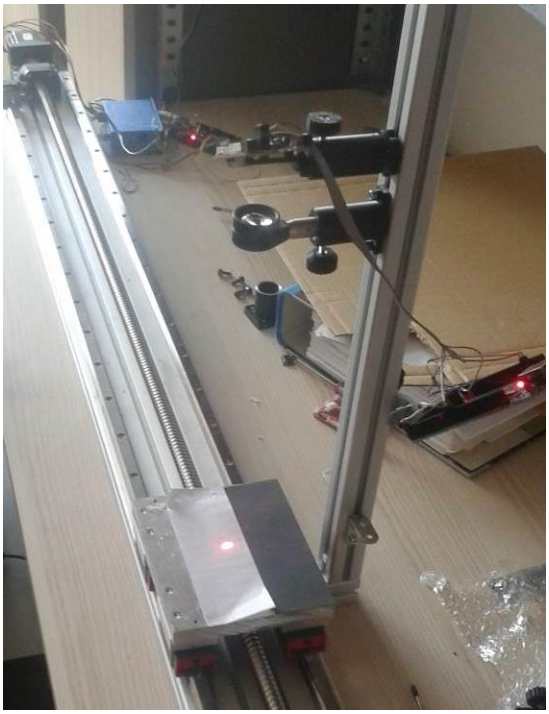
5. Instalace optické soustavy

U vybraného senzoru ADNS-9800 bylo doposud využíváno optické soustavy používané při aplikaci senzoru v počítačových myších. Tato optická soustava však umožňovala měřit pouze ze vzdálenosti několika milimetrů. Pro splnění cíle práce, tedy sestrojení zařízení pro měření ze vzdálenosti několika decimetrů, bylo dalším krokem použití a testování jiných optických soustav.

Při použití původní optické soustavy instalované k detektoru bylo využíváno pro osvětlování měřeného povrchu laseru zabudovaného přímo v pouzdře senzoru. Pro měření z větší vzdálenosti však tento zdroj s výkonem do 716 µW není dostatečný, proto bylo využito externího zdroje. Byl použit 5mW laser o vlnové délce 650 nm. Na tuto vlnovou délku má senzor ADNS-9800 relativní odezvu cca 90 %. Laser byl instalován do krytu zařízení (viz kapitola 6) pod úhlem cca 75° tak, aby bylo možné snímat osvětlený povrch ve vzdálenosti 30 cm, napájení bylo přivedeno z desky Arduino. [11]

5.1. ADNS-9800

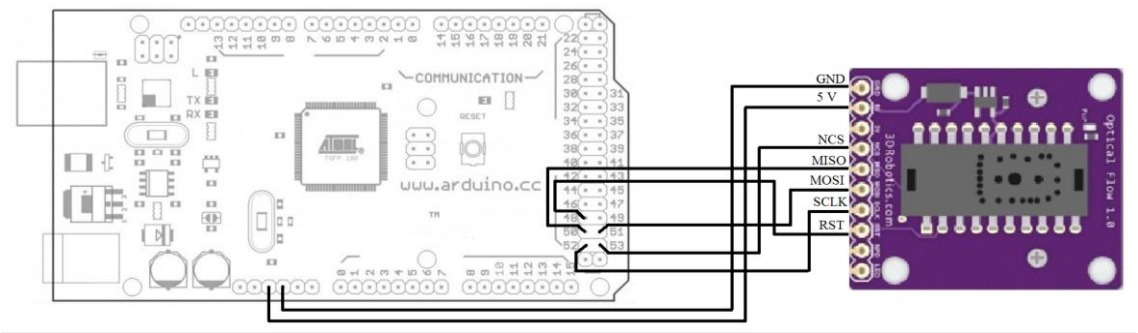
Měření pomocí senzoru ADNS-9800 bylo testováno při použití čoček s ohniskovou vzdáleností 35, 50 a 100 mm. Při použití čočky s ohniskovou vzdáleností 100 mm bylo dosaženo dobré reakce senzoru na pohyb povrchu ve vzdálenosti cca 40 až 60 cm. Použitá čočka však musela být od senzoru ve vzdálenosti cca 12,5 až 17,5 cm, což by přinášelo nevýhodu při konstrukci zařízení jako celku, neboť by bylo rozměrné. Z tohoto hlediska se mnohem více osvědčily čočky s ohniskovou vzdáleností 35 a 50 mm, u kterých byla čočka od senzoru umístěna ve vzdálenosti do deseti centimetrů. Pro další měření byla vybrána čočka s ohniskovou vzdáleností 50 mm.



Obrázek 5-1 – Testování čočky s ohniskovou vzdáleností 35 mm

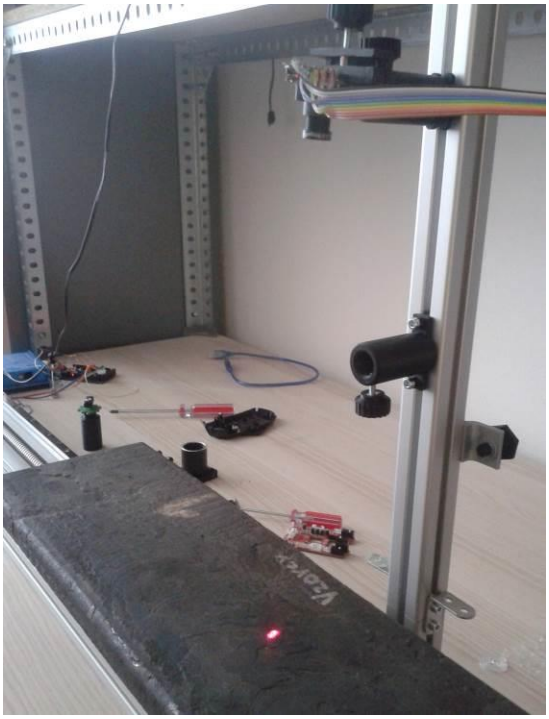
5.2. ADNS-3080

Problém s instalací optické soustavy se zdál být vyřešen u optického senzoru ADNS-3080, taktéž používaného v počítačových myších. V komunitě lidí, zajímajících se o robotiku a drony, je tento senzor používán pro snímání oblasti pod dronem, například pro udržování dronu ve vzduchu na jednom místě. K tomu je využíváno optické soustavy s ohniskovou vzdáleností 4,2 mm, která je instalována k DPS spolu se senzorem. ADNS-3080 má však oproti senzoru ADNS-9800 horší parametry, které neumožní správnou detekci velkých rychlostí. Maximální detekce pohybu tohoto senzoru je 40 ips, zrychlení 15 g. Oproti ADNS-9800 má také možnost využití rozlišení pouze na dvou úrovních, a to 400 a 1600 cpi. V této práci byla hodnota nastavena na 1600 cpi.



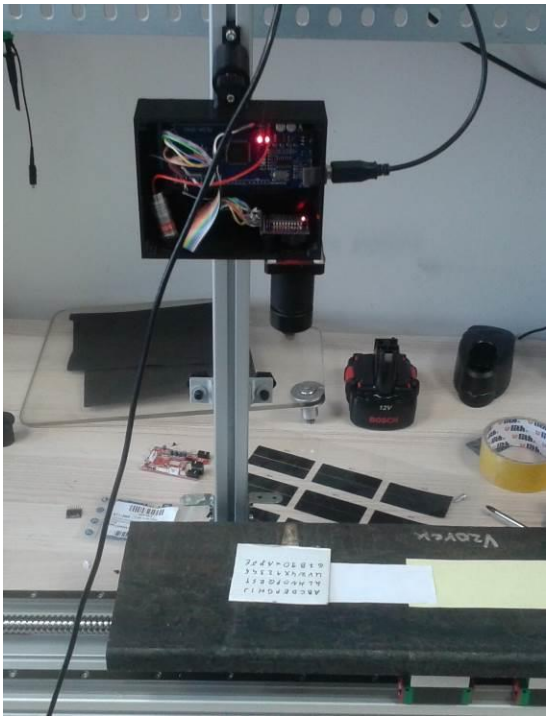
Obrázek 5-2 – Připojení senzoru ADNS-3080 k platformě Arduino Mega2560, [5], [17]

Podobně jako u předchozího senzoru bylo potřeba propojit jej s elektronickou platformou sloužící ke komunikaci a napájení. K tomuto účelu byla opět vybrána deska Arduino Mega2560. Obrázek 5-2 znázorňuje propojení pinu těchto dvou prvků. Komunikace mezi deskou a senzorem probíhala přes sériové periferní rozhraní (SPI), byly tedy kromě vodičů pro spojení 5 V a země připojeny vodiče pro MOSI, MISO, SCLK a NCS. Na rozdíl od ADNS-9800 nebylo potřeba napětí 5 V, se kterým pracuje SPI na desce Arduino, snižovat na 3,3 V, neboť senzor dokáže pracovat i při napětí 5 V z mikrokontroléru. K programování desky bylo použito vývojové prostředí Arduino IDE. Po nahrání zdrojového kódu do desky bylo možné ze senzoru získávat informaci o pohybu nebo přímo surová data v podobě obrázků. V obou případech pracuje senzor v tzv. Burst módu, který umožňuje zkrátit čas potřebný pro přenos dat. Pro získání informace o změně pohybu je používán mód Motion Read, který je zahájen přečtením dat z registru Motion_Burst. Senzor ADNS-3080 poté pošle jednotlivé bajty, které je potřeba načíst (Motion, Delta_X, Delta_Y, SQUAL, Shutter_Upper, Shutter_Lower a Maximum_Pixel). Informaci o velikosti detekovaného pohybu nesou druhý a třetí bajt, čtvrtý značí kvalitu snímaného povrchu (přesněji čtvrtinu počtu rozlišených detailů v obraze), pátý a šestý pak po spojení značí dobu závěrky. Při zapnutí je potřeba pro správnou funkci senzor resetovat pomocí Reset pinu, který byl spojen s pinem 48 na desce Arduino. Zápis a čtení z registrů je podobné jako u senzoru ADNS-9800, viz výše. Změny jsou pouze v časech nutných pro zpoždění při prodlevách mezi příkazy. Kód je uveden v příloze H. [8]

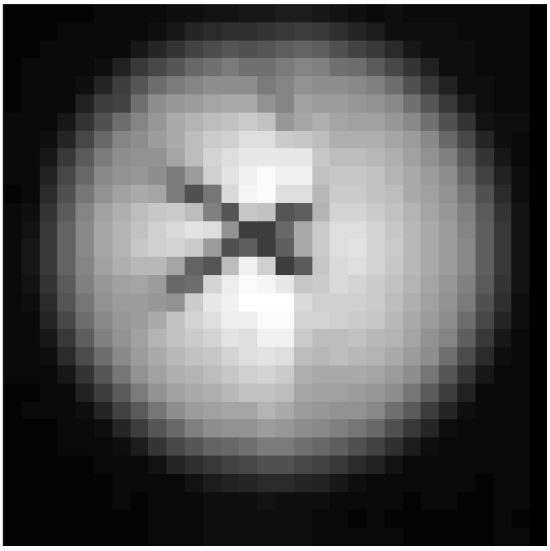


Obrázek 5-3 – Testování senzoru ADNS-3080 s čočkou o ohniskové vzdálenosti 4,2 mm

Pomocí použité optické soustavy je možné snímat na vzdálenost třiceti centimetrů obraz o velikosti cca 20 cm, proto bylo potřeba přidat další čočku, která by při správném zaostření umožnila snímat pouze malou plochu, nejlépe 1 mm^2. Byla testována čočka s ohniskovou vzdáleností 50 mm, kterou se povedlo zaostřit na povrch o ploše cca 1 cm^2. Zaostření bylo provedeno pomocí snímání obrázků povrchu a jejich vykreslování v programu MATLAB. Přestože zaostření se zdálo být dobré, na obrázku je zřetelné X o velikosti cca 5 mm (viz Obrázek 5-5), senzor nedokázal správně reagovat na pohyb snímaného povrchu.



Obrázek 5-4 – Testování přídavné optické soustavy k senzoru ADNS-3080



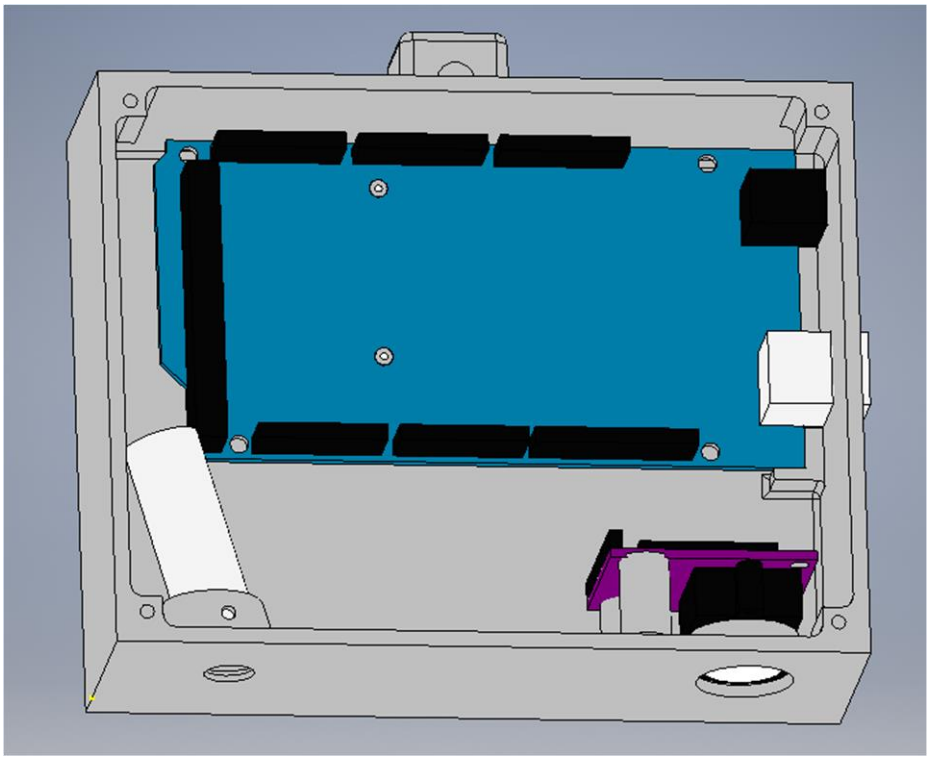
Obrázek 5-5 – Obraz získaný senzorem ADNS-3080 ze vzdálenosti 30cm

6. Návrh mechanické konstrukce rychloměru

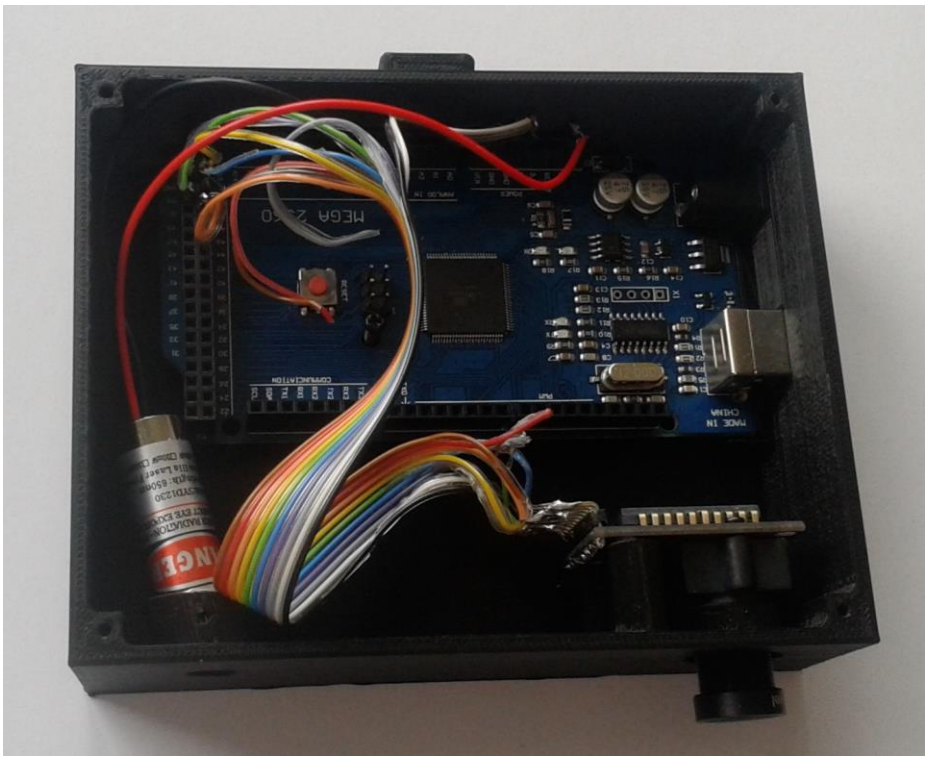
Jednotlivé části rychloměru musí být integrovány do přenosné formy odolávající, pokud možno, všem okolním vlivům a tvořící zařízení. Výsledná konstrukce pouzdra byla navržena pomocí CAD softwaru Autodesk Inventor. Pro senzor ADNS-3080 a senzor ADNS-9800 byly vytvořeny rozdílné konstrukce, neboť DPS s montážními dírami byly u jednotlivých senzorů různé. Zároveň bylo rozdílné natočení senzorů vzhledem ke měřenému objektu. Senzor ADNS-3080 byl instalován kolmo k měřenému povrchu, zatímco ADNS-9800 bylo do krytu instalováno pod úhlem 30°. Důvodem byla konstrukce samotného senzoru.

Navrhnuté součásti byly tištěny na 3D tiskárně Ultimaker 3 Extended založené na technologii aditivní výroby FDM (Fused Deposition Modeling). Přesnost této tiskárny je 12,5, 12,5 a 2,5 µm. Při rychlosti výstavby až 16 mm^3/s probíhal tisk jednotlivých krytů cca 6 – 10 hodin. Na výrobu byl použit materiál PLA (polylactid acid – kyselina polymléčná), konkrétně PLA Extrafill Trafic Black s průměrem 2,85 mm, společnosti Parzlich s.r.o. Na výrobu jednoho krytu zařízení bylo použito cca 100 gramů tohoto materiálu. Výroba takovéhoto krytu vyjde přibližně na 100 Kč. [29]

Rozměry mechanických konstrukcí se odvíjely od velikosti a možnostech vložení jednotlivých součástí, tedy desky Arduino Mega2560 s nahraným programem, senzoru ADNS-3080 a laseru pro osvětlování povrchu objektu. Do víka konstrukce bylo dále potřeba vložit displej. Kryt pro zařízení se senzorem ADNS-3080 byl tedy navrhnut ve tvaru kvádru o rozměrech 120 x 90 x 36,5 mm. Kryt pro zařízení se senzorem ADNS-9800 má rozměry 135 x 85 x 70 mm. Tloušťka stěn byla zvolena 2,5 mm, aby bylo dosaženo dostatečné mechanické pevnosti. Ve stěnách byly ponechány otvory pro přívod napájení k desce Arduino a pro připojení sériové sběrnice k této desce, dále výstupní otvor pro svazek světla z laseru a vstupní otvor pro záření vstupující do detektoru. Upevnění laseru bylo voleno tak, aby bylo možné měřit povrch ve vzdálenosti 30 cm. Proto je u zařízení se senzorem ADNS-3080 laser instalován pod úhlem cca 75°, u senzoru ADNS-9800, kde bylo potřeba umístit optickou soustavu před senzorem blíže k laseru, svírá úhel laseru a spodní strany krytu cca 76,5°.

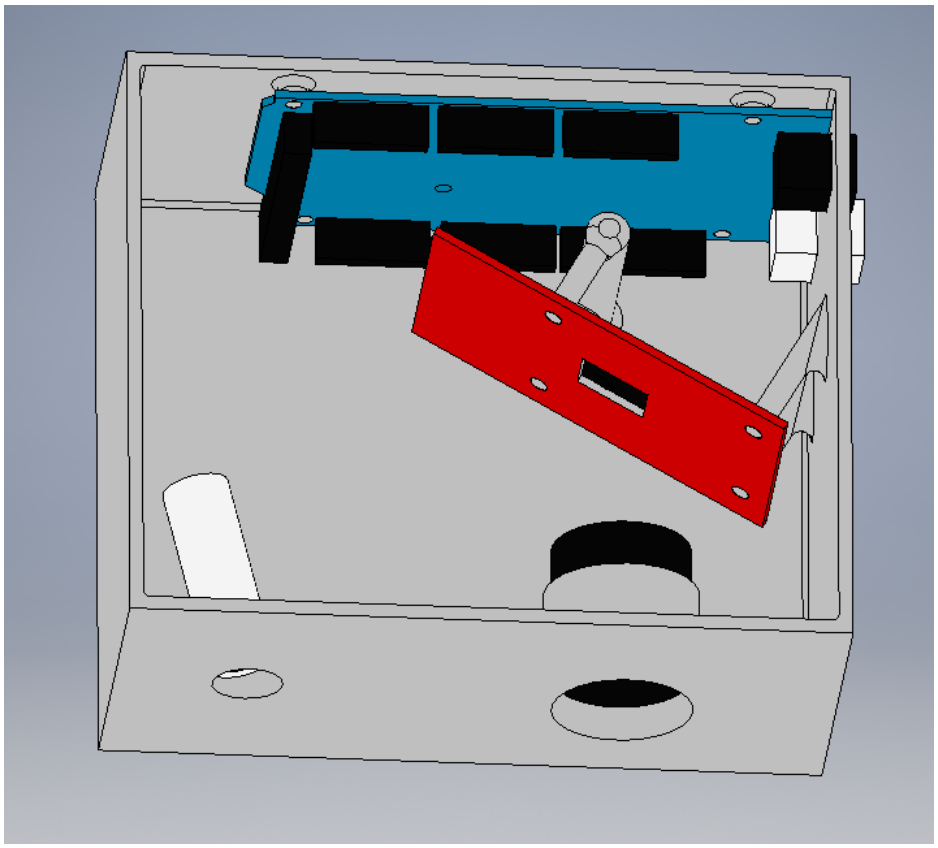


Obrázek 6-1 – Návrh krytu pro senzor ADNS-3080, Arduino a laser

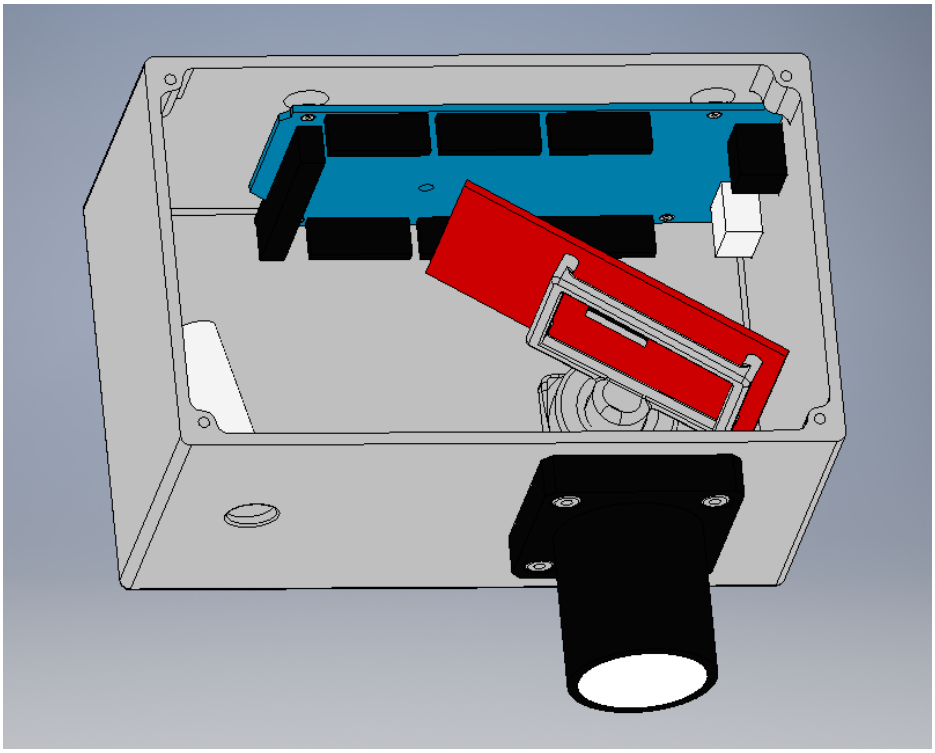


Obrázek 6-2 – Mechanická konstrukce pro ADNS-3080

U zařízení se senzorem ADNS-3080 byl montážní prvek optické soustavy pevně spojen se senzorem, proto bylo možné ukotvit tyto komponenty společně, viz Obrázek 6-1. Pro zařízení se senzorem ADNS-9800 bylo potřeba vhodně ukotvit jak senzor, tak optickou soustavu. Nejdříve byla navržena konstrukce s pevným ukotvením tohoto senzoru, viz Obrázek 6-3. Při měření se však vyskytla situace, kdy bylo potřeba senzor posunout vůči optické soustavě i v jiném směru, než v ose optické soustavy. Proto byla vytvořena soustava montážních prvků a nový kryt, do něhož byla soustava instalována. Po ukotvení senzoru k této soustavě, bylo možné pohybovat senzorem vůči optické soustavě v ose x, y i z. Kryt zařízení řešený tímto způsoben znázorňuje Obrázek 6-4.



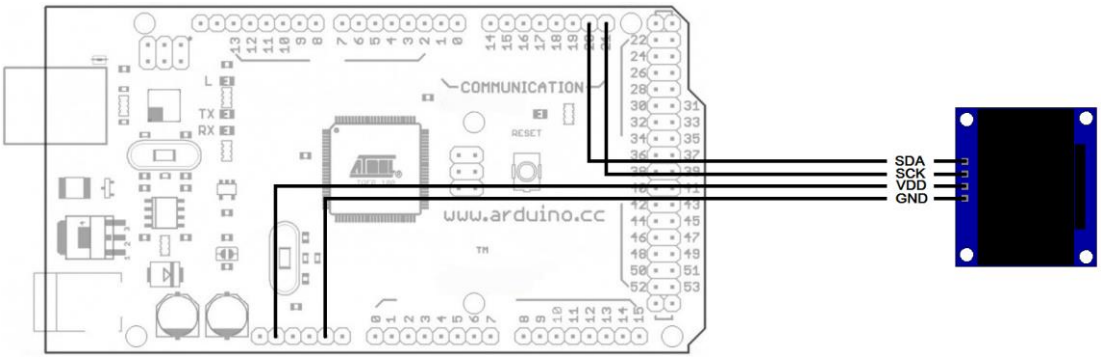
Obrázek 6-3 – Mechanická konstrukce pro zařízení se senzorem ADNS-9800



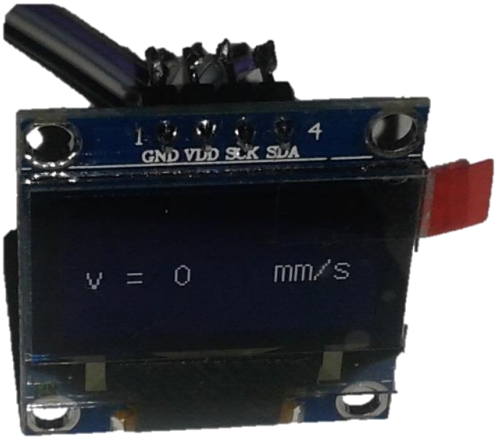
Obrázek 6-4 – Mechanická konstrukce pro zařízení se senzorem ADNS-9800, 2. verze

6.1. Instalace displeje

Primárním výstupem tohoto zařízení jsou informace o aktuální rychlosti posílané přes USB port do dalších zařízení, která tuto informaci zpracují a použijí například ke 3D rekonstrukci povrchu měřeného objektu. Je však příhodné, aby zařízení mělo možnost sdělovat informace obsluze, a to nejen o aktuálním pohybu, ale také například vyskytne-li se chyba, kterou bude muset obsluha řešit. Poslední komponentou instalovanou do zařízení je tedy OLED displej. Velikost 0,96 palců je dostatečně velká pro sdělení aktuální rychlosti, zároveň je však spotřeba energie tohoto displeje velmi nízká. Rozlišení použitého displeje je 128 x 64 pixelů. Napájení 5 V bylo přivedeno z použité elektronické platformy Arduino Mega2560, současně byly propojeny příslušné piny SDA a SCK, viz Obrázek 6-5. Na rozdíl od senzoru ADNS-3080 a ADNS-9800 neprobíhá komunikace mezi deskou a displejem pomocí SPI rozhraní, ale přes sériovou sběrnici I^2C. Do prostředí Arduino IDE bylo potřeba nainstalovat příslušnou knihovnu U8Glib a doplnit kódy o část sloužící ke komunikaci s deskou Arduino. Tato úprava je zahrnuta v kódu v příloze G. [4]



Obrázek 6-5 – Schéma připojení displeje pomocí I^2C k Arduinu, [17]



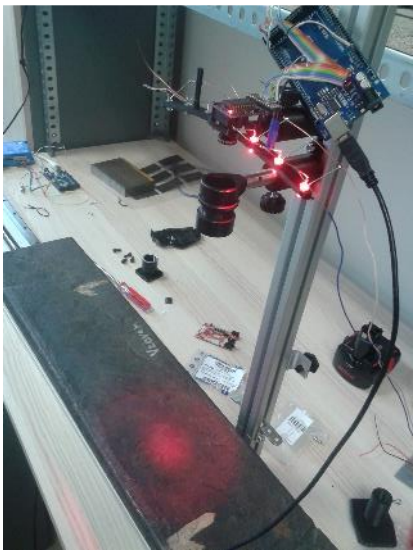
Obrázek 6-6 – Použitý OLED displej

7. Experimentální verifikace funkčnosti rychloměru povrchu

S vytvořenými zařízeními na bázi senzorů ADNS-9800 a ADNS-3080 bylo provedeno experimentální měření rychlosti povrchu u osmi vzorků s rozdílnou odrazivostí. Konkrétně se jednalo o černý papír, kartonový papír, hliníkovou pásku, lesklý a matný žlutý papír, bílý papír, růžový papír a ocelový sochor. Zařízení bylo umístěno 30 cm nad měřený povrch, který byl osvětlován laserem o vlnové délce 650 nm instalovaným v zařízení, a pohyboval se konstantní rychlostí. Bylo provedeno několik různých měření na černém papíru, kartonovém papíru, hliníkové pásce, žlutém, bílém a růžovém papíru a ocelovém sochoru.

7.1. ADNS-3080

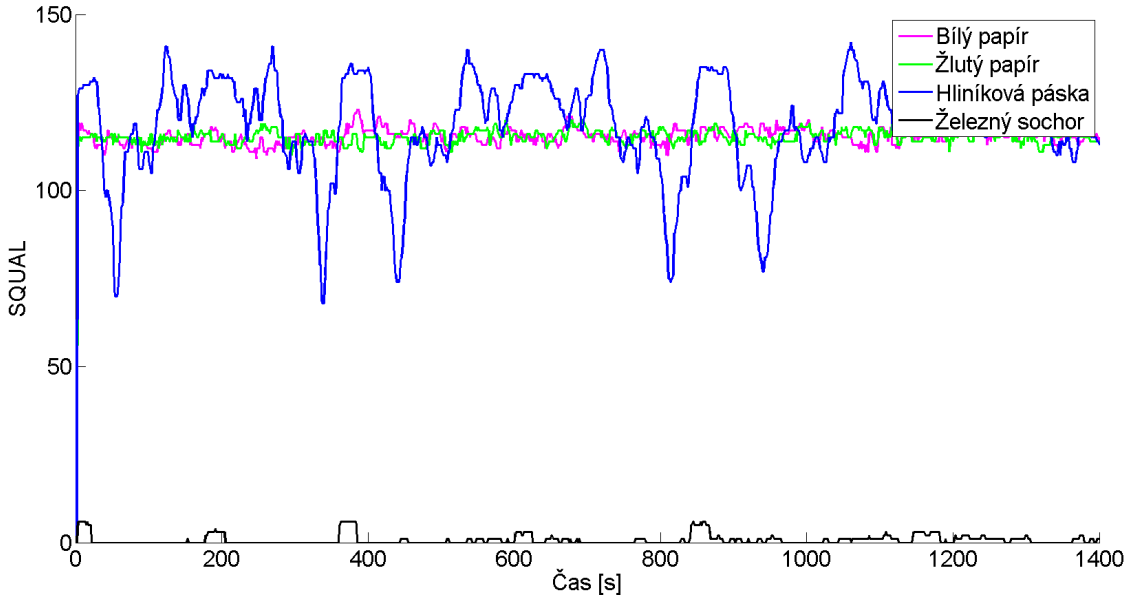
Měření se senzorem ADNS-3080 bylo prováděno jak na každém povrchu zvlášť, tak s plynulým přechodem z povrchu na povrch. Dále bylo také testováno osvětlení povrchu pomocí LED diod místo laseru. Do konstrukce, do níž byl instalován senzor, byly připojeny červené diody. Spolu s optickou soustavou o ohniskové vzdálenosti 4,2 mm, která byla již instalována k senzoru, sloužila pro zaostření na povrch čočka s ohniskovou vzdáleností 50 mm. Byly testovány různé možnosti zapojení diod, kdy byla aktivní jedna či více diod.



Obrázek 7-1 – Testování LED osvětlení povrchu

7.1.1. Kvalita povrchu a vliv okolního osvětlení

Pro správnou detekci pohybu povrchu, musí senzor rozeznávat základní rysy tohoto povrchu. Obraz na detektoru tedy musí být správně zaostřen a povrch dobře osvětlen. To, kolik detailů povrchu senzor rozezná, popisuje hodnota SQUAL (Surface Quality). SQUAL nabývá maximální hodnoty 169. Pokud se pod senzorem nenachází žádný povrch, SQUAL nabývá hodnoty 0. Změny ve snímaném obraze vedou ke změnám detailů, proto se v průběhu měření hodnota SQUAL mění. Na následujícím obrázku je znázorněno SQUAL při měření na čtyřech různých površích pohybujících se rychlostí 10 mm/s cca 15 cm. Je zřejmé, že odrazivost povrchu hodnotu SQUAL značně ovlivní. Hliníková páska laserový svazek odrážela do různých směrů a měřená hodnota tohoto parametru znatelně fluktuovala. Ideální kvalita povrchu byla zaznamenána na bílém a žlutém papíře, zatímco ocelový sochor záření z velké části pohlcoval a kvalita povrchu byla velmi nízká, což zamezovalo správnému měření rychlosti. [8]

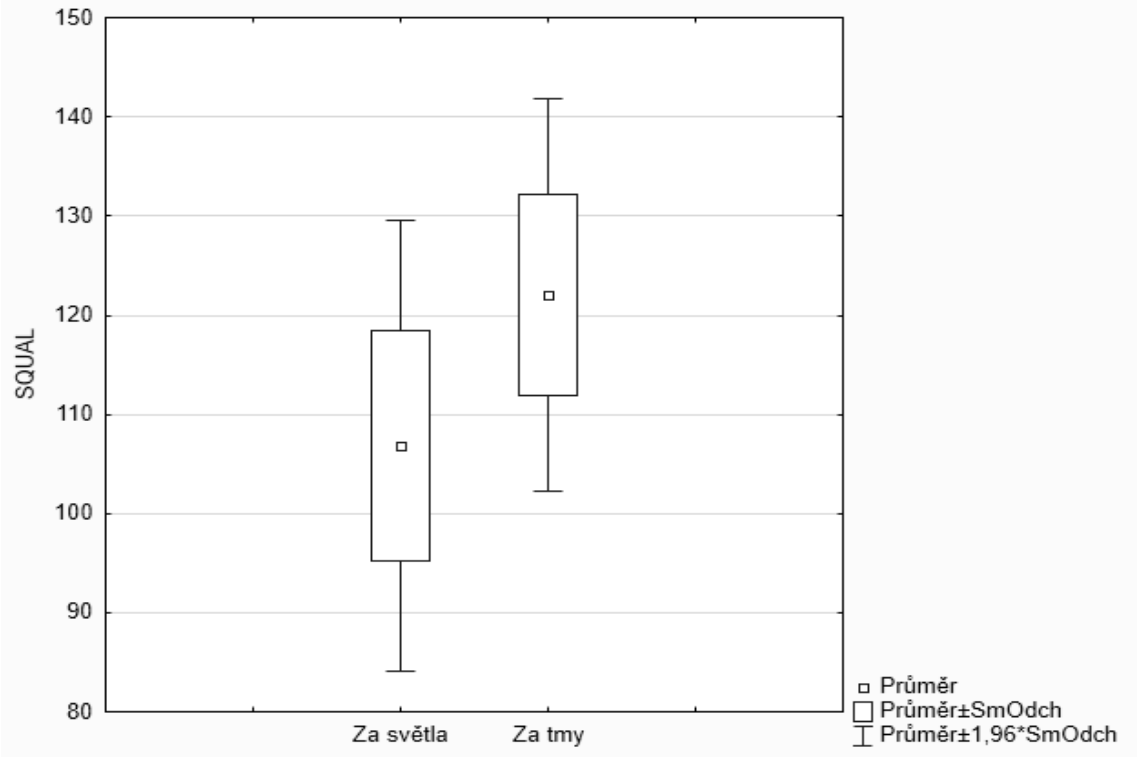


Obrázek 7-2 – Hodnoty SQUAL pro čtyři různé povrchy při konstantním pohybu

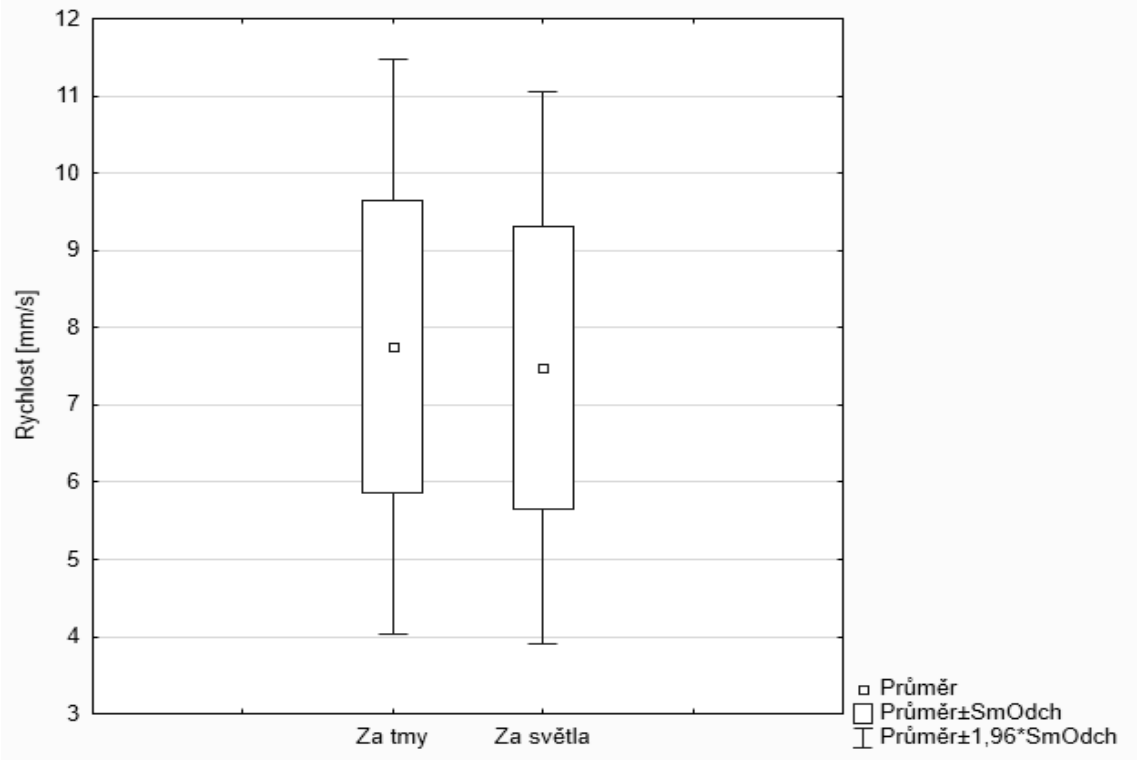
Při měření byl testován vliv osvětlení v měřící místnosti na detekovanou kvalitu povrchu. Bylo provedeno dvanáct měření v místnosti osvětlené zářivkami a světlem vstupujícím oknem a dvanáct měření při zhasnutém osvětlení a zataženými žaluziemi. Nejprve bylo potřeba ověřit, zda má diference závislých prvků normální rozložení, aby bylo možné porovnat tyto dvě skupiny dat parametrickými testy. Byl použit program STATISTICA 12 (StatSoft ČR s.r.o., Praha, Česká republika). Normální rozložení bylo hodnoceno z histogramu a pomocí Shapiro-Wilkova testu. p-hodnota vyšla menší než 0,05, proto nebylo možné pro vyhodnocení závislosti použít parametrický test. Byl tedy

použit neparametrický znaménkový test, s výsledkem p-hodnota = 0,001, což znamená, že mezi naměřenými daty je významný statistický rozdíl. Jak je vidět z krabicového grafu, kvalita povrchu nabývala vyšších hodnot při nižším celkovém osvětlení v místnosti, což je způsobeno pravděpodobně vyšším kontrastem mezi povrchem osvětleným laserovým svazkem světla a neosvětleným povrchem, viz Obrázek 7-3. [30]

Vznikla tak otázka, zda je měřená rychlost ovlivněna mírou osvětlení v místnosti. Bylo porovnáno 18 měření při zapnutém a vypnutém osvětlení místnosti. Hodnoty diferencí měly normální rozložení, byl tedy použit parametrický párový t-test. Naměřená rychlost ve tmě byla vyšší průměrně o 0,28 mm/s, p-hodnota vyšla 0,000001, rozdíl při měření za různých světelných podmínek je tedy významný.



Obrázek 7-3 – Krabicový graf hodnot SQUAL za různých světelných podmínek



Obrázek 7-4 – Krabicový graf rychlostí měřených za různých světelných podmínek

7.1.2. Testování vlivu velikosti osvětlené plochy

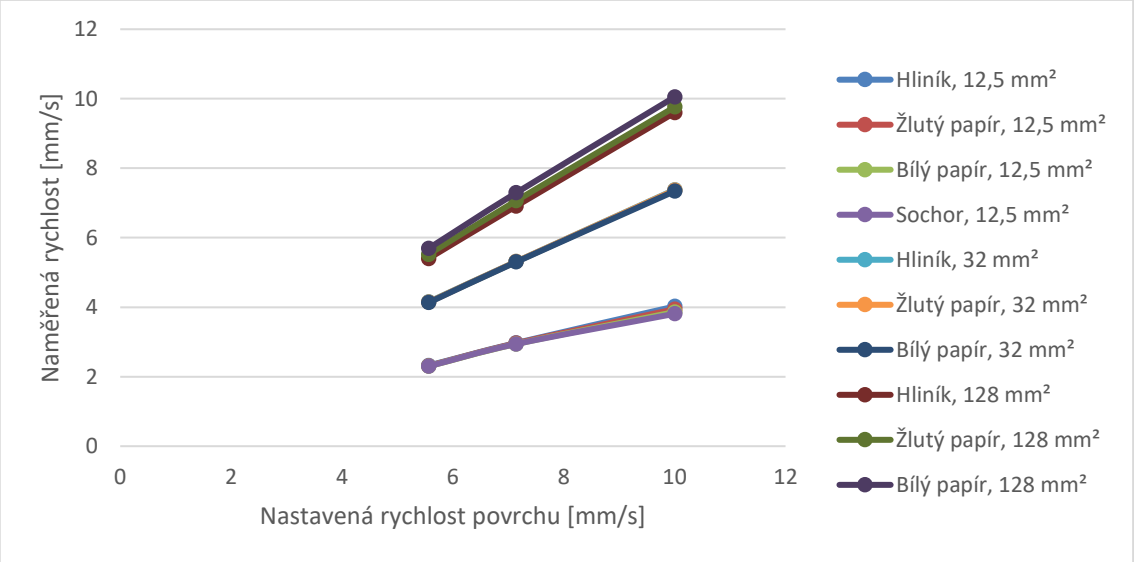
První měření bylo prováděno na hliníkové pásce, žlutém papíru, bílém papíru a ocelovém sochoru. Byla testována odezva na 3 rychlosti, při rozsvíceném osvětlení místnosti i při zhasnutém. Dále byl testován vliv velikosti plochy osvětlené laserem.

Na první tři povrchy reagoval senzor velmi dobře, na čtvrtém povrchu, pravděpodobně kvůli vysoké pohltivosti světla, senzor detekoval rychlost pouze při osvětlení povrchu úzkým svazkem laserového světla. Intenzita odraženého záření byla v tomto případě největší. Z tohoto důvodu byly do statistik zahrnuty pouze první tři povrchy. Plocha osvětlovaná laserem byla nejlépe charakterizovaná jako obdélník s obsahem 12,5 mm^2, 32 mm^2 a 128 mm^2. Velikost osvětlené plochy ovlivňovala hodnotu odhadované rychlosti povrchu (viz Obrázek 7-5), proto nebylo možné jednoduše použít přepočet pomocí příčného zvětšení použité čočky. V grafu je proto zobrazena přímo naměřená rychlost, která je závislá na rychlosti dopravníku. Pro dosažení správného vyhodnocení měření je potřeba provést korekci s vypočtenou konstantou přizpůsobenou jak zvětšení optické soustavy, tak velikosti osvětlené plochy. V grafu jsou zaznamenány naměřené rychlosti na hliníkové fólii, žlutém a bílém papíru při třech rychlostech dopravníku a třech různých divergencích laserového svazku světla. Můžeme pozorovat různou odezvu na pohyb v závislosti na povrchu. Nejmenší odchylka od střední hodnoty, a tedy

nejmenší vliv druhu povrchu, byla zaznamenána při osvětlení 32 mm^2 povrchu. Hodnoty jsou uvedeny v tabulce (Tabulka 7-1). Při rozostření laseru na plochu 128 mm^2 byla detekovaná rychlost velmi blízká nastavené rychlosti dopravníku, vliv druhu povrchu zde byl však větší, proto bylo jako optimální vybráno předchozí nastavení.

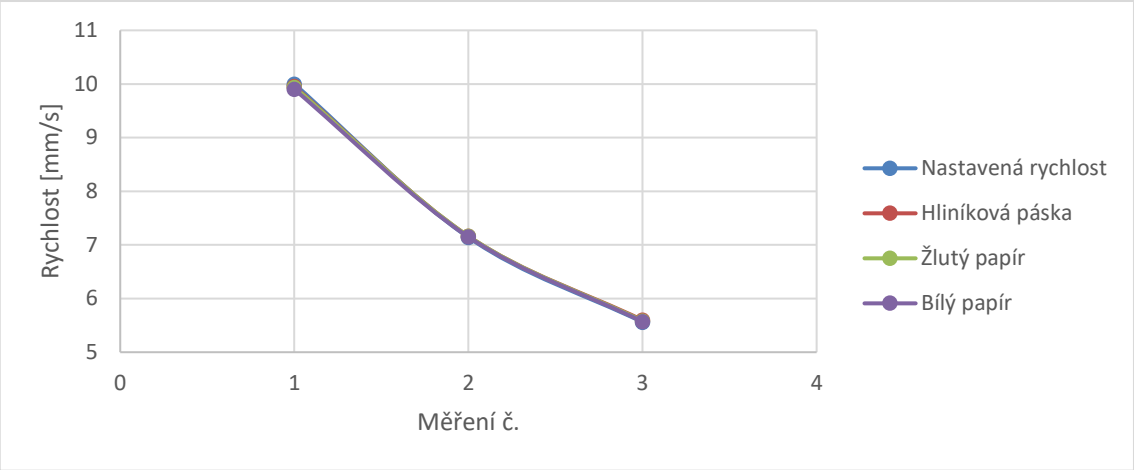
Tabulka 7-1 – Směrodatné odchylky (σ) při různém osvětlení

Osvětlená plocha laserem [mm^2]	σ v osvětlené místnosti	σ při měření za šera
12,5	0,04	0,03
32	0,01	0,01
128	0,19	0,03



Obrázek 7-5 – Graf naměřených hodnot pro tři různé povrchy při různé divergenci laserového osvětlení v osvětlené místnosti

Pro nastavení s osvětlením 32 mm^2 povrchu byla provedena kalibrace díky známé rychlosti povrchu, která byla nastavena řízením krokového motoru. Kalibrační koeficient vyšel 1,35. Při dodržení vzdálenosti 30 cm, osvětlení 32 mm^2 povrchu a zaostření optické soustavy na povrch, je po vynásobení koeficientem 1,35 získána rychlost měřeného povrchu. Nastavenou rychlost a naměřené rychlosti po kalibraci ukazuje Obrázek 7-6.



Obrázek 7-6 – Naměřená rychlost třech povrchů po kalibraci

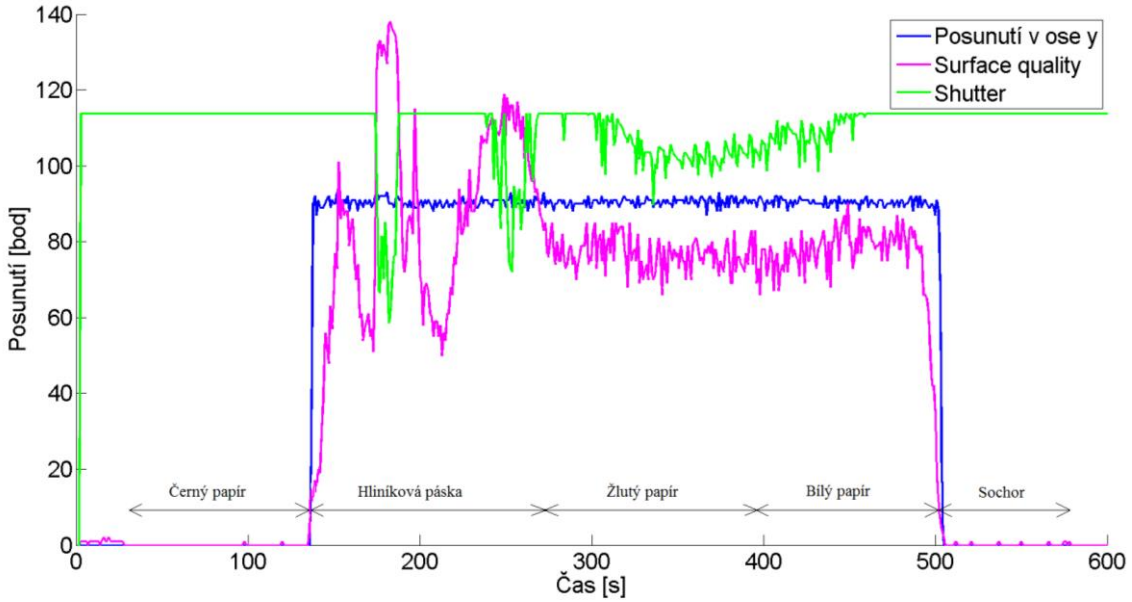
Toto měření sloužící k posouzení vlivu velikosti plochy ozařované laserem na měřenou rychlost vedlo k závěru, že velikost této plochy značně ovlivňuje naměřené hodnoty. Jako optimální řešení se zdá být nastavení, kdy je laserem ozařována plocha o obsahu cca 32 mm^2. Při tomto nastavení je rozptyl na různých površích nejmenší. Z toho důvodu je možné provést kalibraci vedoucí ke správnému určení rychlosti měřeného objektu.

7.1.3. Měření rychlosti různých povrchů

Další měření bylo zaměřeno na změnu detekované rychlosti na černém papíru, hliníkové pásce, žlutém papíru, bílém papíru a ocelovém sochoru. Povrchy byly položeny vedle sebe tak, aby při pohybu posuvníku bylo možné měřit povrchy postupně za sebou, každý povrch cca 10 cm. Měření bylo provedeno třikrát, za různých světelných podmínek. Nejdříve při osvětlení zářivkami a světlem vstupujícím do místnosti oknem, ve druhém měření při zhasnutém osvětlení a třetí měření proběhlo v šeru se zhasnutým osvětlením a zataženými žaluziemi. Na následujícím grafu je záznam měření v ose y, současně snímaná kvalita povrchu a závěrka, kterou senzor přizpůsoboval vstupujícímu světlu. Hodnota závěrky (Shutter) byla vydělena osmdesáti pro lepší znázornění v grafu (Obrázek 7-7).

Snímán byl povrch ze vzdálenosti 30 cm, plocha ozařovaná laserem byla 128 mm^2. Prvních asi 30 sekund se dopravník nepohyboval a senzor byl zaostřen na černý papír. V tomto úseku se kvalita povrchu pohybovala v řádech jednotek. Jakmile se dopravník začal pohybovat, kvalita povrchu klesla k nule a žádný pohyb nebyl detekován. Současně měl senzor nastavenou dobu závěrky na maximum. Díky tomu lze soudit, že intenzita odraženého světla od povrchu byla nedostatečná a bylo by vhodné zaostřit

laser na menší část povrchu. Při změně povrchu na hliníkovou pásku byl již senzor schopný zaznamenat pohyb, hodnota SQUAL prudce stoupla a vzápětí také zareagoval senzor přizpůsobením doby závěrky. Protože hliníková páska měla velkou odrazivost a povrch nebyl dokonale rovný, hodnota SQUAL značně kolísala. Mírně se ustálila, jakmile do zorného pole senzoru vstoupil žlutý papír. V tomto případě změna povrchu neměla vliv na měřenou rychlost. Velmi podobné byly získané hodnoty při zaostření na bílý papír. V posledním úseku, kterým byl ocelový sochor, opět nebyl zaznamenán žádný pohyb a kvalita povrchu byla téměř nulová.



Obrázek 7-7 – Záznam měření pohybu v ose y, SQUAL a Shutter

Dopravník s instalovanými povrchy se pohyboval rychlostí 10 mm/s. Průměrná rychlost naměřená při detekci pohybu, hodnoty SQUAL a Shutter jsou uvedeny v následující tabulce. Při větší intenzitě osvětlení byla naměřená rychlost nižší, SQUAL vyšší a doba závěrky kratší.

Tabulka 7-2 – Naměřené rychlosti, SQUAL a hodnoty závěrky při různém osvětlení místnosti

Měření č.	Naměřená rychlost [mm/s]	SQUAL	Shutter
1	14,39	82	8683
2	14,45	81	9035
3	14,54	79	9080

7.2. ADNS-9800

Se zařízením založeném na detekci pohybu senzorem ADNS-9800, optickou soustavou o ohniskové vzdálenosti 50 mm zaostřující na povrch ve vzdálenosti 30 cm bylo provedeno několik měření zaměřených na testování vlivu různých činitelů na detekci pohybu. Měření bylo prováděno na různých typech povrchů za účelem zjištění reakcí na povrchy s různou odrazivostí a homogenitou. Byl testován vliv použití různého nastavení optické soustavy a laseru osvětlujícího měřený povrch, přičemž byla vyhodnocena různá reakce na pohyb povrchů na lineárním dopravníku. Žádná z testovaných podmínek však nevyhovovala všem typům povrchu současně.

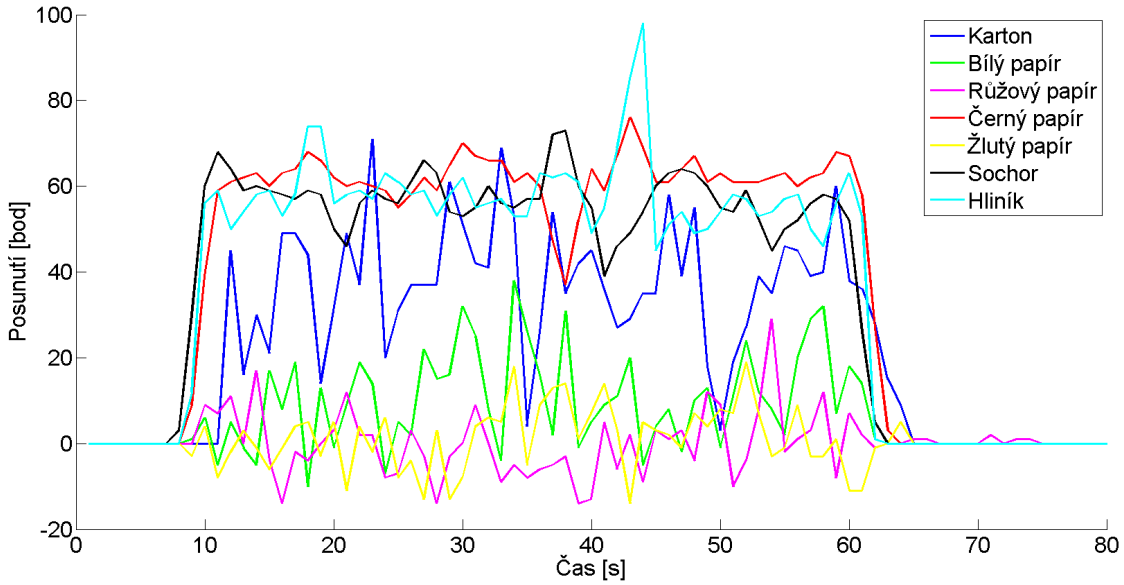
7.2.1. Měření rychlosti různých povrchů

První měření proběhlo na sedmi různých typech povrchů. Senzor zaostřený na místo ozařované laserem snímal pohyb kartonu, bílého papíru, lesklého růžového papíru, matného černého papíru, lesklého žlutého papíru, ocelového sochoru a hliníkové pásky.

Tabulka 7-3 – Hodnoty SQUAL pro různé povrchy za čtyř různých podmínek

Osvětlená plocha [mm^2]	Osvětlení místnosti	Karton	Bílý papír	Růžový papír	Černý papír	Žlutý papír	Ocelový sochor	Hliníková páska
18	Rozsvíceno	51	49	49	68	47	73	93
	Šero	54	51	50	70	48	74	96
85	Rozsvíceno	45	39	37	65	35	68	92
	Šero	47	40	38	68	37	72	97

Hodnoty SQUAL (kvalita povrchu) jsou ve všech případech vyšší při měření v zatemněné místnosti, v průměru o 2,2.



Obrázek 7-8 – Záznam měření na různých typech povrchů, osvětlená plocha 85 mm²

Tabulka 7-4 obsahuje naměřené rychlosti na různých površích. Je zde patrný rozdíl v jednotlivých případech, na žlutém, růžovém a bílém papíře byly naměřené hodnoty velmi kolísavé. Reakce na pohyb kartonu byla o něco lepší, povrch zde nebyl tak homogenní. Nejlepší reakce byla na členité a tmavé povrchy, viz Obrázek 7-8, zobrazující průběh měření posunutí v ose y, dopravník se posunul cca o 6 cm.

Tabulka 7-4 – Naměřená rychlost různých povrchů

Osvětlená plocha [mm²]	Osvětlení místnosti	Karton	Bílý papír	Růžový papír	Černý papír	Žlutý papír	Ocelový sochor	Hliníková páska
18	Rozsvíceno	1,89	0,54	0,00	3,00	0,00	3,18	2,31
	Šero	1,24	0,44	0,19	3,01	0,00	3,08	2,28
85	Rozsvíceno	1,51	0,42	0,18	2,22	0,24	2,07	2,16
	Šero	1,31	0,45	0,00	2,30	0,09	2,13	2,12

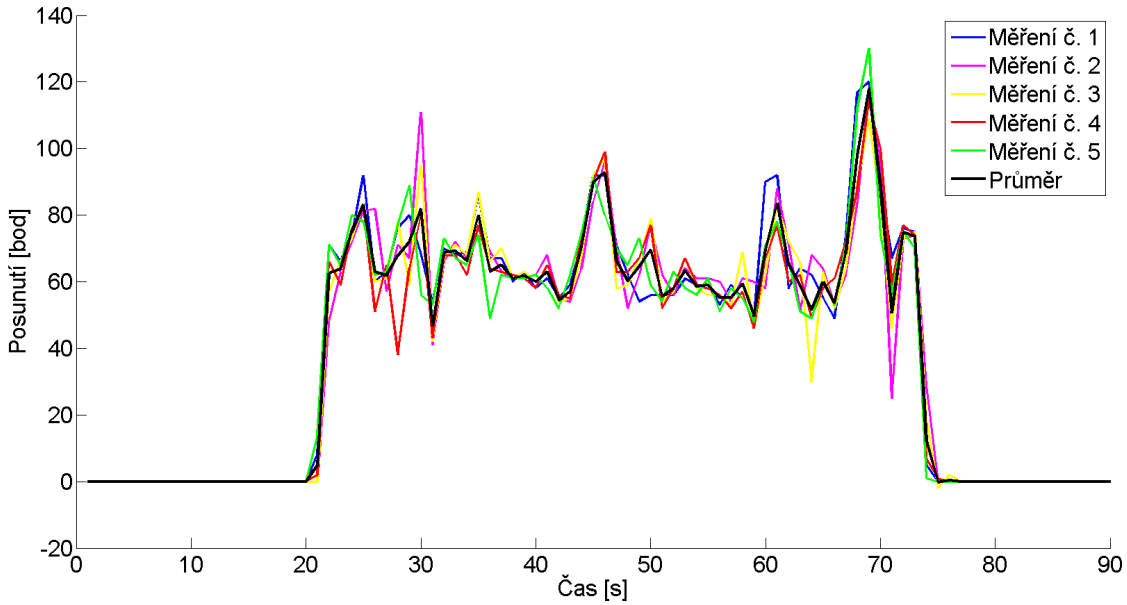
Tabulka 7-5 – Hodnoty závěrky

Osvětlená plocha [mm²]	Osvětlení místnosti	Karton	Bílý papír	Růžový papír	Černý papír	Žlutý papír	Ocelový sochor	Hliníková páska
18	Rozsvíceno	198	124	117	1236	116	1442	65
	Šero	226	137	128	1373	121	1583	58
85	Rozsvíceno	677	386	395	4015	400	3593	242
	Šero	653	382	378	4094	389	3802	182

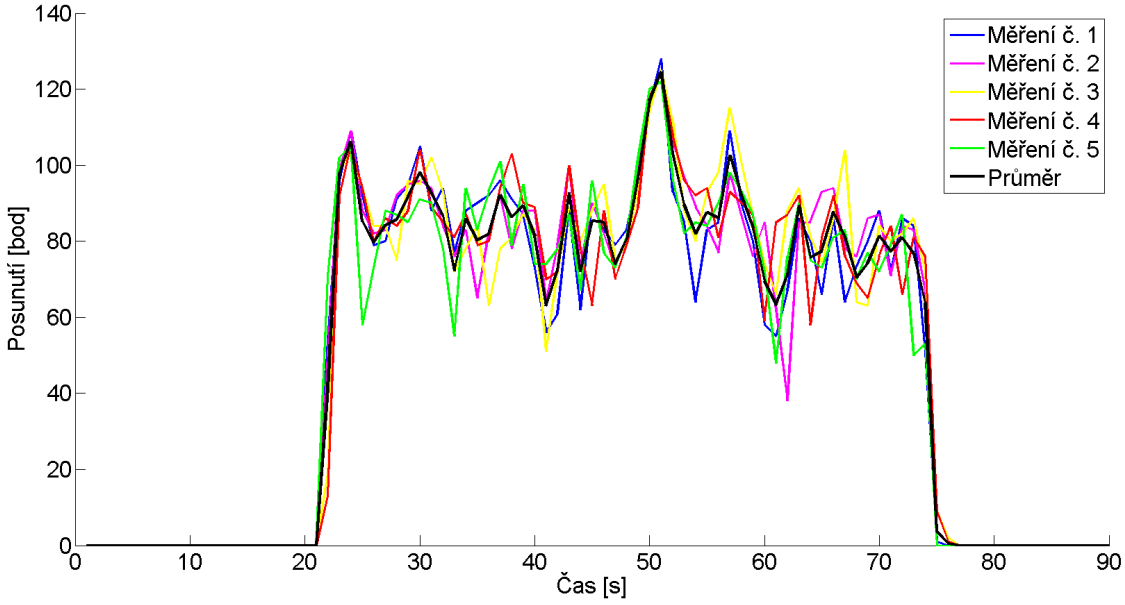
Senzor automaticky s každým novým obrazem upravuje dobu závěrky (Shutter), aby byly hodnoty jednotlivých pixelů v obraze v normálních operačních hodnotách. Jednotky jsou počty cyklů vnitřního oscilátoru. Tato doba se také lišila v závislosti na povrchu, viz Tabulka 7-5. Nejdelší čas závěrky byl nastaven při měření na černém papíru a na ocelovém sochoru, neboť byly tyto povrchy nejtmavší. Naopak na hliníkové pásce, která dobře odrážela světlo z laseru, byla závěrka nastavena na velmi krátký čas. Nastavená doba závěrky neměla žádný vliv na kvalitu měření, neboť senzor reagoval jak na pohyb hliníkové pásky, kde byla doba závěrky nejkratší, tak na černý papír a ocelový sochor, kde závěrka byla i o dva řády vyšší. [11]

Nejmenší rozptyl naměřených rychlostí za různých podmínek byl patrný u hliníkové pásky a černého papíru, podobné přesnosti bylo dosaženo také u měření na sochoru.

Měření jednotlivých povrchů probíhalo opakovaně za stejných podmínek, pro snížení vlivu náhodných chyb. Při porovnání pěti naměřených dat z měření na sochoru je patrné, že na aktuální měřenou rychlost má vliv nehomogenita povrchu, viz Obrázek 7-9 a Obrázek 7-10. Pro vyloučení možnosti, že opakující se vzor změn rychlosti je způsoben nelinearitou použitého lineárního dopravníku, bylo provedeno další měření, viz kapitola 7.2.2.

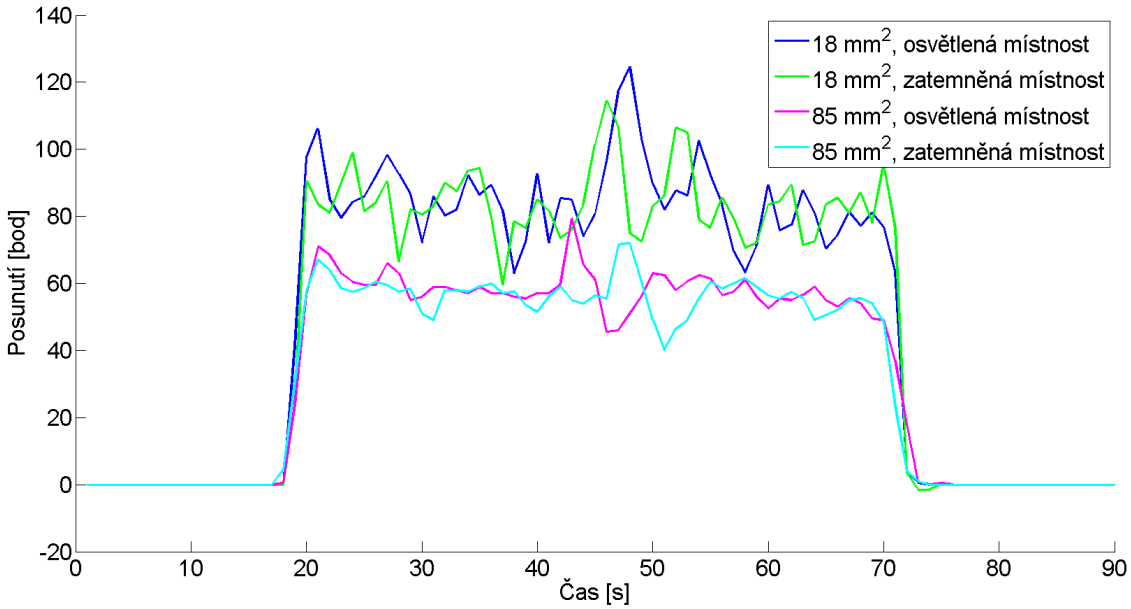


Obrázek 7-9 – Záznam pěti měření na hliníkové fólii za stejných podmínek



Obrázek 7-10 – Záznam pěti měření na sochoru za stejných podmínek

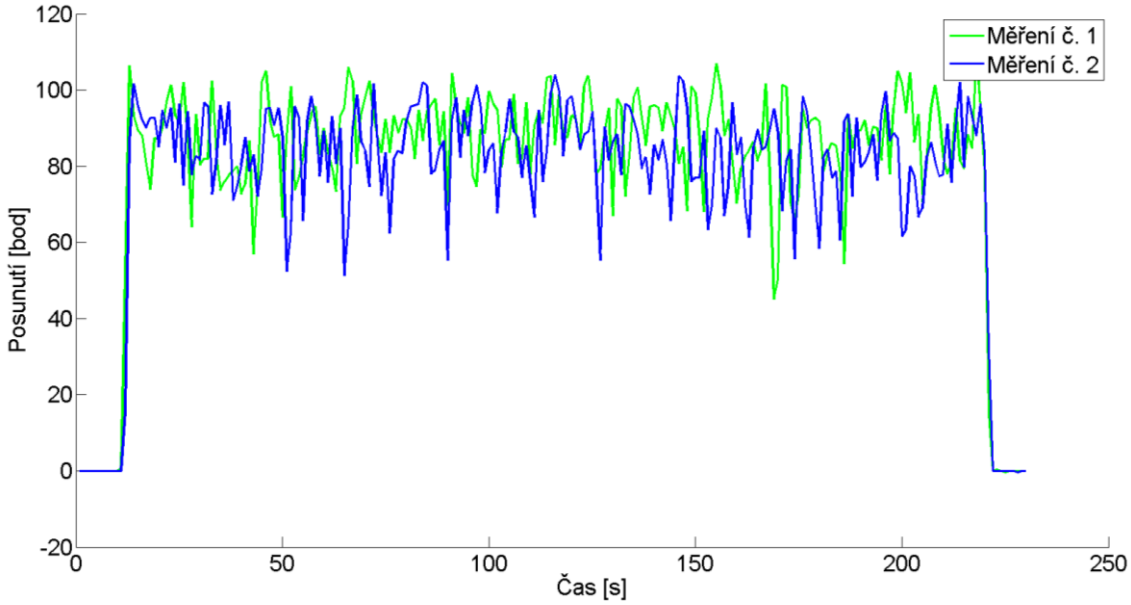
Na následujícím obrázku (Obrázek 7-11) je záznam měření na části sochoru za čtyř různých podmínek. Dopravník se pohyboval vždy rychlostí 10 mm/s, odpovídající naměřené rychlosti pro čtyři situace jsou v první tabulce na stránce 62 v osmém sloupci. Rychlost byla vypočtena z naměřených hodnot delta x a y se započtením rozlišení senzoru 7200 cpi. Tuto rychlost je však potřeba násobit koeficientem závislým na použité optické soustavě a velikosti osvětleného povrchu. Zobrazena jsou data před korekcí.



Obrázek 7-11 – Naměřená data pohybu v ose y za čtyř různých podmínek

7.2.2. Ověření závislosti měření na nehomogenitách povrchu

V předešlém měření bylo zjištěno, že naměřená rychlost kolísá v závislosti na vlastnostech povrchu. Je možné, že toto kolísání hodnot je způsobeno nelinearitou použitého dopravníku, proto byla provedena dvě měření při stejné poloze a pohybu dopravníku, avšak se změnou polohy sochoru, aby byla domněnka potvrzena nebo vyloučena. Každé z těchto měření bylo provedeno třikrát a průměry byly porovnány mezi sebou. Srovnání nabízí Obrázek 7-12. Dopravník se posunul o cca 20 cm, snímané místo bylo osvětleno na ploše 12,5 mm^2. Mezi kolísáním hodnot jednotlivých měření není patrná žádná závislost, lze tedy odvodit, že toto kolísání je závislé na vlastnostech povrchu, a není způsobeno nelinearitou povrchu.



Obrázek 7-12 – Srovnání měření na různých částech sochoru

Totožným způsobem bylo provedeno měření se zvětšenou plochou osvětlenou laserem na 32 mm^2. Ani při tomto měření nebyla shledána shoda ve vzoru kolísání hodnot a ověřilo se tak předchozí tvrzení.

7.2.3. Měření na ocelovém sochoru

Nastavení jednotlivých prvků zařízení je nutné optimalizovat pro měření na ocelových sochorech podobných sochoru, na kterém probíhaly předchozí měření, aby bylo zařízení použitelné v průmyslovém prostředí. V předchozích měřeních bylo zjištěno, že senzor reagoval na tento povrch lépe než na hladké povrchy. Toto měření bylo provedeno také za účelem odhalení vlivu osvětlení v měřící místnosti na měření. Bylo provedeno měření třech rychlostí lineárního dopravníku, na kterém byl instalován sochor –

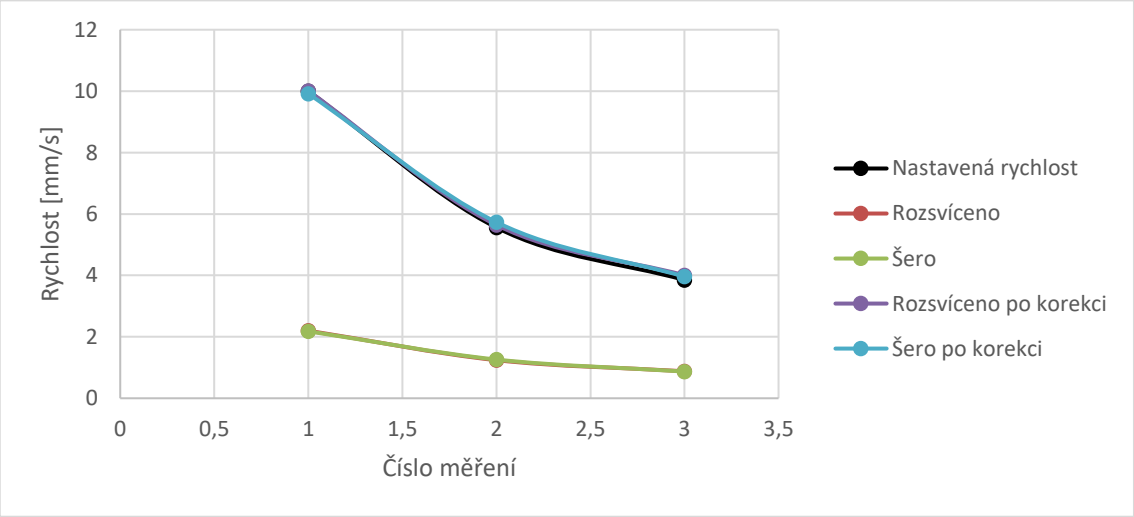
10 mm/s, 5,6 mm/s a 3,9 mm/s. Svazek laserového světla byl v prvním ze dvou případů konvergentní a osvětloval cca 1 mm^2, ve druhém případě divergentní a osvětloval plochu cca 32 mm^2. Dále byl při měření testován vliv osvětlení v místnosti.

Přestože je možné vypočíst příčné zvětšení použité optické soustavy, je patrné, že měřenou rychlost ovlivňuje také velikost oblasti osvětlené laserem. Proto je nutné vypočíst koeficient pro přepočet závislý na těchto dvou parametrech. Z tabulky (Tabulka 7-6) je vidět, že při osvětlení 32 mm^2 povrchu nemá na měření vliv okolní osvětlení v místnosti, zatímco při osvětlení pouze 1 mm^2 ano. Stejně jako u senzoru ADNS-3080 se tato velikost zdá být optimální.

Tabulka 7-6 – Naměřená rychlost pro tři různě nastavené rychlosti dopravníku

Osvětlená plocha [mm^2]	Osvětlení místnosti	Číslo měření		
		1	2	3
1	Rozsvíceno	5,6	2,8	1,89
	Šero	4,45	2,35	1,54
32	Rozsvíceno	2,2	1,24	0,88
	Šero	2,18	1,26	0,87

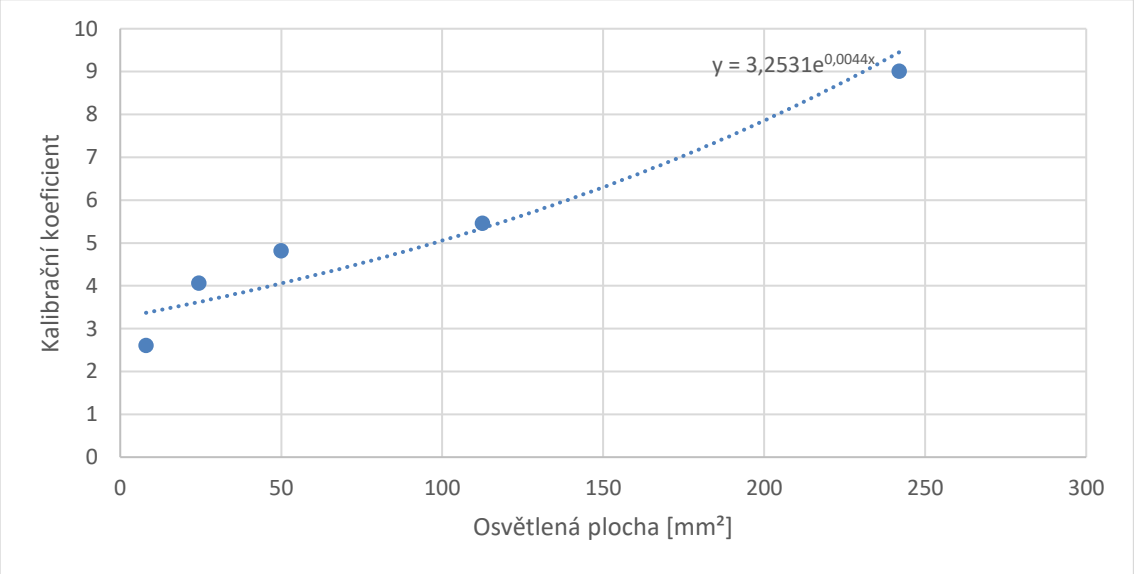
Na grafu s vyznačenou nastavenou rychlostí dopravníku jsou vyznačeny hodnoty naměřené v osvětlené i zatemněné místnosti a jejich korekce pomocí vypočteného koeficientu k = 4,55.



Obrázek 7-13 – Graf nastavených a naměřených rychlostí

7.2.4. Kalibrace pro měření na ocelovém sochoru

Na sochoru bylo provedeno měření při pěti různých velikostech osvětlené plochy povrchu. Pro tři různé rychlosti bylo provedeno vždy deset měření. Z měření rychlosti 10 mm/s byl vypočten koeficient potřebný pro získání nastavené rychlosti. Proložením těchto hodnot závislých na velikosti osvětlené plochy byla získána kalibrační křivka. Do rovnice kalibrační křivky byly dosazeny příslušné velikosti osvětlené plochy a vynásobením naměřené rychlosti byla získána rychlost po kalibraci. Kalibrační křivka však dobře neprokládá hodnoty jednotlivých kalibračních koeficientů a dochází k velkým chybám. Tímto způsobem tedy není možné kalibraci provádět a je vhodnější použít pevné nastavení laseru na určitou plochu povrchu a kalibrační koeficient použít pevný pro dané nastavení.



Obrázek 7-14 – Křivka kalibračních koeficientů

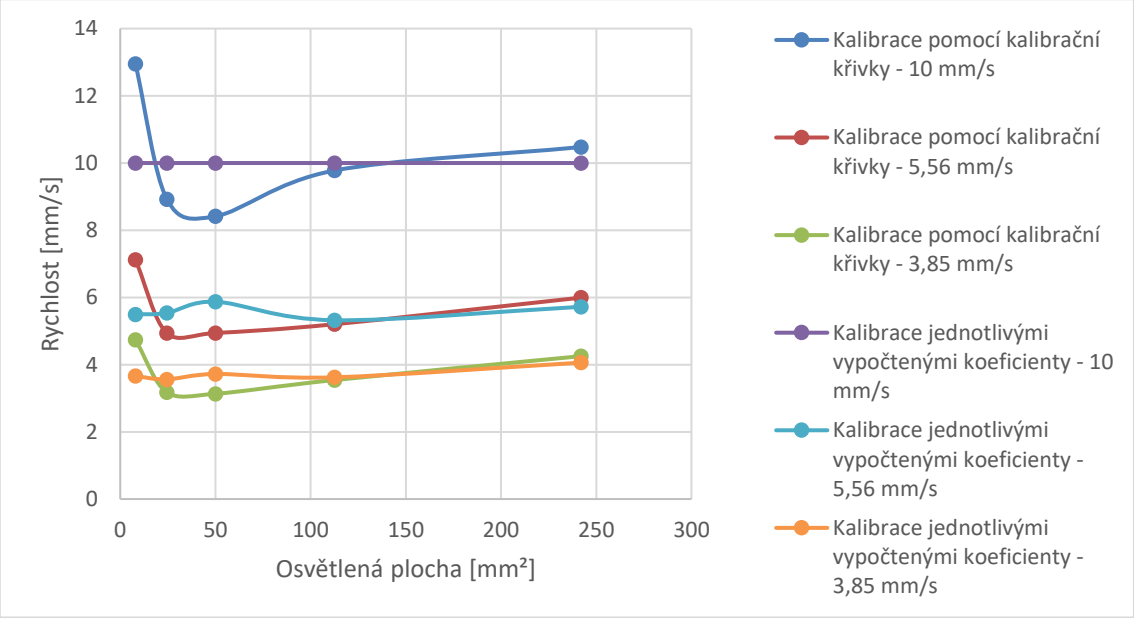
Tabulka 7-7 – Naměřená rychlost po kalibraci pomocí kalibrační křivky

Plocha [mm^2]	Naměřené rychlosti při 10 mm/s	Naměřené rychlosti při 5,56 mm/s	Naměřené rychlosti při 3,85 mm/s	Relativní chyba měření při 10 mm/s	Relativní chyba měření při 5,56 mm/s	Relativní chyba měření při 3,85 mm/s
8	12,95	7,12	4,74	29,49	28,01	23,08
24,5	8,93	4,94	3,18	10,74	11,11	17,43
50	8,42	4,94	3,13	15,84	11,10	18,64
112,5	9,78	5,20	3,54	2,23	6,43	8,00
242	10,48	5,99	4,25	4,75	7,80	10,45

Tabulka 7-8 – Naměřená rychlost po kalibraci koeficientem pro každé měření zvlášť

Plocha [mm²]	Naměřené rychlosti při 10 mm/s	Naměřené rychlosti při 5,56 mm/s	Naměřené rychlosti při 3,85 mm/s	Relativní chyba měření při 10 mm/s	Relativní chyba měření při 5,56 mm/s	Relativní chyba měření při 3,85 mm/s
8	10,00	5,50	3,66	0,00	1,14	4,95
24,5	10,00	5,54	3,56	0,00	0,42	7,50
50	10,00	5,87	3,72	0,00	5,63	3,32
112,5	10,00	5,32	3,62	0,00	4,29	5,90
242	10,00	5,72	4,06	0,00	2,91	5,44

Z tabulek je vidět, že kalibrace pro různé velikosti osvětlené plochy není možná. Je nutné v zařízení použít jedno konkrétní nastavení a provést kalibraci vhodným koeficientem.



Obrázek 7-15 – Graf hodnot po kalibraci

Obrázek 7-15 znázorňuje naměřené hodnoty po kalibraci koeficienty vypočtenými pro každou velikost osvětlené plochy zvlášť a při použití kalibrační křivky. Je patrné, že při použití konstantní velikosti osvětlení je vhodné vypočíst příslušný koeficient a pro kalibraci jej používat. Jak je vidět z výše uvedených tabulek, po kalibraci na rychlost 10 mm/s vycházelo měření pro dvě další měření s relativní chybou do osmi procent.

8. Diskuse

Vytvořené zařízení je limitováno parametry použitého senzoru, jako je například frekvence snímání obrázků povrchu, od které se odvíjejí další, např. maximální rychlost a zrychlení snímaného povrchu. Přesto je možné zařízení dále doplňovat o funkce, které by mohly zlepšit jeho funkci. Jednou z možností je použití infračerveného laseru a infračerveného filtru pro detektor. V této práci bylo jak při použití senzoru ADNS-3080, tak senzoru ADNS-9800 použito laseru o vlnové délce 650 nm. První zmíněný senzor má na tuto vlnovou délku odezvu cca 98 %. ADNS-9800 má však nejlepší odezvu při vlnových délkách kolem 840 nm, na použité záření má odezvu cca 92 % maximální citlivosti. Při použití infračerveného laseru by tedy zařízení se senzorem ADNS-9800 mělo pracovat lépe. Takovéto lasery jsou však mnohem méně dostupné.

Možným dalším směřováním práce, které by zřejmě vedlo ke zlepšení dosažených výsledků, je například připojení ultrazvukového senzoru pro měření vzdálenosti objektu od tohoto senzoru. Tento doplněk, instalovaný na spodní straně zařízení, by zjišťoval aktuální vzdálenost přístroje od měřeného objektu, která se v závislosti na podmínkách může měnit. V nejjednodušším případě by tato informace sloužila pouze k upozornění, že vzdálenost neodpovídá nastavení, a o této skutečnosti by například akustickým signálem byla informována obsluha, která by problém vyřešila. Při změně vzdálenosti by nebyl optický senzor zaostřen správně na povrch a vznikaly by nepřesnosti. Vzdálenost od měřeného objektu také ovlivňuje zvětšení vytvořené optickou soustavou, proto je nutné znát ji pro výpočet. Avšak i pokud bychom neuvažovali rozostření, nestačilo by adaptivně měnit algoritmus pro výpočet rychlosti, neboť samotnou konstrukcí krytu je zařízení nastaveno na určitou vzdálenost. V našem případě je to konkrétně 30 cm, což je dáno úhlem, pod kterým použitý laser osvětluje povrch měřeného objektu.

Závěr

Úvodní část této práce je věnována teoretickému seznámení s optickými senzory. Druhá část popisuje experimentální pracoviště a práci s vybraným senzorem ADNS-9800 a senzorem ADNS-3080. Bylo popsáno, jak je možné pomocí elektronické platformy Arduino Mega2560 připojeným senzorem měřit rychlost povrchu pod ním a jak získat obraz snímaného povrchu. S touto sestavou bylo provedeno experimentální měření pro určení rychlosti valníku. Rozdíly mezi naměřenými hodnotami a reálnými hodnotami byly malé, což potvrdilo možnost použití pro měření rychlosti.

Pro umožnění měření rychlosti pohybujících se povrchů byly instalovány a testovány různé optické soustavy, jak k senzoru ADNS-3080, tak senzoru ADNS-9800. Pro konstantní vzdálenost 30 cm mezi pohybujícím se povrchem a rychloměrem, byly vybrány čočky s ohniskovou vzdáleností 4,2 mm a 50 mm, vzhledem k rozměrům jednotlivých senzorů a vhodně malé konstrukce výsledného zařízení. Jako externí zdroj pro osvětlení měřeného povrchu byl použit laser, dále bylo testováno také LED osvětlení.

V programu Autodesk Inventor byly navrženy mechanické konstrukce pro zařízení, do kterých byly jednotlivé komponenty instalovány. Vzájemná poloha senzoru a laseru zajišťuje, že jsou výsledná zařízení vhodná pro měření ze vzdálenosti 30 cm.

S dvěma vytvořenými zařízeními byla provedena experimentální měření. Pro zařízení založeném na snímání povrchu senzorem ADNS-3080, byly jako nejvhodnější povrchy určeny povrchy s malou absorpcí světla. Senzor ADNS-9800, využívaný v herních optických myších, má lepší parametry, primárně frekvenci snímání, od čehož se odvíjí maximální rychlost povrchu, jakou senzor dokáže správně zaznamenat. Proto je pro použití ve vytvořeném zařízení vhodnější. Nejlepší reakce na pohyb snímaného povrchu byly u tohoto senzoru hliníková páska, černý papír a ocelový sochor. Právě na ocelovém sochoru bude probíhat měření v průmyslu, pokud se zařízení osvědčí. Doposud totiž měření v průmyslovém prostředí nebylo provedeno, z důvodu rekonstrukce haly ve Třineckých železárnách, kde se mělo testování uskutečnit.

Náklady na výrobu jednotlivých zařízení se odvíjejí od použitých komponent. Vývojové prostředí Arduino IDE je open-source projekt a v práci využitý program MATLAB ve výsledném zařízení není potřebný. Díky tomu nejsou náklady na zařízení nijak velké. V obou případech byla použita elektronická platforma Arduino Mega2560, červený laser o výkonu 5 mW a byla vyrobena mechanická konstrukce. Deska Mega2560 vyjde na českých internetových obchodech na necelých 500 Kč, při objednání například přímo

z Asie vyjde tento prvek na necelou polovinu. Podobně to platí u laseru. Použitý laser je u nás dostupný za cca 150 Kč. Množství materiálu na výrobu krabičky je dostupné za přibližně 100 Kč. ADNS-3080, který byl v této práci použit, je dostupný za cca 650 Kč. Spolu se samotným senzorem ADNS-9800 byly použity také části počítačové myši A4tech Bloody Terminator TL8, proto byla koupena jako celek za přibližně 900 Kč. Dále byl k zařízení připojen displej v hodnotě cca 250 Kč. Celkové náklady na výrobu zařízení založeném na senzoru ADNS-3080 dosahovaly výše 1400 Kč. Naopak celkové náklady na výrobu zařízení se senzorem ADNS-9800 se pohybovaly okolo 1900 Kč.

Literatura

[1] AGILENT TECHNOLOGIES, INC. *Agilent ADNS-2051: Optical mouse sensor*. 2003, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-2051-Avago-datasheet-32784417.pdf

[2] AGILENT TECHNOLOGIES, INC. *T-1$^{3/4}$ (5 mm) Precision Optical Performance AlInGaP LED Lamps: Technical Data.* 2004. Dostupné také z: https://www.digchip.com/datasheets/parts/datasheet/021/HLMP-ED80-XX000-pdf.php

[3] Arduino MEGA 2560 & Genuino MEGA 2560. *Arduino* [online]. 2016 [cit. 2016-12-29]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[4] ARDUINO-SHOP.CZ. *IIC I2C OLED display 0,96" 128x64 Bílý* [online]. [cit. 2017-04-23]. Dostupné z: http://arduino-shop.cz/arduino/1569-iic-i2c-oled-display-0-96-128x64-bily-1487765029.html

[5] ARDUINO-SHOP.CZ. *Optical Flow Sensor APM2.5 Multicopter ADNS 3080 Optický Senzor pro Arduino* [online]. [cit. 2017-05-14]. Dostupné z: http://arduino-shop.cz/arduino/1383-optical-flow-sensor-apm2-5-multicopter-adns-3080-opticky-senzor-pro-arduino-1472722976.html

[6] ATMEL CORPORATION. *ATmega640/1280/1281/2560/2561*. 2014, [cit. 2016-12-27]. Dostupné také z: http://www.microchip.com/wwwproducts/en/ATmega2560

[7] AVAGO TECHNOLOGIES. *ADNS-2120: Solid-State Optical Mouse Lens.* 2008, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-2120-Avago-datasheet-10309025.pdf

[8] AVAGO TECHNOLOGIES. *ADNS-3080: High-Performance Optical Mouse Sensor*. 2008, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-3080-Avago-datasheet-10310392.pdf

[9] AVAGO TECHNOLOGIES. *ADNS-6000: Laser Mouse Sensor*. 2008, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-6000-Avago-datasheet-10314636.pdf

[10] AVAGO TECHNOLOGIES. *ADNS-9500: LaserStream™ Gaming Sensor.* 2011, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-9500-Avago-datasheet-10311751.pdf

[11] AVAGO TECHNOLOGIES. *ADNS-9800: LaserStream™ Gaming Sensor.* 2012, [cit. 2017-05-03]. Dostupné také z: http://datasheet.octopart.com/ADNS-9800-Avago-datasheet-10666463.pdf

[12] BRZICOVÁ, M. a E. ANDRIANTSARAZO. *Myší princip* [online]. Praha, 2013 [cit. 2016-10-16]. Dostupné z: http://fyzsem.fjfi.cvut.cz/2012-2013/Zima12/proc/mysi.pdf. Fakulta jaderná a fyzikálně inženýrská.

[13] BUTBAIA, Gigi. *Arduino Tutorial: Get Traveled Distance Using ADNS-9800 Laser Mouse Sensor*. In: Instructables [online]. [cit. 2017-05-11]. Dostupné z: http://www.instructables.com/id/Arduino-Tutorial-ADNS-9800-Laser-Mouse-Traveled-Di/

[14] ČÍP, Ondřej a Zdeněk BUCHTA. *Přesné měření délek pomocí laserové interferometrie* [online]. 2011 [cit. 2017-05-14]. Dostupné z: http://www.crr.vutbr.cz/system/files/brozura_06_1110.pdf.

[15] FILLAMENTUM. *PLA Extrafill: Datasheet.* Czech Republic. Dostupné také z: https://www.dropbox.com/s/xhzxtbtarhqax5z/datasheet_pla_extrafill.pdf?dl=0

[16] FONT, Davinia, Marcel TRESANCHEZ, Tomàs PALLEJÀ, Mercè TEIXIDÓ a Jordi PALACÍN. Characterization of a Low-Cost Optical Flow Sensor When Using an External Laser as a Direct Illumination Source. *Sensors*. 2011, 11(12), 11856-11870. DOI: 10.3390/s111211856. ISSN 1424-8220. Dostupné také z: http://www.mdpi.com/1424-8220/11/12/11856/

[17] HW Kitchen. *Arduino Mega 2560 Rev3* [online]. 2015 [cit. 2017-05-14]. Dostupné z: http://www.hwkitchen.com/products/arduino-mega-2560-rev3/

[18] KAMPHUIS, W. P. H. *Using optical mouse sensors for sheet position measurement*. Eindhoven, 2007. Traineeship report. Technische Universiteit Eindhoven.

[19] Laser surface velocimeter. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-10-16]. Dostupné z: https://en.wikipedia.org/wiki/Laser_surface_velocimeter

[20] Laserové diody 2 - Typy a struktury laserových diod. *Elektrorevue* [online]. Olomouc, 2001 [cit. 2016-10-16]. Dostupné z: [1] http://www.elektrorevue.cz/clanky/01043/index.html

[21] LaserSpeed LS4000. *Limab* [online]. 2016 [cit. 2016-10-16]. Dostupné z: http://www.limab.co.uk/product/laserspeed-ls4000/

[22] *Logitech* [online]. 2016 [cit. 2016-10-16]. Dostupné z: http://www.logitech.com

[23] Measurement Principle. *Elovis* [online]. Germany [cit. 2016-10-16]. Dostupné z: http://www.elovis.de/en/produkte/produkt_2/messprinzip.html

[24] *Optical Mice and How They Work.* www.logitech.com [online]. 2010, 2 [cit. 2016-10-16]. Dostupné z: http://www.digikey.com/en/pdf/b/broadcom/optical-mice-how-they-work

[25] PIXART IMAGING INC. *PMW3310DH-AWQT: Low Power LED Gaming Mouse Sensor*. Dostupné také z: http://www.pixart.com.tw/upload/PMW3310DH-AWQT_NNDS_02022016_20160902193420.pdf

[26] PIXART IMAGING INC. *PMW3320DB-TYDU: Entry-Gaming Optical Navigation Sensor*. Dostupné také z: http://www.pixart.com/upload/PMW3320DB-TYDU_NNDS_02022016_20160902193714.pdf

[27] PIXART IMAGING INC. *PMW3360DM-T2QU: Optical Gaming Navigation Sensor.* Dostupné také z: http://www.pixart.com/upload/PMS0058-PMW3360DM-T2QU-NNDS-R1.30-06042016_20160902201411.pdf

[28] PIXART IMAGING INC. *SDNS-3988: High Performance Gaming Sensor*. Dostupné také z: http://www.pixart.com.tw/upload/SDNS3988_NNDS_02022016_20160902193457.pdf

[29] ULTIMAKER. *Ultimaker 3 Extended: Specification sheet.* [cit. 2017-05-11] Dostupné také z: https://d2py9w124w2itd.cloudfront.net/src/media/data/pdf/products/um3/Ultimaker_3_Extended_specification_sheet.pdf

[30] VANĚK, O. *Komparativní studie měření teploty těla různými technologiemi*. Brno, 2015. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce prof. Ing. Ivo Provazník, Ph.D.

[31] VODA, Z. *Průvodce světem Arduina* [online]. [cit. 2016-12-29]. Dostupné z: https://drive.google.com/file/d/0B5k9VLyI1vUoTWVndVhub3J6ZUU/edit

Příloha

A. Skript pro měření rychlosti počítačové myši přepočtem z pozice kurzoru – MATLAB

```
condition = 1;
x1 = []; y1 = []; v1 = [];
while condition
  x = [];
  y = [];
    for i = 1:2
       loc= get(0, 'PointerLocation');
       fprintf('X:%d Y:%d\n', loc(1), loc(2));
       x(i) = loc(1);
       y(i) = loc(2);
       if i == 1;
           p = 0.05;
           pause(p); % zdržení programu mezi dvěma změřenými body
       end
    end
  dx = x(2)-x(1); % vzdálenost mezi dvěma změřenými body v ose x
  dy = y(2)-y(1); % vzdálenost mezi dvěma změřenými body v ose y
  vx = dx/t; % výpočet rychlosti kurzoru v ose x [pixel/s]
  vy = dy/t; % výpočet rychlosti kurzoru v ose y [pixel/s]
  a = 382.98; % velikost monitoru v mm v ose x (uživatel doplní sám podle použitého
monitoru, přepočet např. na http://axofiber.no-ip.org/inside/pixel.size.en.htm)
  b = 215.43; % velikost monitoru v mm v ose y
  s = get(0, 'screensize'); % získání informace o rozlišení monitoru
  pix = a/s(3); % velikost pixelu
  PPmm = s(3)/a; % pixels per millimeter - počet pixelů na jeden milimetr
  PPI = PPmm*25.4; % pixels per inch - počet pixelů na jeden palec
  CPI = 1000; % counts per inch - rozlišení myši
  k = CPI/PPI; % koeficient pro přepočet rychlosti kurzoru na rychlost myši
  vx = vx/k; % přepočet rychlosti kurzoru na rychlost myši v ose x
  vy = vy/k; % přepočet rychlosti kurzoru na rychlost myši v ose y
  vx = round(vx*pix); % rychlost myši v milimetrech za sekundu v ose x
  vy = round(vy*pix); % rychlost myši v milimetrech za sekundu v ose y
  % if (vx | vy) ~= 0     % vypisovat pouze pokud nastal pohyb
  v = round(sqrt(vx^2+vy^2));
  fprintf('vx:%d \t\t vy:%d \t\t celková rychlost:%d\n', vx, vy, v);
  % end
  x1(1,end+1) = vx;  y1(1,end+1) = vy;  v1(1,end+1) = v;    % záznam hodnot
end
```

B. Kód pro měření rychlosti – ADNS-9800 – Arduino IDE

```
#include <SPI.h>
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "registry_adns9800.h"
#include "srom_adns9800.h"

byte initComplete = 0;
volatile int xydat[2];
volatile byte movementFlag = 0;
const int NCS = 53;
volatile int interupt3_flg = 0;

byte read_reg(byte reg_addr) {          // Funkce pro čtení z adresáře
  digitalWrite(NCS, LOW);               // Aktivace pinu chip select
  SPI.transfer(reg_addr & 0x7f );       // Poslání adresy registru s MSB = 0, jako
indikace čtení
  delayMicroseconds(100);               // Pozastavení programu na čas tSRAD
  byte data = SPI.transfer(0);          // Načtení dat
  delayMicroseconds(1);        // Pozastavení na čas tSCLK-NCS pro operaci čtení (120 ns)
  digitalWrite(NCS, HIGH);              // Deaktivace pinu chip select
```

```
  delayMicroseconds(19);               // Pozastavení na čas tSRW/tSRR (20us) od načtení
dat (mínus zdržení tSCLK-NCS)
  return data;                         // Vrácení hodnoty data
}

void write_reg(byte reg_addr, byte data) {         // Funkce pro zápis do adresáře
  digitalWrite(NCS, LOW);
  SPI.transfer(reg_addr | 0x80 );                  // Poslání adresy registru s MSB = 1,
jako indikace zápisu
  SPI.transfer(data);                              // Zapsání dat
  delayMicroseconds(20);      // Pozastavení na čas tSCLK-NCS pro operaci zápisu (20 us)
  digitalWrite(NCS, HIGH);
  delayMicroseconds(100);     // Pozastavení na čas tSWW/tSWR (120us) mínus tSCLK-NCS
}

void firmware() {                              // Funkce pro nahrání firmware
  write_reg(REG_Configuration_IV, 0x02);       // Zapsání 0x02 do registru
Configuration_IV pro výběr velikosti SROM 3 K
  write_reg(REG_SROM_Enable, 0x1d);            // Zapsání 0x1d do registru SROM_Enable
pro inicializaci
  delay(10);
  write_reg(REG_SROM_Enable, 0x18);            // Zapsání 0x18 do registru SROM_Enable
pro stažení SROM
  digitalWrite(NCS, LOW);
  SPI.transfer(REG_SROM_Load_Burst | 0x80);        // Zapisování SROM souboru do
SROM_Load_Burst registru, první data musí začínat adresou registru
  delayMicroseconds(15);
  unsigned char c;
  for (int i = 0; i < firmware_length; i++) {      // Zapsání celého firmware
    c = (unsigned char)pgm_read_byte(firmware_data + i);
    SPI.transfer(c);
    delayMicroseconds(15);
  }
  digitalWrite(NCS, HIGH);
}

void startup(void) {                           // Funkce pro zajištění správného startu
  digitalWrite(NCS, LOW);                      // Ujištění, že sériový port je resetován
  digitalWrite(NCS, HIGH);
  write_reg(REG_Power_Up_Reset, 0x5a);         // Zapsání 0x5a do registru Power_Up_Reset
pro resetování čipu
  delay(50);
  read_reg(REG_Motion);                        // Přečtení registru Motion, který zastaví
hodnoty v následujících 4 registrech
  read_reg(REG_Delta_X_L);
  read_reg(REG_Delta_X_H);
  read_reg(REG_Delta_Y_L);
  read_reg(REG_Delta_Y_H);
  firmware();                                  // Nahrání firmware
  delay(100);
  write_reg(REG_Configuration_I, 0x09);        // Nastavení rozlišení senzoru, hodnota
0x09 odpovídá 1800 cpi
  delay(100);
  byte laser_ctrl0 = read_reg(REG_LASER_CTRL0);       // Zjištění hodnoty registru
LASER_CTRL0, abychom bity 7, 6, 5, 4 nastavili na původní hodnotu
  write_reg(REG_LASER_CTRL0, laser_ctrl0 & 0xf0 );    // Nastavení laseru do norláního
módu (bit 0 = 0, bity 3,2,1 = 000b)
  delay(1);
}

void UpdatePointer(void) {               // Funkce pro přečtení dat o pohybu, pokud
nastal pohyb (interrupt na pinu Motion)
  if (initComplete == 1) {               // Podmínka, zda proběhl setup a čip je nastaven
    digitalWrite(NCS, LOW) ;
    xydat[0] = (int)read_reg(REG_Delta_X_L) ;  // Načtení informace o pohybu v ose x
    xydat[1] = (int)read_reg(REG_Delta_Y_L) ;  // Načtení informace o pohybu v ose y
    digitalWrite(NCS, HIGH) ;
    movementFlag = 1;                          // Indikace, že pohyb byl detekován
  }
}

int z2dop(int b) {                             // Funkce pro převod z dvojkového doplňku
  if (b & 0x80) {
    b = -1 * ((b ^ 0xff) + 1) ;
  }
  return b;
}
```

```
ISR(TIMER3_COMPA_vect) {                    // Interrupt Service Routine - spustí se
vždy, když časovač přeteče
  interupt3_flg = 1;
}

void setup() {
  noInterrupts(); // Vypnutí interruptů, kvůli nastavení časovače 3 (TIMER3)
  TCCR3A = 0;     // Nastavení celého registru "Timer/Counter Control Register 3 A" na 0
  TCCR3B = 0;
  OCR3A = 6249;             // Doba časovače: počet=((požadovaný čas)/(rozlišení
časovače))-1 = ((0,1)/(1/((16*10^6)/256)))-1
  TCCR3B |= (1 << WGM32);   // Spuštění CTC módu
  TCCR3B |= (1 << CS32);    // Nastavení CS12 na 1, tedy použití dělení časovače 256
  TIMSK3 |= (1 << OCIE3A);  // Aktivace časovacího interruptu
  interrupts();             // Povolení interruptů

  Serial.begin(115200);
  pinMode (2, INPUT);       // Nastavení pinu 2 (interrupt) jako vstup
  pinMode (50, INPUT);      // Nastavení pinu 50 (MISO) jako vstup

  attachInterrupt(0, UpdatePointer, FALLING);            // Nastavení interruptu

  SPI.begin();              // Inicializace SPI linky nastavením SCK, MOSI a NCS jako
výstupy, nastavením SCK a MOSI na low a NCS na high
  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE3));      // Nastavení
komunikace pomocí SPI

  startup();         // Zavolání funkce pro start senzoru
  delay(100);
  initComplete = 1;
}

int deltaX = 0;
int deltaY = 0;

void loop() {
  if (interupt3_flg == 1) {               // Podmínka pro časování, pokud platí, vypiš
pohyb v osách x a y. Děje se každých 100 ms
    interupt3_flg = 0;
    Serial.print( deltaX );
    Serial.print( ", ");
    Serial.println( deltaY );

    deltaX = 0;
    deltaY = 0;
  }
  if (movementFlag) {                     // Pokud nastal pohyb a došlo k jeho přečtení,
dojde ke sčítání hodnoty, po čas nastavený pomocí interruptu
    int dX = z2dop(xydat[0]) ;
    int dY = z2dop(xydat[1]) ;
    deltaX = deltaX + dX;
    deltaY = deltaY + dY;
    movementFlag = 0;
  }
}
```

C. Skript nacitani.m pro získání rychlosti – ADNS-3080 a ADNS-9800 – MATLAB

```
clear all; close all; clc
instrreset; % Odpojení a vymazání všech objektů
seriak = serial('COM5', 'BaudRate', 115200, 'DataBits', 8, 'StopBits', 1, 'Terminator',
13, 'FlowControl', 'none','InputBufferSize',8192); % Nastavení sériového portu
fopen(seriak); % Připojení sériového portu
pause(1);
x1 = [0];
y1 = [0];
squal = [0];
shutter = [0];
while(1)
    xy = fscanf(seriak);  % Načtení dat ze sériového portu
    ii = findstr(xy,','); % Rozdělení načtených dat do složek x a y
```

```
    jj = findstr(xy,'.');
    kk = findstr(xy,'/');
    x= str2num(xy(1:(ii-1)));
    y= str2num(xy((ii+1):(jj-1)));
    sq = str2num(xy((jj+1):(kk-1)));
    sh = str2num(xy((kk+1):end));
    squal(end+1,1) = sq;
    shutter(end+1,1) = sh;
    x= str2num(xy(1:(ii-1)));
    y= str2num(xy((ii+1):end));
    if (isempty(x)), x1(end+1,1) = 1111; % Ošetření chyb
    else x1(end+1,1) = x; end % Uložení informace o pohybu v ose x do vektoru
    if (isempty(y)), y1(end+1,1) = 1111;
    else y1(end+1,1) = y; end
    fprintf('%2.2f %2.2f\n', x, y) % Vypsání hodnot pohybu v ose x a y
end
fclose(seriak); % Odpojení sériového portu
```

D. Skript zpracovani.m pro zpracování dat ze senzoru ADNS-3080 a ADNS-9800 - MATLAB

```
close all; clear all; clc;
load('merenixy.mat') % Načtení signálu
%% Filtrace signálu
N = length(y1);
y1(y1 == 1111) = 0; % Odstranění chybných vzorků
x1(x1 == 1111) = 0; % Odstranění chybných vzorků
y = y1;
y(y<7) = 0; % Výběr pouze kladných hodnot větších než 6
y = medfilt1(y,5); % Filtrace mediánovým filtrem setříděním pěti vzorků

figure(1)
hold on
plot(y1,'g')
title('Originální a filtrovaný signál')
plot(y)
hold off
legend('Originální signál','Signál po filtraci')

%% Nalezení nástupné a sestupné hrany
% Určení celistvých úseků kde byl detekován pohyb
ii = zeros(size(y1));
for k = 5:(N-5)
    pred = sum(y((k-4):(k))~=0); % Počet vzorků různých od nuly v y(k-4) až y(k)
    po = sum(y((k):(k+4))~=0);   % Počet vzorků různých od nuly v y(k) až y(k+4)
    if (pred>=4)
        ii(k) = 1;
    end
    if (po>=4)
        ii(k) = 1;
    end
end

% Určení nástupné a sestupné hrany
nastup = find(diff(ii)==1);
sestup = find(diff(ii)==-1);
nastup = nastup+4;  % Odstranění přechodového děje
sestup = sestup-8;
pocet_mereni = length(nastup);

figure(2);
hold on
plot((1:N)/10,y,'b',(1:N)/10,ii,'r')
plot(nastup/10, zeros(size(nastup)),'*g')
plot(sestup/10, zeros(size(sestup)),'og')
xlabel('Čas [s]','FontSize',20)
ylabel('Posunutí [bod]','FontSize',20)
set(gca,'FontSize',20)
hold off
legend('Filtrovaný signál','Detekce pohybu','Nástupná hrana','Sestupná hrana')


%% Vypočtení rychlosti
```

```
CPI = 1800; % Rozlišení senzoru
fprintf('krok \t impulsy y \t impulsy x \t rychlost \n')
for o = 1:pocet_mereni
    y_soucet(o) = sum(y1(nastup(o):sestup(o))); % Součet impulzů v jednom měření v ose y
    x_soucet(o) = sum(x1(nastup(o):sestup(o))); % Součet impulzů v jednom měření v ose x
    cas(o) = sestup(o)-nastup(o);               % Počet impulzů jednoho měření pro
výpočet času (1 impulz = 0,1 s)

    gama(o) = sqrt((x_soucet(o)^2)+(y_soucet(o)^2));    % Výpočet pohybu v rovině xy

%     vy(o) = ((y_soucet(o)*25.4)/CPI)/(cas(o)/10);     % Rychlost v mm/s v ose y
    v(o) = ((gama(o)*25.4)/CPI)/(cas(o)/10);    % Výpočet rychlosti v mm/s

    fprintf('%d \t\t %5.0d \t\t %5.0d \t\t z5.4f\n', o,y_soucet(o),x_soucet(o),v(o));
end
```

E. Kód pro získání obrazu – ADNS-9800 – Arduino IDE

```
#include <SPI.h>
#include <avr/pgmspace.h>
#define Motion                  0x02
#define Configuration_I         0x0f
#define Power_Up_Reset          0x3a
#define LASER_CTRL0             0x20
#define Frame_Capture           0x12
#define Pixel_Burst             0x64
const int NCS = 53;                     // Definování pinu chip select
unsigned char Foto[900] ;               // Vektor pixelů obrazu

byte read_reg(byte reg_addr) {          // Funkce pro čtení z adresáře
  digitalWrite(NCS, LOW);               // Aktivace pinu chip select
  SPI.transfer(reg_addr & 0x7f );       // Poslání adresy registru s MSB = 0, jako
indikace čtení
  delayMicroseconds(100);               // Pozastavení programu na čas tSRAD
  byte data = SPI.transfer(0);          // Načtení dat
  delayMicroseconds(1);         // Pozastavení na čas tSCLK-NCS pro operaci čtení (120 ns)
  digitalWrite(NCS, HIGH);              // Deaktivace pinu chip select
  delayMicroseconds(19);                // Pozastavení na čas tSRW/tSRR (20us) od načtení
dat (mínus zdržení tSCLK-NCS)
  return data;                          // Vrácení hodnoty data
}

void write_reg(byte reg_addr, byte data) {    // Funkce pro zápis do adresáře
  digitalWrite(NCS, LOW);
  SPI.transfer(reg_addr | 0x80 );             // Poslání adresy registru s MSB = 1, jako
indikace zápisu
  SPI.transfer(data);           // Zapsání dat
  delayMicroseconds(20);        // Pozastavení na čas tSCLK-NCS pro operaci zápisu (20 us)
  digitalWrite(NCS, HIGH);
  delayMicroseconds(100);       // Pozastavení na čas tSWW/tSWR (120us) mínus tSCLK-NCS
}

void startup(void) {                          // Funkce pro zajištění správného startu
  digitalWrite(NCS, LOW);                     // Ujištění, že sériový port je resetován
  digitalWrite(NCS, HIGH);
  write_reg(Power_Up_Reset, 0x5a);            // Zapsání 0x5a do registru Power_Up_Reset
pro resetování čipu
  delay(100);
  write_reg(Configuration_I, 0x09);           // Nastavení rozlišení senzoru, hodnota
0x09 odpovídá 1800 cpi
  delay(100);

  byte laser_ctrl0 = read_reg(LASER_CTRL0);       // Zjištění hodnoty registru
LASER_CTRL0, abychom bity 7, 6, 5, 4 nastavili na původní hodnotu
  write_reg(LASER_CTRL0, laser_ctrl0 & 0xf1);     // Vypnutí laseru - bit 0 = 1
(Nastavení laseru do norláního módu (bit 0 = 0, bity 3,2,1 = 000b))
  delay(1);
}

void setup(){
  Serial.begin(115200);
  SPI.begin();              // Inicializace SPI linky nastavením SCK, MOSI a NCS jako
výstupy, nastavením SCK a MOSI na low a NCS na high
```

```
  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE3));      // Nastavení
komunikace pomocí SPI
  pinMode (2, INPUT);       // Nastavení pinu 2 jako vstup
  pinMode (50, INPUT);      // Nastavení pinu 50 (MISO) jako vstup
  startup();
  delay(100);
}
void loop(){
   digitalWrite(NCS, HIGH);
   delay(1);

// Zapsání 0x93 do registru Frame_Capture
   digitalWrite(NCS, LOW);               // Aktivování NCS
   SPI.transfer(Frame_Capture | 0x80 );  // Poslání MSBit = 1 pro indikaci procesu write
   SPI.transfer(0x93);                   // Poslání dat
   delay(1);

// Zapsání 0xc5 do registru Frame_Capture
   SPI.transfer(Frame_Capture | 0x80 );
   SPI.transfer(0xc5);
   delay(10);

// Ověření bitu 0 registru Motion
   bool cekej = true ;
   while(cekej){
      SPI.transfer(Motion & 0x7f );             // Poslání MSBit = 0 s adresou Motion
pro indikaci čtení
      delayMicroseconds(100);
      byte motion = SPI.transfer(0);            // Načtení dat registru Motion
      //Serial.println(motion,BIN) ;
      if((motion & 0x01)!=0) cekej = false ;    // Čekání, pokud je nultý bit 0, pokud
je 1, je dostupný první pixel obrazu a program může pokračovat
      delay(100);
     }
// Načítání 900 pixelů obrázku
   for(int loop=0;loop<900;loop++){
      SPI.transfer(Pixel_Burst & 0x7f );
      delayMicroseconds(500);
      Foto[loop] = SPI.transfer(0);
   }
  Serial.println(10000);

// Vypsání hodnot všech 900 pixelů
   for(int loop=0;loop<900;loop++)  {
     Serial.println(Foto[loop]);
     //if(loop!=899) Serial.print(",");
   }
   digitalWrite(NCS, HIGH);
   delay(10);
}
```

F. Skript pro načtení a vykreslení obrazu – ADNS-9800 – MATLAB

```
clear all; close all; clc; instrreset;
seriak = serial('COM5', 'BaudRate', 115200, 'DataBits', 8, 'StopBits', 1, 'Terminator',
'CR/LF', 'FlowControl', 'none','InputBufferSize',8192);
fopen(seriak);
pause(1);
while(1)
    if(fscanf(seriak,'%d')==10000)
        for k = 1:30
            for l = 1:30
                obraz(k,l) = fscanf(seriak,'%d'); ;
            end
        end                                          % Přečtení sériového portu
    end
%     obraz=rot90(obraz,3);                          % Rotace obrazu o 270°
    figure(1)
    axes('Position',[0 0 1 1])
    a= max(obraz(:)); if (a==0), a=0.1;end
    imshow(obraz./a);
    drawnow;
end
fclose(seriak);
```

G. Kód pro měření rychlosti v tzv. Burst módu – ADNS-9800 – Arduino IDE

```
#include <SPI.h>
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "srom_adns9800.h"
#include "U8glib.h"                 // připojení knihovny U8glib
U8GLIB_SSD1306_128X64 OLED(U8G_I2C_OPT_NONE); // inicializace OLED displeje z knihovny
U8glib

#define Configuration_I          0x0f
#define SROM_Enable              0x13
#define LASER_CTRL0              0x20
#define Configuration_IV         0x39
#define Power_Up_Reset           0x3a
#define Motion_Burst             0x50
#define SROM_Load_Burst          0x62

const int NCS = 53;
volatile int interupt3_flg = 0;

byte read_reg(byte reg) {              // Funkce pro čtení z adresáře
  digitalWrite(NCS, LOW);              // Aktivace pinu chip select
  SPI.transfer(reg & 0x7f );  // Poslání adresy registru s MSB = 0, jako indikace čtení
  delayMicroseconds(100);              // Pozastavení programu na čas tSRAD
  byte data = SPI.transfer(0);         // Načtení dat
  delayMicroseconds(1);       // Pozastavení na čas tSCLK-NCS pro operaci čtení (120 ns)
  digitalWrite(NCS, HIGH);             // Deaktivace pinu chip select
  delayMicroseconds(19);               // Pozastavení na čas tSRW/tSRR (20us) od načtení
dat (mínus zdržení tSCLK-NCS)

  return data;                         // Vrácení hodnoty data
}

void write_reg(byte reg, byte data) {  // Funkce pro zápis do adresáře
  digitalWrite(NCS, LOW);
  SPI.transfer(reg | 0x80 );  // Poslání adresy registru s MSB = 1, jako indikace zápisu
  SPI.transfer(data);         // Zapsání dat
  delayMicroseconds(20);      // Pozastavení na čas tSCLK-NCS pro operaci zápisu (20 us)
  digitalWrite(NCS, HIGH);
  delayMicroseconds(100);     // Pozastavení na čas tSWW/tSWR (120us) mínus tSCLK-NCS
}

void firmware() {                      // Funkce pro nahrání firmware
  write_reg(Configuration_IV, 0x02);   // Zapsání 0x02 do registru Configuration_IV pro
výběr velikosti SROM 3 K
  write_reg(SROM_Enable, 0x1d); // Zapsání 0x1d do registru SROM_Enable pro inicializaci
  delay(10);
  write_reg(SROM_Enable, 0x18); // Zapsání 0x18 do registru SROM_Enable pro stažení SROM

  digitalWrite(NCS, LOW);
  SPI.transfer(SROM_Load_Burst | 0x80);        // Zapisování SROM souboru do
SROM_Load_Burst registru, první data musí začínat adresou registru
  delayMicroseconds(15);

  unsigned char c;
  for (int i = 0; i < firmware_length; i++) {      // Zapsání celého firmware
    c = (unsigned char)pgm_read_byte(firmware_data + i);
    SPI.transfer(c);
    delayMicroseconds(15);
  }
  digitalWrite(NCS, HIGH);
}

void startup(void) {                        // Funkce pro zajištění správného startu
  digitalWrite(NCS, HIGH);                  // Ujištění, že sériový port je resetován
  digitalWrite(NCS, LOW);
  write_reg(Power_Up_Reset, 0x5a);          // Zapsání 0x5a do registru Power_Up_Reset
pro resetování čipu
  delay(50);
  read_reg(0x02);
  read_reg(0x03);
  read_reg(0x04);
  read_reg(0x05);
```

```
  read_reg(0x06);
  firmware();                                // Nahrání firmware
  delay(100);

  write_reg(Configuration_I, 0x24);          // Nastavení rozlišení senzoru, hodnota
0x09 odpovídá 1800 cpi, 0x24 = 7200 cpi
  delay(100);

  byte laser_ctrl0 = read_reg(LASER_CTRL0);       // Zjištění hodnoty registru
LASER_CTRL0, abychom bity 7, 6, 5, 4 nastavili na původní hodnotu
  write_reg(LASER_CTRL0, laser_ctrl0 & 0xf0 );    // Nastavení laseru do normálního módu
(bit 0 = 0, bity 3,2,1 = 000b)

  delay(100);
}

int z2dop(int d){                            // Funkce pro převod z dvojkového doplňku
  if (d & 0x8000){
    d = -1 * ((d ^ 0xffff) + 1);
  }
  return d;
}

ISR(TIMER3_COMPA_vect){                      // Interrupt Service Routine - spustí se,
vždy, když časovač přeteče
  interupt3_flg = 1;
}

void setup(){
  noInterrupts();           // Vypnutí interruptů, kvůli nastavení časovače 3 (TIMER3)
  TCCR3A = 0;     // Nastavení celého registru "Timer/Counter Control Register 3 A" na 0
  TCCR3B = 0;

  OCR3A = 6249;             // Doba časovače: počet=((požadovaný čas)/(rozlišení
časovače))-1 = ((0,1)/(1/((16*10^6)/256)))-1
//  OCR3A = 31249;
  TCCR3B |= (1 << WGM32);   // Spuštění CTC módu
  TCCR3B |= (1 << CS32);    // Nastavení CS12 na 1, tedy použití dělení časovače 256
  TIMSK3 |= (1 << OCIE3A);  // Aktivace časovacího interruptu
  interrupts();             // Povolení interruptů

  pinMode (2, INPUT);       // Nastavení pinu 2 (interrupt) jako vstup
  pinMode (50, INPUT);      // Nastavení pinu 50 (MISO) jako vstup

  Serial.begin(115200);
  SPI.begin();              // Inicializace SPI linky nastavením SCK, MOSI a NCS jako
výstupy, nastavením SCK a MOSI na low a NCS na high
  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE3));      // Nastavení
komunikace pomocí SPI

  startup();                // Zavolání funkce pro start sezoru
}

unsigned int xdat;
unsigned int ydat;
unsigned int Burst[14];
unsigned int shutter;
int deltaX = 0;
int deltaY = 0;
int gama = 0;
int v = 0;

void loop(){
  if (interupt3_flg == 1){                  // Podmínka pro časování, pokud platí,
vypíše pohyb v osách x a y. Děje se každých 100 ms
    interupt3_flg = 0;
    Serial.print( deltaX );
    Serial.print( ", ");
    Serial.print( deltaY );
    //Serial.print( ", ");
    //Serial.println(Burst[6]);

    Serial.print( ". ");
    Serial.print(Burst[6]);

    Serial.print( "/ ");
    Serial.println(shutter);
```

```
    gama = sqrt((sq(deltaX))+(sq(deltaY)));
    v = (gama*25.4*3.8/7200)/0.1;        // (DPI * 25,4*kalibrační_koeficient/CPI)/čas

    OLED.firstPage();
    do {
       OLED.setFont(u8g_font_unifont);
      /* OLED.setPrintPos(10,20);
       OLED.print("Rychlost_x: ");
       OLED.setPrintPos(100,20);
       OLED.print(deltaX);
       OLED.setPrintPos(10,40);
       OLED.print("Rychlost_y: ");
       OLED.setPrintPos(100,40);
       OLED.print(deltaY);*/
       OLED.setPrintPos(10,40);
       OLED.print("v =");
       OLED.setPrintPos(45,40);
       OLED.print(v);
       OLED.setPrintPos(85,40);
       OLED.print("mm/s");
       }
    while( OLED.nextPage() );

    deltaX = 0;
    deltaY = 0;
    }

  digitalWrite(NCS, LOW) ;
  SPI.transfer(Motion_Burst & 0x50 );
  delayMicroseconds(100);
  for(int loop=0;loop<14;loop++){
    Burst[loop] = SPI.transfer(0);
  }

  digitalWrite(NCS, HIGH) ;
  delayMicroseconds(1);       // tBEXIT = 500 ns
  xdat = Burst[3];     // Zařazení bajtů DELTA_X_L a DELTA_X_H za sebe do jedné proměnné
  xdat = xdat<<8;
  xdat|= Burst[2];
  ydat = Burst[5];
  ydat = ydat<<8;
  ydat|= Burst[4];
  shutter = Burst[10];        // Zařazení bajtů shutter za sebe do jedné proměnné
  shutter = shutter<<8;
  shutter|= Burst[11];
  int dX = z2dop(xdat) ;
  int dY = z2dop(ydat) ;
  deltaX = deltaX + dX;
  deltaY = deltaY + dY;
}
```

H. Kód pro měření rychlosti – ADNS-3080 – Arduino IDE

```
#include <SPI.h>                          // Zahrnutí knihovny SPI
#define Motion_Burst          0x50        // Definování registru pro Motion Burst mód
#define Configuration_bits    0x0A

const int RES = 48;                       // Pin 48 - reset
const int NCS = 53;                       // Pin 53 - chip select
volatile int interupt3_flg = 0;           // Definování proměnné pro časovač

void setup() {

  noInterrupts(); // Vypnutí interruptů, kvůli nastavení časovače 3 (TIMER3)
  TCCR3A = 0;     // Nastavení celého registru "Timer/Counter Control Register 3 A" na 0
  TCCR3B = 0;

  OCR3A = 6249;             // Doba časovače: počet=((požadovaný čas)/(rozlišení
časovače))-1 = ((0,1)/(1/((16*10^6)/256)))-1
  TCCR3B |= (1 << WGM32);   // Spuštění CTC módu
  TCCR3B |= (1 << CS32);    // Nastavení CS12 na 1, tedy použití dělení časovače 256
  TIMSK3 |= (1 << OCIE3A);  // Aktivace časovacího interruptu
  interrupts();             // Povolení interruptů
```

```
  Serial.begin(115200);     // Spuštění komunikace rychlostí 115200 baudů
  SPI.begin();              // Inicializace SPI linky nastavením SCK, MOSI a NCS jako
výstupy, nastavením SCK a MOSI na low a NCS na high
  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE3));      // Nastavení
komunikace pomocí SPI

  pinMode(NCS, OUTPUT);     // Nastavení pinu chip select na výstup
  pinMode(RES, OUTPUT);     // Nastavení pinu reset na výstup
  reset();                  // Zavolání funkce reset
  delay(50);

  byte nastaveni_laseru = read_reg(Configuration_bits);               // Přečtení
registru pro nastavení
  write_reg(Configuration_bits, nastaveni_laseru | 0x10);             // Nastavení
rozlišení na 1600
  delay(50);
}

void write_reg(byte reg, byte data) {  // Funkce pro zápis do adresáře
  digitalWrite(NCS, LOW);
  SPI.transfer(reg | 0x80 ); // Poslání adresy registru s MSB = 1, jako indikace zápisu
  delayMicroseconds(75);
  SPI.transfer(data);        // Zapsání dat
  delayMicroseconds(1);      // Pozastavení na čas tSCLK-NCS pro operaci zápisu (120 ns)
  digitalWrite(NCS, HIGH);
  delayMicroseconds(50);              // Pozastavení na čas tSWW/tSWR (50us)
}

byte read_reg(byte reg) {             // Funkce pro čtení z adresáře
  digitalWrite(NCS, LOW);             // Aktivace pinu chip select
  SPI.transfer(reg & 0x7f );          // Poslání adresy registru s MSB = 0, jako
indikace čtení
  delayMicroseconds(50);              // Pozastavení programu na čas tSRAD
  byte data = SPI.transfer(0);        // Načtení dat
  delayMicroseconds(1);       // Pozastavení na čas tSCLK-NCS pro operaci čtení (120 ns)
  digitalWrite(NCS, HIGH);            // Deaktivace pinu chip select
  delayMicroseconds(20);
  return data;                        // Vrácení hodnoty data
}

unsigned int Burst[7];
unsigned int shutter;
byte xdat;
byte ydat;
int deltaX = 0;
int deltaY = 0;

void loop() {

    if (interupt3_flg == 1) {     // Podmínka pro časování, pokud platí, je vypsán pohyb
v osách x a y. Děje se každých 100 ms
    interupt3_flg = 0;
    Serial.print( deltaX );
    Serial.print( ", ");
    Serial.print( deltaY );
    Serial.print( ". ");
    Serial.print(Burst[3]);
    Serial.print( "/ ");
    Serial.println(shutter);

    deltaX = 0;
    deltaY = 0;
    }

  digitalWrite(NCS, LOW);                 // Aktivace pinu chip select
  SPI.transfer(Motion_Burst & 0x7f );     // Poslání adresy registru s MSB = 0, jako
indikace čtení
  delayMicroseconds(50);                  // Pozastavení programu na čas tSRAD
     for(int loop=0;loop<7;loop++){
      Burst[loop] = SPI.transfer(0);      // Načtení sedmi dostupných bitů
    }

  delayMicroseconds(1);       // Pozastavení na čas tSCLK-NCS pro operaci čtení (120 ns)
  digitalWrite(NCS, HIGH);                // Deaktivace pinu chip select
  delayMicroseconds(4);                   // tBEXIT

  xdat = Burst[1];
```

```
  ydat = Burst[2];
  int dX = z2dop(xdat) ;                    // Převod z dvojkového doplňku
  int dY = z2dop(ydat) ;
  deltaX = deltaX + dX;
  deltaY = deltaY + dY;
  shutter = Burst[4];            // Zařazení bajtů shutter za sebe do jedné proměnné
  shutter = shutter<<8;
  shutter|= Burst[5];
}

ISR(TIMER3_COMPA_vect) {         // Interrupt Service Routine - spustí se vždy, když
časovač přeteče
  interupt3_flg = 1;
}

int z2dop(int d) {               // Funkce pro převod z dvojkového doplňku
  if (d & 0x80) {
    d = -1 * ((d ^ 0xff) + 1) ;
  }
  return d;
}

void reset(void) {               // Funkce pro resetování senzoru
  digitalWrite(RES, HIGH);
  delayMicroseconds(250);
  digitalWrite(RES, LOW);
  delay(35);                     // tPU-RESET - 35 ms
}
```

I. Kód pro získání obrazu – ADNS-3080 – Arduino IDE

```
#include <SPI.h>
#define Frame_Capture         0x13
#define Pixel_Burst           0x40
const int RES = 48;
const int NCS = 53;
volatile int interupt3_flg = 0;
unsigned int frame[1536];

void setup() {
  Serial.begin(115200);              // Spuštění komunikace rychlostí 115200 baudů
  SPI.begin();          // Inicializace SPI linky nastavením SCK, MOSI a NCS jako
výstupy, nastavením SCK a MOSI na low a NCS na high
  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE3));          // Nastavení
komunikace pomocí SPI
  pinMode(NCS, OUTPUT);
  pinMode(RES, OUTPUT);
  digitalWrite(NCS, HIGH);
}

void loop() {
  delayMicroseconds(10) ;                // Zpoždění nutné před načtením nového obrázku
  digitalWrite(NCS, LOW);                // Aktivování NCS
  delayMicroseconds(1);                  // tNCS-SCLK 120 ns
  SPI.transfer(Frame_Capture | 0x80);    // Poslání adresy s MSB = 1 pro indikaci zápisu
  SPI.transfer(0x83);        // Zapsání hodnoty 0x83 do registru Frame_Capture pro
připravení obrázku
  delayMicroseconds(1);      // Pozastavení na čas tSCLK-NCS pro operaci zápisu (120 ns)
  digitalWrite(NCS, HIGH);
  delayMicroseconds(10);                           // tCAPTURE (10 us)
  digitalWrite(NCS, LOW);                          // Aktivování NCS
  SPI.transfer(Pixel_Burst & 0x7f);
  delayMicroseconds(50);                 // Pozastavení programu na čas tSRAD (50 us)

  for(int loop=0;loop<1536;loop++){
    frame[loop] = SPI.transfer(0);
    delayMicroseconds(10);               // tLOAD (10 us)
  }
  digitalWrite(NCS, HIGH);               // Deaktivace pinu chip select
  delayMicroseconds(4);                  // tBEXIT (4 us)
  for(int i=0;i<1536;i++){               // Kontrolování postupně načítaných pixelů
    if (frame[i] & 0x40){         // Pokud má bit 6 hodnotu 1, je to první pixel obrázku
      frame[i] = frame[i] & 0xbf;      // Vynulování bitu 6, který značí začátek obrázku
      for (int j=0;j<900;j++){
```

```
        frame[j+i] = frame[j+i] & 0x3f;           // Vynulování MSB
        Serial.println(frame[i+j]);               // Vypsání 900 pixelů
      }
      break;                            // Ukončení hledání, když byl nalezen první pixel
    }
  }
delay(600);
}
```

J. Skript pro načtení a vykreslení obrazu – ADNS-3080 – MATLAB

```
clear all; close all; clc; instrreset;
seriak = serial('COM5', 'BaudRate', 115200, 'DataBits', 8, 'StopBits', 1, 'Terminator',
'CR/LF', 'FlowControl', 'none','InputBufferSize',8192);
obraz = zeros(30,30);                           % Definování vektoru pro uložení obrazu
fopen(seriak);

while(1)
    for k = 1:30
        for l = 1:30
            obraz(k,l) = fscanf(seriak,'%d');    % Přečtení sériového portu
        end
    end

    obraz = rot90(obraz);                        % Rotace matice o 90°
    obraz = fliplr(obraz);                       % Převrácení matice zleva doprava
    obraz4 = mat2gray(obraz);                    % Převedení matice do stupňů šedi
    figure(1)
    axes('Position',[0 0 1 1])
    imshow(obraz4)
    drawnow;
end
fclose(seriak);
```