Korištenje Arduino mikrokontrolera za analizu gibanja

Margetić, Zoran

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Zoran Margetić

Korištenje Arduino mikrokontrolera za analizu gibanja

Diplomski rad

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER: NASTAVNIČKI

Zoran Margetić

Diplomski rad

Korištenje Arduino mikrokontrolera za izvođenje pokusa u fizici

Voditelj diplomskog rada: izv. prof. dr. sc. Emil Tafra

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2024.

Želio bih se zahvaliti mentoru izv. prof. dr. sc. Emilu Tafri na strpljivosti i savjetima pri izradi ovog diplomskog rada. Isto tako bih se htio zahvaliti profesoricama Katarini Jeličić i Karolini Matejak Cvenić na pomoći i susretljivosti tijekom izvođenja pokusa za ovaj rad. Također, zahvaljujem se svojoj obitelji, što su me poticali i trpili tijekom mog studiranja.

Sažetak

Ovaj istraživački rad se bavi primjenom Arduino mikrokontrolera u pokusima fizike, a fokus je stavljen na pokuse s gibanjima. U radu se pobliže upoznaje s radom u platformama za programiranje mikrokontrolera. Izradom kôda programiraju se komponente. Koristeći mikrokontroler, akcelerometar, i senzor za udaljenost, dobivamo podatke koje grafički prikazujemo i analiziramo rezultate triju pokusa s pokusnim kolicima. Prvi i drugi pokus uključuju gibanje kolica na kosini pri različitim nagibima kosine, a treći periodičko titranje kolica na ravnoj podlozi. U ovom radu istražujemo potencijal za unapređenje obrazovanja učenika u fizici.

Ključne riječi: Arduino mikrokontroler, pokusi fizike, pokusi s mikrokontrolerima, akcelerometar, ultrazvučni senzor, programiranje

Sadržaj

1 Uvod 2

2 Planiranje pokusa 3

2.1 Hardware 3

2.1.1 Mikrokontroler – Arduino Leonardo 3

2.1.2 ADXL345 Akcelerometar 5

2.1.3 HC-SR04 ultrazvučni senzor 6

2.1.4 Spajanje 7

2.2 Arduino IDE – programiranje mikrokontrolera i senzora 8

2.3 Processing – obrada i crtanje podataka 10

3 Pokusi (gibanje) 12

3.1 Prvi pokus 13

3.1.1 Prvo mjerenje 14

3.1.2 Drugo mjerenje 17

3.1.3 Treće mjerenje 19

3.1.4 Četvrto mjerenje 21

3.1.5 Peto mjerenje - sve preko udaljenosti 22

3.2 Drugi pokus 24

3.2.1 Prvo mjerenje 25

3.2.2 Drugo mjerenje 26

3.3 Treći pokus 27

3.3.1 Prvo mjerenje 29

3.3.2 Drugo mjerenje 30

3.4 Ograničenja i potencijalna poboljšanja 31

4 Motivacija za rad 33

4.1 Uporaba u nastavi 33

4.2 Prednosti 33

4.3 Primjena u cijelom kurikulumu 33

4.4 Međupredmetna povezanost s informatikom 34

5 Zaključak 35

Dodaci 37

Dodatak A – Arduino kôd 37

Dodatak B – Processing kôd 40

Dodatak C – Izmjena Processing kôda za korištenje podataka jednog senzora 45

Literatura 47

1 Uvod

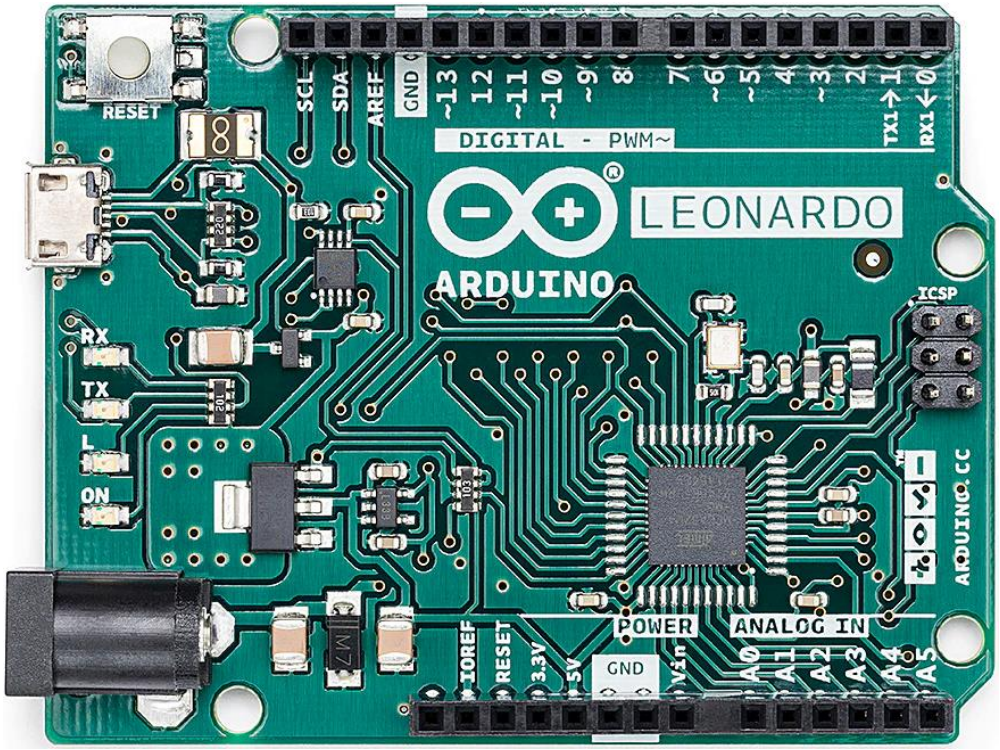
U doba ubrzanog tehnološkog napretka, upotreba Arduino mikrokontrolera postaje sveprisutna u raznim područjima, uključujući i obrazovanje. Ovaj rad istražuje primjenu Arduino mikrokontrolera u fizici, s fokusom na pokuse vezane uz gibanje tijela. Kroz eksperimentalni postav s mikrokontrolerom, akcelerometrom, i senzorom za udaljenost, analizirat ću rezultate pokusa koji uključuju kretanje pokusnih kolica niz kosinu i periodičko titranje na ravnoj podlozi. Osim toga, rad će razmotriti prednosti i mane korištenja mikrokontrolera i senzora. Kroz ovo istraživanje, želimo sagledati potencijal ovih tehnologija za unapređenje nastave fizike u osnovnim i srednjim školama, kao i razmatrati mogućnosti primjene u drugim kontekstima izvan školskog okvira.

2 Planiranje pokusa

2.1 Hardware

2.1.1 Mikrokontroler – Arduino Leonardo

Mikrokontroleri su jedan od osnovnih elemenata moderne digitalne elektronike. Mikrokontroler (MCU) je integrirani strujni krug koji u sebi sadrži mikroprocesor, memoriju, periferne jedinice i druge komponente potrebne za izvršavanje specifičnih zadataka ugrađenih sustava. To je kompaktan računalni sustav koji se koristi za upravljanje elektroničkim uređajima i sustavima. Mikrokontroleri se često koriste u elektronici, robotici, automobilskoj industriji, kućnoj automatizaciji, medicinskim uređajima, i brojnim drugim područjima. Uglavnom se koriste za ugradnju u druge uređaje čime se postiže automatizacija specifičnih funkcija.



Slika 1 - Arduino Leonardo mikrokontroler, Slika preuzeta iz [1]

Glavni dijelovi mikrokontrolera uključuju mikroprocesor koji izvršava programski kôd i obavlja računalne operacije te kontrolira druge komponente; memorije, kao što su ROM (Read-Only Memory) ili trajna memorija koja sadrži trajni programski kôd, RAM (Random Access Memory) ili radna memorija koja privremeno pohranjuje podatke i varijable tijekom izvođenja programa te flash Memory koja se koristi za pohranu programa i podataka te omogućuje ponovno programiranje; periferne jedinice u koje ubrajamo ulazno/izlazne (I/O) jedinice (odnosno pinove) koje omogućuju komunikaciju s vanjskim

uređajima, poput senzora; komunikacijsko sučelje koje omogućuje povezivanje s drugim mikrokontrolerima ili vanjskim uređajima (serijski prijenos podataka, SPI, I2C, ...), te jedinice za mjerenje vremena koje služe za upravljanje vremenskim funkcijama i vremenskim rasporedom događaja, dajući mikrokontroleru sposobnost da kontrolira vremenski raspored raznih operacija (jednostavno rečeno, daju takt); analogno-digitalni pretvornici (ADC) i digitalno-analogni pretvornici (DAC) koji omogućuju mikrokontroleru komunikaciju s analognim uređajima te pretvaranje analognih signala u digitalne i obrnuto.

Mikrokontroleri izvršavaju programski kôd pohranjen u programskoj memoriji. Mikroprocesor čita i izvršava instrukcije, a rezultati se pohranjuju u radnoj memoriji. Ulazno/izlazni sustav omogućava mikrokontroleru interakciju s okolinom putem digitalnih i analognih signala.

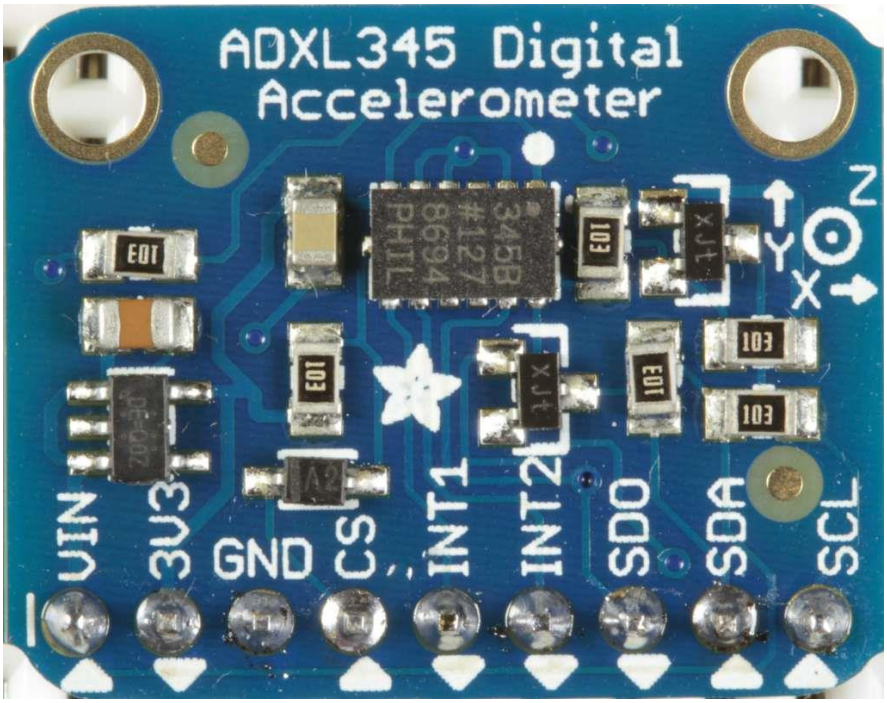
Mikrokontroleri su malih dimenzija pa su pogodni za ugradnju u druge uređaje, ne troše puno energije pa se mogu koristiti i u prijenosnim uređajima koji nemaju konstantan izvor napajanja kao što su mobiteli, prilagodljivi su i imaju širok spektar upotrebe jer im se repogramiranjem funkcionalnost može promijeniti po želji.

Mikrokontrolere se s računalom može povezati putem raznih sučelja. Arduino Leonardo podržava USB (Universal Serial Bus) priključak za programiranje i serijsku komunikaciju s računalom. Postoje još UART (Universal Asynchronous Receiver-Transmitter) koji koristi serijsku komunikaciju preko hardverskog ili virtualnog serijskog porta, te SPI (Serial Peripheral Interface) i I2C (Inter-Integrated Circuit) koji su pogodni za komunikaciju s drugim mikrokontrolerima ili perifernim uređajima. I2C omogućuje da više uređaja međusobno komunicira koristeći zajedničku sabirnicu. Za komunikaciju koristi dvije žice: liniju serijskih podataka (SDA) i liniju serijskog sata (SCL). U ovom radu je korišten USB priključak za spajanje mikrokontrolera s računalom, te I2C komunikacija između mikrokontrolera i senzora.

Postoje različite generacije Arduino mikrokontrolera i za svaku verziju se lako mogu naći specifikacije. Svaka generacija i verzija mikrokontrolera uvodi neku promjenu u funkcionalnosti i značajkama koje imaju. Ako se za izvođenje pokusa koristi neki drugi mikrokontroler treba provjeriti kompatibilnost korištenih biblioteka i moguće promjene u kôdu kako bi sve radilo bez problema. Točne komponente od kojih je sastavljen Arduino Leonardo mikrokontroler koji je korišten u ovom radu, prikazan na slici (Slika *1*), mogu se naći u dokumentaciji na službenoj Arduino web stranici. [2]

2.1.2 ADXL345 Akcelerometar

Akcelerometar je osjetljiv senzor koji mjeri ubrzanje tijela na koje je montiran u 3 osi gibanja. Postoji više vrsta akcelerometara: piezoelektrični, piezootporni, kapacitivni, i dr. U ovom radu je korišten kapacitivni akcelerometar. Takav akcelerometar u sebi sadrži minijaturne kondenzatore. Jedna ploča svakog kondenzatora je učvršćena dok je druga ploča povezana s oprugom. Kada objekt na kojem je senzor ubrzava, ploče kondenzatora se pomiču time mijenjajući kapacitet kondenzatora. Promjene u kapacitetu mjere se i akcelerometar ih pretvara u digitalne vrijednosti koje mikrokontroler može interpretirati. Ova vrsta senzora ima široku primjenu u svakodnevnim uređajima poput pametnih telefona (za okretanje ekrana), igraćih konzola, sportskih uređaja (za praćenje pokreta), i drugih uređaja gdje je praćenje ubrzanja bitno.



Slika 2 - ADXL345 akcelerometar, Slika preuzeta iz [3]

Pri prvoj uporabi treba obratiti pozornost u kojim mjernim jedinicama je akcelerometar postavljen. U većini slučajeva akceleracija se automatski mjeri u g-ovima, odnosno vrijednosti gravitacije. Akcelerometar koji sam koristio već je bio postavljen tako da mjeri u m/s^2. Mjerne jedinice se vrlo lako pretvaraju u kôdu gdje prilikom printanja vrijednosti samo treba pomnožiti s određenim faktorom, na primjer iz g-ova u m/s koristimo faktor 9.81.

U ovom radu je korišten ADXL345 model akcelerometra, prikazan na slici (Slika *2*), no mogao se koristiti bilo koji akcelerometar ADXL3xx serije jer su svi kompatibilni s

korištenim bibliotekama. ADXL345 razlikuje se od drugih senzora u istoj seriji po svojim karakteristikama poput opsega mjerenja i rezolucije. Detalji o ovom senzoru mogu se pronaći u tehničkoj dokumentaciji proizvođača [4] . Ako se koriste drugi mikrokontroler ili akcelerometar treba paziti koji se pinovi koriste te kako promjena pinova utječe na Arduino kôd.

Akcelerometar se povezuje s mikrokontrolerom putem određenih pinova. ADXL345 koristi pinove za napajanje (VCC i GND), te komunikaciju (SDA i SCL). Pinovi SDA i SCL koriste se za I2C komunikacijski protokol. Način na koji su svi dijelovi povezani vidi se na slici 4.

2.1.3 HC-SR04 ultrazvučni senzor

Za mjerenje udaljenosti je korišten ultrazvučni senzor. Ultrazvučni senzori koriste ultrazvučne valove za mjerenje udaljenosti. Takav senzor odašilje ultrazvučni signal koji se širi sve dok ne naiđe na barijeru odnosno tijelo čiju udaljenost mjeri. Signal se odbija od barijere te se dio signala vraća do senzora. Senzor detektira povratni signal s pomoću prijamnika (eng. receiver) te mjeri vrijeme koje je bilo potrebno da se odaslani signal vrati. Ova vrsta senzora često se koristi kod autonomnih vozila u sustavima za izbjegavanje prepreka te za pomoć pri parkiranju.

Slika 3 - HC-SR04 ultrazvučni senzor, slika preuzeta iz [5]

U ovom radu korišten je model HC-SR04, prikazan na slici (Slika *3*). Alternativni modeli ultrazvučnih senzora mogu se razlikovati u preciznosti, brzini odziva i drugim karakteristikama kao što su različiti opsezi mjerenja, kutovi pokrivanja, i slično. Važno je odabrati senzor koji najbolje odgovara namjeni. Također postoje druge vrste senzora za

mjerenje udaljenosti koji rade na drugim principima, no budući da nisu korišteni u ovom radu, neće biti razmatrani. Ako se koristi neki drugi model ultrazvučnog senzora za mjerenje udaljenosti treba provjeriti koja je kompatibilna biblioteka, ako se koristi, te promjene u kôdu koje su potrebne da radi bez problema.

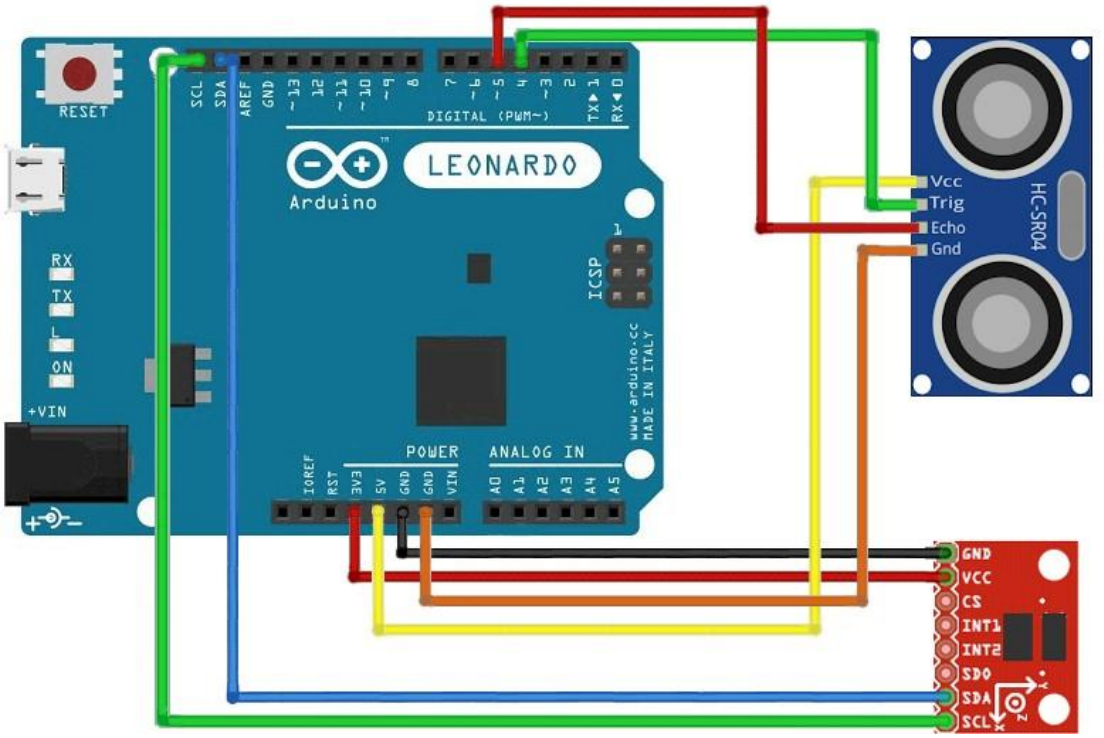
HC-SR04 se sastoji od dva glavna dijela: odašiljača (ultrazvučni predajnik) i prijemnika (ultrazvučni prijemnik). Odašiljač emitira ultrazvučne signale, a prijemnik detektira reflektirane signale. HC-SR04 ima kut pokrivanja od 15 stupnjeva, te je ograničen na mjerenje udaljenosti do maksimalno 4 metra. Maksimalna preciznost senzora je do 3 mm, no tijekom mjerenja sam primijetio da ovaj senzor ne očitava pouzdano promjene manje od 1 cm. Postoje fluktuacije koje se manifestiraju kao šum u podacima te je zbog toga u radu ograničena preciznost rada na promjene od 1 cm i više. Osim toga, treba paziti da frekvencija mjerenja ne bude prevelika jer ovaj senzor može pouzdano mjeriti svakih 0.025 sekundi. Korištenjem veće frekvencije mjerenja može doći do interferencije zvučnih signala te, posljedično, do greške u mjerenju. Frekvencija mjerenja je ograničena brzinom zvuka i maksimalnom udaljenosti mjerenja [6] . Pri biranju eksperimentalne opreme treba obratiti pozornost na to kakav je objekt od kojeg se odbija ultrazvučni signal. U radu je korišten obični karton koji reflektira zvučne signale bez problema. Ako se koriste neki materijali koji zvuk ne reflektiraju dobro, rezultati mogu varirati.

HC-SR04 ima četiri pina: VCC (napajanje), GND (zemlja), Trig (ultrazvučni signal) i Echo (prijemnik reflektiranog signala). Senzor se s mikrokontrolerom povezuje s pomoću digitalnih pinova. HC-SR04 za komunikaciju koristi modulaciju širine impulsa (PWM) kako bi mjerio udaljenost na temelju razlike u vremenu između odaslanih ultrazvučnih signala i povratka od prepreke na senzor. Spojen je preko digitalnih pinova na mikrokontroler te koristi digitalne signale za upravljanje i mjerenje udaljenosti.

2.1.4 Spajanje

Sve je povezano žicama na način prikazan na slici (Slika *4*). Treba pripaziti na koje se pinove spajaju senzori. Digitalni pinovi 2 i 3 na korištenom mikrokontroleru imaju istu funkciju kao pinovi SDA i SCL te se stoga digitalni pinovi 2 i 3 ne mogu koristiti za spajanje ultrazvučnog senzora. Kod drugih mikrokontrolera također treba pripaziti koji pinovi se koriste za što i imaju li dodatne uloge kao što je to ovdje slučaj s pinovima 2 i 3. Također treba paziti da su Arduino kôd i pinovi koji su korišteni usklađeni jer je inače nemoguće dobiti bilo kakve podatke. Breadboard koji je korišten u radu nije prikazan na slici (Slika *4*).

Sam Breadboard nije nužan, ali koristan je da bi cijeli sklop bio uredan te olakšava stabiliziranje i učvršćivanje senzora.



Slika 4 - Shematski prikaz spajanja senzora s mikrokontrolerom

2.2 Arduino IDE – programiranje mikrokontrolera i senzora

U ovom potpoglavlju ću razmotriti softverske aspekte programiranja mikrokontrolera, s posebnim naglaskom na Arduino platformu.

Za programiranje mikrokontrolera i senzora korištena je platforma Arduino IDE (Integrated Development Environment). Kôd u ovom programu (dalje u radu Arduino kôd), služi za odabir postavki senzora i uspostavu konekcije između mikrokontrolera, senzora, i računala. Osim toga služi za kalibraciju te obradu podataka. Neke alternative su platforme poput PlatformIO, MPLAB X IDE, ili Eclipse IDE s odgovarajućim dodacima. No u slučaju njihovog korištenja treba paziti na kompatibilnost kôda i biblioteka.

U ovom radu je korišten Arduino IDE zbog jednostavnosti korištenja. Sučelje je pogodno za početnike. U sučelju se mogu naći alati koji uvelike olakšavaju rad. Neki od tih alata su „Boards Manager“ koji omogućuje dodavanje podrške za različite mikrokontrolere i platforme. Pristupa se preko Tools -> Board -> Boards Manager u alatnoj traci. Velik broj biblioteka (eng. library), odnosno predefiniranih funkcija i rutina koje olakšavaju dodavanje funkcionalnosti kôdu te uključuju i podršku za senzore, zaslone, komunikacijske protokole, i dr., mogu se naći i instalirati u samom sučelju platforme. Tzv. Library Manageru pristupa

se preko Sketch -> Include Library -> Manage Libraries u alatnoj traci. Biblioteke često razvijaju članovi zajednice ili sami proizvođači komponenti te se redovito ažuriraju.

Serial Monitor omogućuje praćenje serijskog komunikacijskog toka između računala i mikrokontrolera. Jednostavnim ispisom na monitor možemo vidjeti podatke koje mjerimo ili uređujemo. Može se pristupiti preko Tools -> Serial Monitor ili u gornjem desnom kutu stisnuti ikonicu s povećalom. Važno je znati da kada je serial monitor uključen, on koristi odabrani Port, te ako želimo kasnije crtati grafove, serial monitor mora biti isključen inače ćemo dobiti grešku „Port is busy“.

Serial Plotter nam daje grafički prikaz podataka dobivenih putem serijske komunikacije. Grafovi se crtaju u stvarnom vremenu. S pomoću Serial.plot() funkcije moguće je izravno crtati grafove iz kôda. No ova značajka je dodana u jednoj od najnovijih verzija te još uvijek ima ograničene mogućnosti. Nije bila dostatna za mjerenja koja su se izvodila u ovom radu pa je bilo potrebno koristiti dodatni program koji crta grafove.

„Sketch“ je naziv koji se koristi za projekte u Arduino IDE-u. Svi podaci, uključujući kôd, biblioteke i postavke, nalaze se u jednoj .ino datoteci.

Arduino IDE dolazi s mnogo ugrađenih primjera koji pokrivaju širok spektar potreba. Pristupa se preko File -> Examples u alatnoj traci. Istraživanje ovih primjera te učenje iz vrlo jednostavnih, kratkih kôdova je brz način za upoznavanje sa sintaksom programskog jezika. Osobno sam se, kao početnik u programiranju, koristio ovime kako bih stekao potrebno znanje za izradu ovog rada.

Jedna od velikih prednosti je i velika aktivna zajednica korisnika koja je dostupna za pomoć i podršku. Službeni Arduino forum pruža mjesto za postavljanje pitanja i razmjenu iskustava. [7]

Osim toga, na službenoj Arduino web stranici postoji i lista s detaljnim opisima funkcija, biblioteka, i kratkim primjerima kôda koji su vrlo korisni ljudima koji nisu upoznati s programskim jezikom koji se koristi. [8] Arduino language (baziran na C/C++) je jednostavan programski jezik s funkcijama za lakšu upotrebu. Moguće je koristiti C ili C++ u sklopu Arduino IDE-a, ali Arduino Language je preferirani programski jezik zbog jednostavnosti.

Arduino IDE verzija na kojoj je rađen ovaj rad je 2.2.1. Cijeli Arduino kôd se može naći u dodacima, pod Dodatak A.

U vezi kôda, imam nekoliko važnih napomena. Vrlo je važno da se ne koristi „delay“ funkcija jer ona pauzira bilo kakve procese koji se događaju u specificiranom intervalu vremena. To znači da čak i ako se postavi da akcelerometar radi mjerenja frekvencijom od 800 Hz, zbog „delay“ funkcije to se neće raditi, nego će se mjerenje raditi svakih onoliko milisekundi koliko je specificirano u delay funkciji. Umjesto toga koristio sam „millis“ funkciju koja nije tzv. „blocking“ funkcija kao što je delay funkcija.

Alpha vrijednost se koristi pri računanju takozvanog eksponencijalnog pomičnog prosjeka (EMA) koji uzima zadnje isprintano mjerenje i pridodaje ga najnovijem mjerenju. Težina kojom se pridodaje zadnje mjerenje određena je faktorom alpha. Ovom tehnikom izglađujemo krivulju koju dobivamo kasnije u grafu, no treba biti oprezan. Ako se postavi premala alpha vrijednost može doći do brojnih problema koje ću pokazati kasnije u radu. Način na koji se računa EMA može se naći u Arduino kôdu.

2.3 Processing – obrada i crtanje podataka

Platforma koja je korištena u ovom radu zove se Processing. Processing je posebno popularan zbog svoje jednostavnosti i fokusa na vizualnu umjetnost. Alternativni alati nude različite prednosti, poput veće snage izražavanja u C++ (OpenFrameworks), web integracije (p5.js), ili specijaliziranosti za interaktivne performanse (TouchDesigner). Odabir ovisi o specifičnim zahtjevima projekta. U mojem slučaju bio je potreban jednostavan alat koji može obraditi podatke, nacrtati ih u grafovima koje sami dizajniramo s pomoću kôda.

Jedini problem na koji sam naišao jest da Processing koristi vlastiti programski jezik temeljen na Java-i. Prilagođen je umjetnicima i dizajnerima te je jednostavan za učenje. No ipak je bilo potrebno upoznati se s dva programska jezika tijekom izrade ovog rada. Processing se prvenstveno koristi vlastitim jezikom, ali može se proširiti na povezivanje s drugim jezicima kao što su JavaScript, Python itd. Službena stranica Processinga sadrži mnogo informacija, primjera i referenci koje mogu pomoći pri izradi željenog kôda. [9]

Za uvoz podataka iz Arduino kôda koristimo „import processing.serial.*;“ koji onda u stvarnom vremenu daje podatke Processing kôdu da crta grafove. Pri tome se treba pobrinuti da brzina prijenosa bude ista kao i u Arduino kôdu te da se koristi odabrani Port na koji je spojen mikrokontroler.

Verzija Processinga na kojoj je rađen ovaj rad je 4.2. Cijeli Processing kôd se može naći u Dodatku B. Tamo se može vidjeti da osim crtanja grafova, u Processingu također računamo brzinu. Brzina se računa preko promjene udaljenosti koju ultrazvučni senzor očitava. Moguće je namjestiti da se brzina računa preko očitanja akceleracije, no u ovom radu to nije obrađeno. I na kraju, u Processing kôdu je dodano da se podaci koji se crtaju na grafovima, spremaju u csv datoteku. Pomoću podataka spremljenih u tu datoteku moguće je, na primjer, izračunati srednju vrijednost akceleracije, kako je to pokazano kasnije u radu. Isto tako koristi se za obradu i crtanje u nekom drugom softwareu, kao što su excel ili qtiplot.

3 Pokusi (gibanje)

U ovom poglavlju prikazujem primjenu mikrokontrolera isprogramiranog za izvođenje pokusa. Pokusi koje izvodim uključuju ubrzano gibanje kako bih pokazao i očitanja akcelerometra, no moguće je izvoditi pokuse s bilo kakvim gibanjima, kao što su jednoliko gibanje ili kružno gibanje. No, u tom slučaju je potrebno prilagoditi Processing kôd ovisno o namjeni korištenja.

Csv datoteke, u koje se spremaju zabilježeni podaci, uređene su da imaju spremljen samo relevantni dio mjerenja. Treba napomenuti da se vremenske oznake ne poklapaju uvijek s onima prikazanim u Processing grafu, odnosno moguće je da su sva mjerenja pomaknuta za određeno vrijeme, no sami intervali su i dalje jednaki stoga taj pomak ne predstavlja problem. Razlog nepoklapanja vremenskih oznaka jest u tome što postoji određena odgoda u crtanju podataka na grafu zbog sporosti programa.

U svim pokusima koje izvodim određene parametre kôda ne mijenjam. To je broj uzoraka potrebnih za početnu kalibraciju „const int numSamples = 500;“. Početnom kalibracijom eliminiraju se očitanja gravitacije.

Iza toga imam dvije linije kôda:

```
accel.writeRegister(ADXL345_REG_BW_RATE, ADXL345_DATARATE_800_HZ);
accel.writeRegister(ADXL345_REG_DATA_FORMAT, ADXL345_RANGE_2_G);
```

Te linije služe za namještanje frekvencije mjerenja te osjetljivost akcelerometra koji koristimo. Za ove postavke ne mogu se postaviti proizvoljne vrijednosti. Vrijednosti koje se mogu postaviti su specificirane u tehničkoj dokumentaciji senzora koji se koristi. [4] Te dvije postavke nisam mijenjao jer prilikom testiranja kôda mijenjanje osjetljivosti nije pokazalo nikakvu značajnu razliku od mjerenja pri standardnoj osjetljivosti od ±2g kod izvođenja tipa pokusa izvođenih u ovom radu. Frekvencija mjerenja je automatski postavljena na 100 Hz te kao takva nije bila dovoljno velika. Uzimanje 800 Hz kao najveću moguću frekvenciju mjerenja bez potrebe mijenjanja kôda se pokazalo sasvim adekvatno. To znači da Arduino radi mjerenje otprilike svakih $1.25*10^{-3}$ s. Maksimalna frekvencija mjerenja koja je specificirana u tehničkoj dokumentaciji ADXL345 akcelerometra je 3200 Hz no ta opcija nije bila kompatibilna s bibliotekom korištenom u kôdu.

3.1 Prvi pokus

Slika 5 – Prikaz postava prvog pokusa

Slika 6 – Prikaz postava prvog pokusa iz drugog kuta

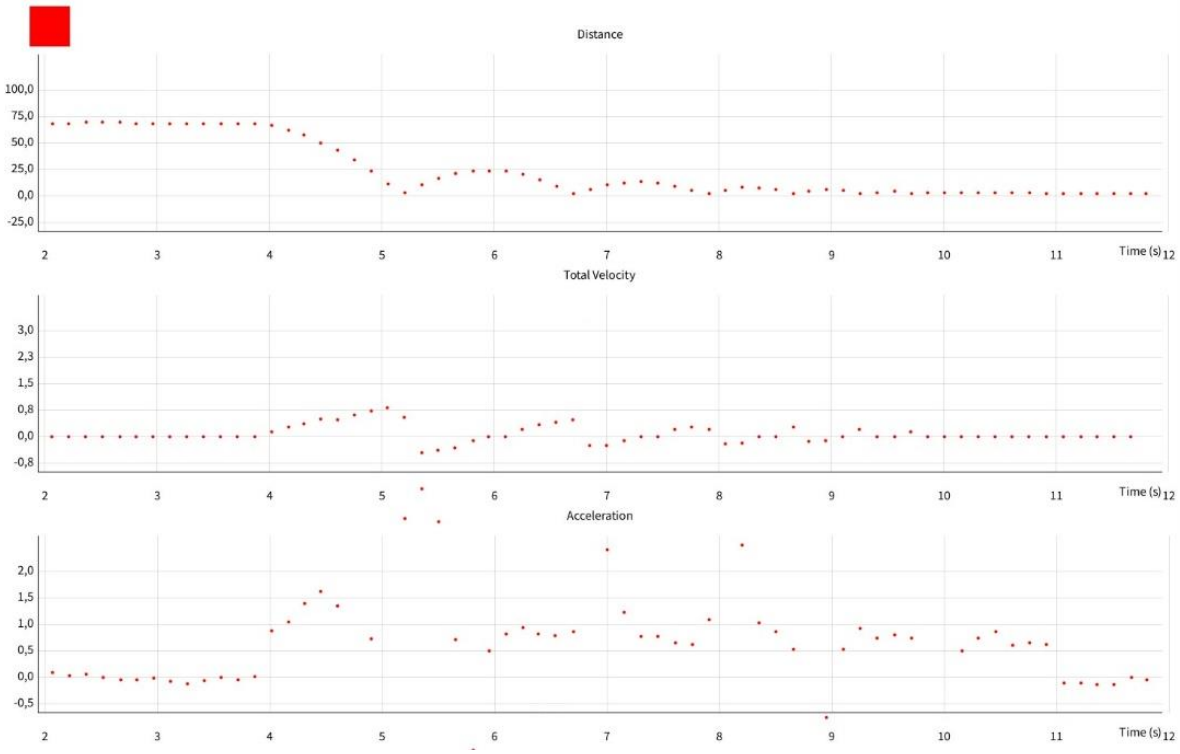
Pribor koji je korišten za izvođenje pokusa su pokusna kolica s oprugom na prednjoj strani te spremnikom za utege, stol, klupa koja se koristi za podizanje jednog kraja stola, branici pričvršćeni na rub stola, karton ili neki drugi zastor kao zvučna barijera, utezi za

kolica, te dvostrana ljepljiva traka. Isto tako se koristi i sav pribor spomenut u prošlom poglavlju: mikrokontroler, senzori, žice potrebne za spajanje i računalo.

Opis Pokusa (Slika *5* i Slika *6*): Kolica su puštena niz nakošeni stol, te se iz stanja mirovanja gibaju i ubrzavaju do prepreke. Odbijaju se od prepreke te se pušta kolica da se gibaju dok se ne vrate u stanje mirovanja, prislonjena uz branike. Akcelerometar i ultrazvučni senzor su montirani na breadboard koji je dobro učvršćen na prednjem dijelu kolica s ljepljivom trakom kako bi se senzori što manje tresli. Senzori su okrenuti tako da y os akcelerometra usklađena sa smjerom gibanja kako bi pojednostavili mjerenje. Tako je dobiveno gibanje u samo jednoj dimenziji odnosno mjeri se samo jedna os akceleracije. Ultrazvučni senzor je okrenut u smjeru gibanja tako da zvučna barijera (karton na slikama Slika *5* i Slika *6*) bude prva stvar koja je detektirana i pokazuje udaljenost do barijere, odnosno do branika. Treba naglasiti da je postav kalibriran u stanju mirovanja, u već nagnutom položaju, kako bih dobio očitanje samo za y os kako je već navedeno. Isto tako je namješteno da je brzina pozitivna kada se udaljenost od kartona smanjuje pošto sam definirao pozitivan smjer gibanja prema branicima, tj. kartonu koji je postavljen kod branika. Interval između crtanja vrijednosti je 150 milisekundi za sva mjerenja u prvom pokusu. U prvom pokusu nagib stola je otprilike 11 stupnjeva.

3.1.1 Prvo mjerenje

U ovom mjerenju puštao sam niz kosinu kolica s dva utega. Postavio sam parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 0.8. U Processing kôdu odredio sam da se za računanje brzine ne detektira promjena udaljenosti jednaka i manja od jednog centimetra. Rezultat prvog mjerenja prikazan je na slici (Slika *7*).



Slika 7 - Rezultati prvog mjerenja

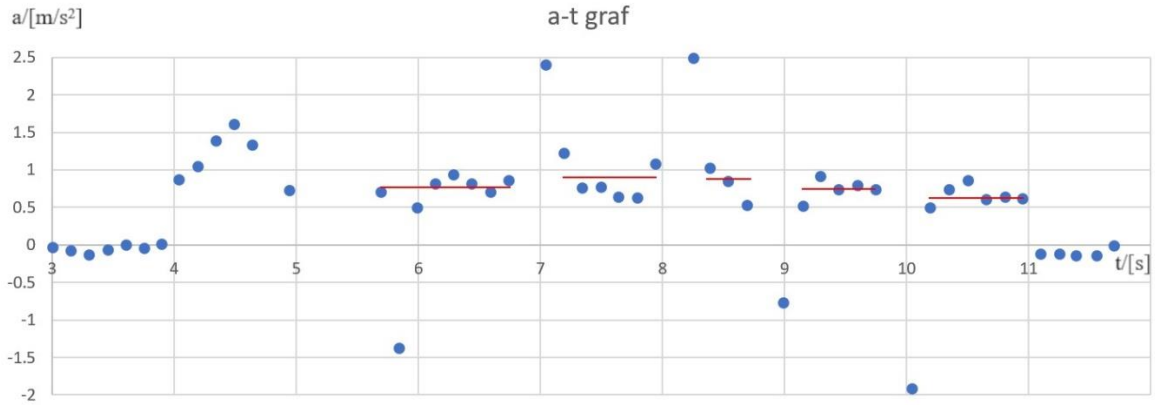
Na s-t grafu se vidi kako krivulja vjerno prikazuje stvarnost gibanja. Sam senzor ima ograničenu preciznost mjerenja pa zato pri sporom gibanju izgleda kao da se kolica ne miču. No to je posljedica već spomenute ograničenosti senzora, relativno velike frekvencije kojom se crtaju točke, te sporosti promjene smjera gibanja samih kolica zbog velike inercije. Samim time na v-t grafu posljedično dobivam da očitanja brzine, koja se računaju preko promjene udaljenosti, budu „stepenasta“. Tim stepenastim mjerenjima pridodaje i činjenica da u kôdu specificiram ako je promjena udaljenosti manja od 1 cm, kôd u grafu crta prijašnju vrijednost brzine.

Iz v-t grafa mogu izračunati grubu vrijednost prosječne akceleracije, pa usporediti to s dobivenim rezultatima iz a-t grafa. U prvom usponu krivulje na v-t grafu vidim da su kolica iz stanja mirovanja ubrzala do 0.8 m/s u intervalu od 1.05 sekundi. Time dobivam grubu procjenu da bi prosječna akceleracija trebala biti oko 0.76 m/s^2.

Što se tiče rezultata na a-t grafu, vrijednosti „divljaju“, pogotovo tijekom prvog spusta niz stol, akceleracija prvih par mjerenja ima trend rasta te potom pada, umjesto da je konstantna. Pretpostavljeni uzrok tome jest, uključujući ali ne i ograničeno na, nesavršenost eksperimentalnog postava kao što su zapinjanje kotača, proklizavanje pokusnih kolica, mali nepredviđeni pomaci i vibracije koje se događaju tijekom mjerenja, te nepričvršćeni utezi

smješteni u kolicima koji mogu poskakivati ili klizati naprijed-nazad pri gibanju kolica. Udarom u branike akceleracija naglo postaje negativna, i ta očitanja se ne vide na grafu nacrtanom u Processingu. Provjerom u csv datoteci mogu se vidjeti točne vrijednosti. Nakon prvog segmenta gibanja, odnosno prvog spusta kolica iz stanja mirovanja do udarca u branike, zabilježeni podaci prilično dobro reflektiraju stvarnost gibanja. Akceleracija je konstantna do trenutka udara u branike gdje akceleracija ponovo ode u veliku negativnu vrijednost.

Zanemarujući očitanja koja ekstremno odstupaju od očekivanih vrijednosti i trenutke udara u branike, računam prosječnu vrijednost akceleracije. Koristim se s podacima iz csv datoteke te radim graf u excelu kako bih usporedio podatke. Prosjek za prvi spust kolica nije računat jer rezultati previše odskaču od očekivanih vrijednosti.



Slika 8 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.76 m/s^2, 0.85 m/s^2, 0.80 m/s^2, 0.74m/s^2, te 0.66 m/s^2

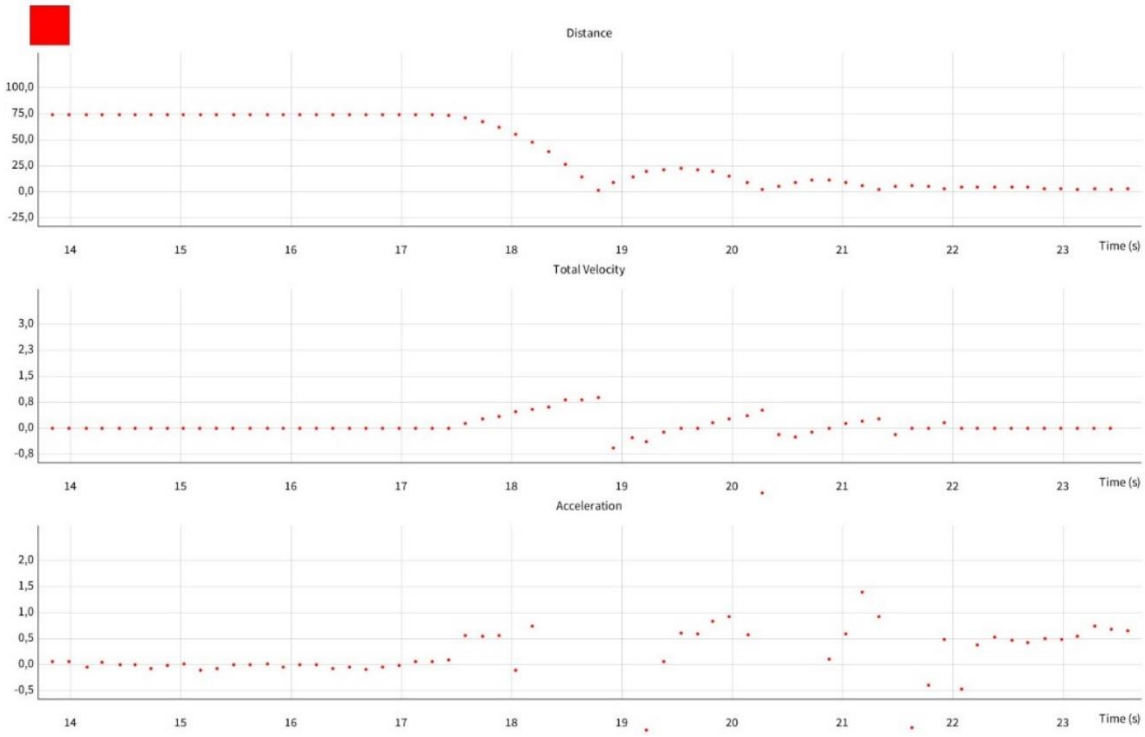
Računajući prosječnu akceleraciju tijekom cijelog gibanja kao aritmetička sredina prosjeka prikazanih na slici (Slika *8*), ne težinski po vremenu, dobijem 0.76 m/s^2 što se savršeno slaže s grubom procjenom iz v-t grafa. Na ovaj način će se računati ukupni prosjeci akceleracije ostalih mjerenja.

Na samom kraju mjerenja, između desete i jedanaeste sekunde, vidi se da i udaljenost i brzina prikazuju kao da kolica miruju dok akceleracija prikazuje da kolica još uvijek konstantno ubrzavaju. To se dogodi kada je ultrazvučni senzor postavljen tako da ne prikazuje male promjene udaljenosti te se iz grafa ne vidi da se kolica još uvijek zapravo gibaju, odnosno odbijaju visokom frekvencijom od prepreke zbog tvrde opruge na prednjem dijelu kolica. Pošto negativnu akceleraciju dobivamo samo u trenutku udara kolica u branike

znamo da su ta mjerenja mnogo kraća nego pozitivna akceleracija koju dobivamo u ostatku gibanja. Prema tome vidimo da je vjerojatnost da akcelerometar očita negativnu vrijednost na kraju mnogo manja nego da očita pozitivnu vrijednost. To objašnjava zašto je akceleracija tu konstantna i pozitivna cijelo vrijeme iako bi realno trebala fluktuirati svakim udarcem kolica u branike.

3.1.2 Drugo mjerenje

U ovom mjerenju puštao sam kolica s dva utega unutra te sam postavio parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 0.6. U Processing kôdu i dalje je zadano da se za računanje brzine ne uzima promjena udaljenosti jednaka i manja od jednog centimetra. Smanjenjem alpha koeficijenta se nadam da će krivulje u a-t grafu biti izglađenije nego u prvom mjerenju, tj. da neće biti toliko skokova u vrijednostima. Rezultat mjerenja se vidi na slici (Slika *9*).



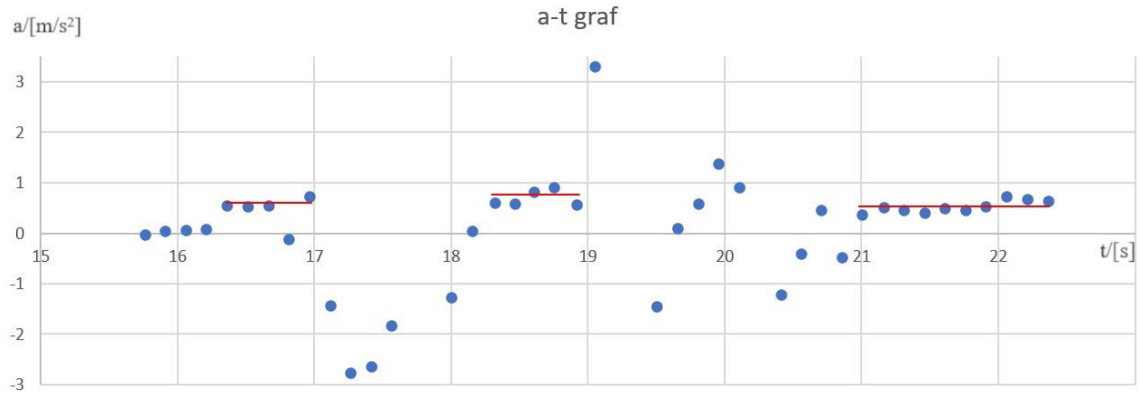
Slika 9 - Rezultati drugog mjerenja

Na s-t grafu imam lijepo očitanje udaljenosti, te samim time i brzine. Što se tiče brzine, vidi se periodičko povećanje brzine nakon svakog udara u branike te je dobivena malo veća maksimalna brzina u usporedbi s prvim mjerenjem. To je zato što sam pustio kolica s veće početne udaljenosti od branika. Osim toga, i dalje imam blagu „stepenastost“ grafa brzine zbog već navedenih razloga.

Iz v-t grafa računam grubu vrijednost prosječne akceleracije, pa uspoređujem s dobivenim rezultatima iz a-t grafa. U prvom usponu krivulje na v-t grafu vidi se da su kolica iz stanja mirovanja ubrzala do 0.85 m/s u intervalu od 1.05 sekundi. Time se dobije gruba procjena da bi prosječna akceleracija trebala biti oko 0.81 m/s^2.

Na početku se vidi lijep prikaz konstante akceleracije sve do trenutka kad očitanja akceleracije postanu ekstremna te izađu iz okvira grafa tik prije udara u prepreku, te sporo povećavanje vrijednosti akceleracije iz negativnih vrijednosti na početnu konstantu vrijednost. Ova sporost u promjeni podataka pojavljuje se zbog postavljanja akcelerometra na početku gdje je u kôdu određeno da se alpha vrijednost od EMA smanji s 0.8 iz prvog pokušaja na 0.6. Smanjenjem vrijednosti alpha izgladio sam dobivene podatke, no ova postavka čini da mi podaci ne reflektiraju stvarnost gibanja jer nisu dovoljno osjetljivi na brze promjene kao što su udarci ili sudari.

Iz csv datoteke mogu vidjeti da akceleracija postane negativna prije prvog udarca u branike. Na a-t grafu se vidi da su realno samo tri intervala gdje je akceleracija relativno konstantna.



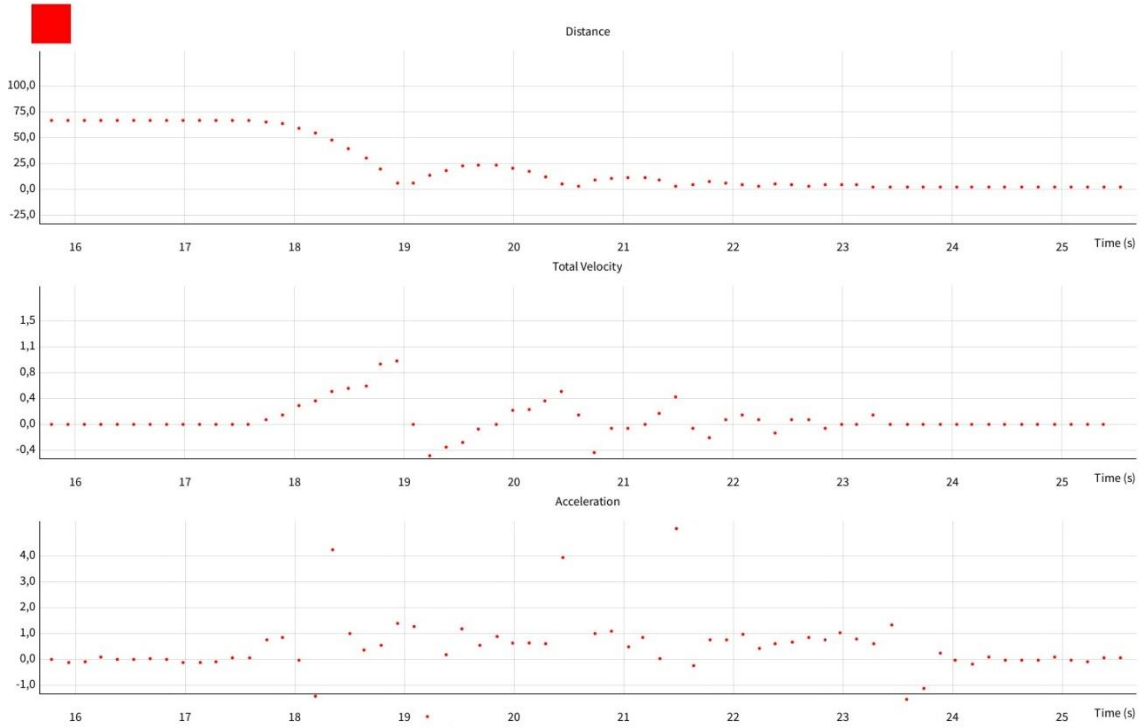
Slika 10 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.60 m/s^2, 0.70 m/s^2, te 0.53 m/s^2

Računajući s pomoću podataka prikazanih na slici (Slika *10*) dobivam vrijednost ukupne prosječne akceleracije od otprilike 0.61 m/s^2 što je manje od ukupne prosječne akceleracije iz prvog mjerenja. To nema smisla zbog toga što postav eksperimenta nije promijenjen nego je samo kôd izmijenjen. Isto tako se ne slaže s grubom procjenom koju smo dobili iz v-t grafa.

Uspoređujući s rezultatima kasnijih mjerenja vidi se da rezultati ovog mjerenja dosta odstupaju od očekivanih vrijednosti. No usprkos lošim rezultatima i dalje sam uključio ovo mjerenje u rad da prikažem rezultate mjerenja kad je alpha koeficijent stavljen na premalu vrijednost za ovaj tip pokusa. Iz ovog mjerenja zaključujem da se alpha vrijednost treba znatno povećati.

3.1.3 Treće mjerenje

U ovom mjerenju puštao sam kolica s dva utega unutra te sam postavio parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 0.95. U Processing kôdu sam maknuo dio gdje se za računanje brzine ne uzima promjena udaljenosti jednaka i manja od jednog centimetra. Pretpostavio sam da ću, povećanjem vrijednosti alpha koeficijenta, u a-t grafu imati veću brzinu odziva na promjene nego u prijašnjim mjerenjima. U ovom mjerenju je također adaptirana skala grafova kako bi se bolje prikazale promjene. Rezultat mjerenja prikazan je na slici (Slika *11*).

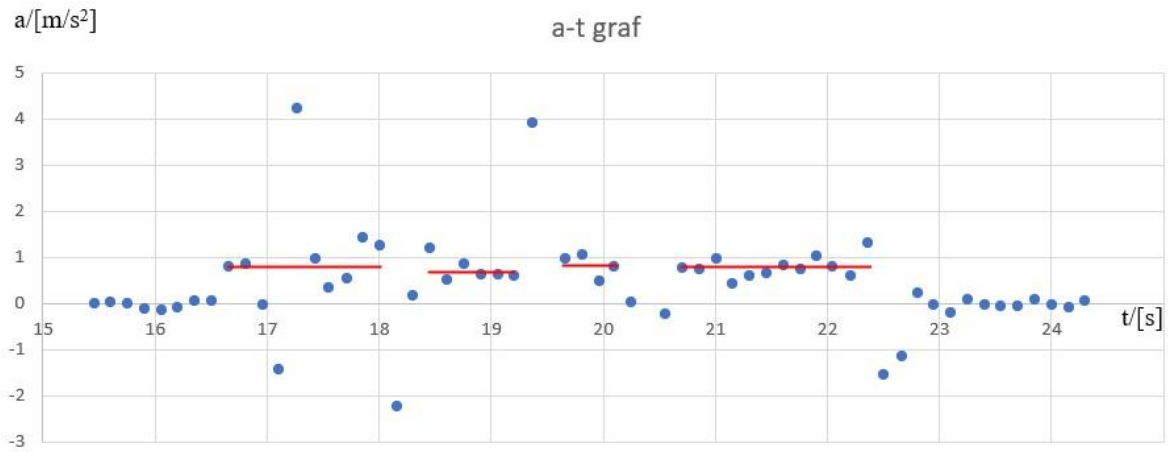


Slika 11 - Rezultati trećeg mjerenja

Pošto nisam mijenjao ništa s postavom pokusa, promjene u kôdu koje sam napravio ne utječu na mjerenje udaljenosti, vidi se da s-t graf i dalje vjerno opisuje stvarno gibanje koje je bilo u pokusu. Malo se promijenila početna udaljenost puštanja kolica od branika, no to ne utječe na ishod eksperimenta stoga nije važno držati to konstantnim.

Za v-t graf, međutim, vidi se jedna zanimljiva posljedica nesavršenosti mjernih instrumenata. Na prvom porastu brzine, tj. prilikom prvog spusta kolica niz kosinu, imam prikazan neočekivani skok u brzini. Brzina naglo skoči s 0.6 na 0.85 m/s. Ova pojava se događa upravo zato što interval vremena između mjerenja nije uvijek egzaktno isti, u mom slučaju bi trebao biti 150 milisekundi. Nažalost, tu se ništa ne može napraviti. S obzirom na to da čak i kad se definira interval od 150 ms pri crtanju, u serial monitoru može se vidjeti da taj interval nije uvijek jednak. Ako se pomnije promotri vremenski razmak točkica, može se primijetiti da je zamjetno manji interval prije nego poslije te točkice koja opisuje skok u brzini. Ako provjerimo csv datoteku možemo vidjeti da je uistinu taj interval mnogo kraći. Iz v-t grafa računam grubu vrijednost prosječne akceleracije, pa uspoređujem s dobivenim rezultatima iz a-t grafa. Ako se pogleda prvi uspon krivulje na v-t grafu vidi se da su kolica iz stanja mirovanja ubrzala do 0.95 m/s u intervalu od 1.35 sekundi. Time dobijem grubu procjenu da bi prosječna akceleracija trebala biti oko 0.7 m/s^2.

Za a-t graf sad očekivano imam ekstremne vrijednosti koje nemaju smisla, npr. prilikom prvog spusta kolica dobivam negativnu akceleraciju iako se brzina povećava. No ako isključim te ekstremne vrijednosti vidi se da je akceleracija relativno konstantna.



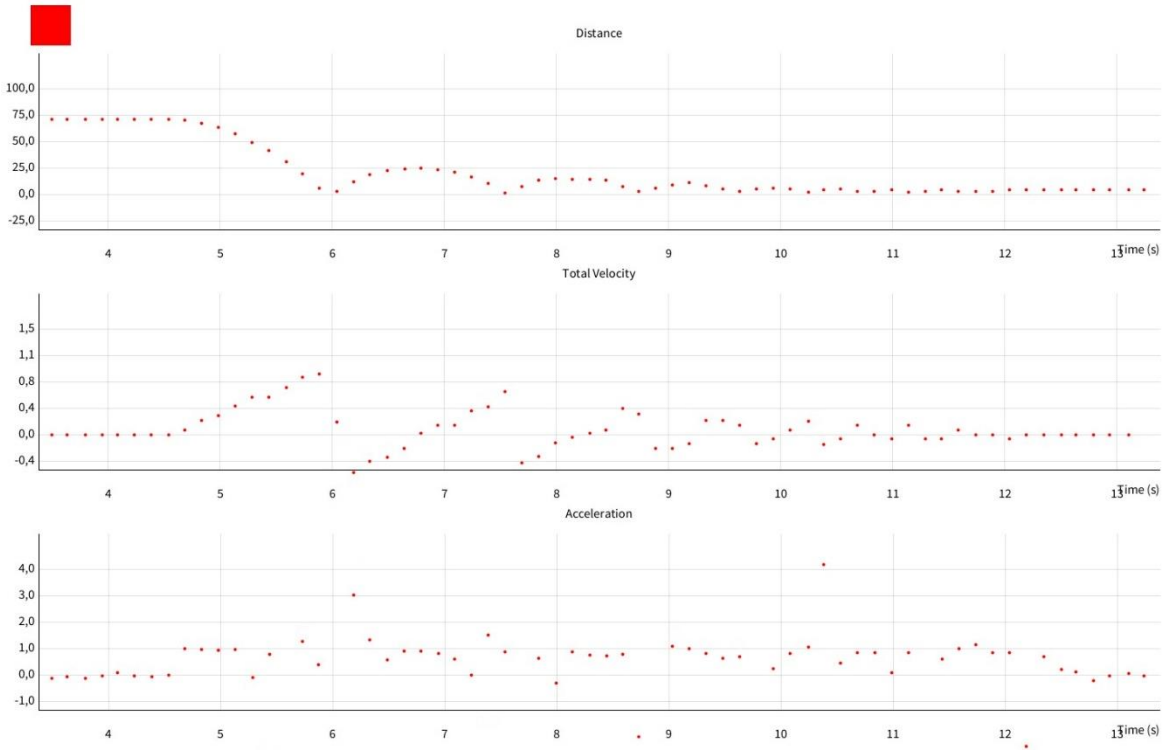
Slika 12 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.76 m/s^2, 0.73 m/s^2, 0.82 m/s^2, te 0.78 m/s^2

Računajući prosječnu akceleraciju preko prosjeka skiciranih na slici (Slika *12*) dobijem 0.77 m/s^2, što je malo veće od grube procjene koju sam dobio, no slaže se s prvim mjerenjem koje sam radio. To slaganje sam i očekivao pošto eksperimentalni postav nije mijenjan pa ne očekujem niti promjene u akceleraciji.

Najzanimljiviji dio ovog mjerenja je zadnji „rep“ akceleracije, gdje prema s-t i v-t grafovima kolica miruju, a a-t graf očitava akceleraciju. Tu se vide i negativne vrijednosti akceleracije. To je zato što je akcelerometar napravio mjerenje točno u trenutku kada su kolica udarila u prepreku i time je dobivena negativna vrijednost akceleracije.

3.1.4 Četvrto mjerenje

U ovom mjerenju puštao sam kolica s tri utega unutra te sam postavio parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 1.0 što znači da uopće ne želim izglađivanje krivulje grafa, no time dobivam maksimalnu brzinu odziva na promjene. Za Processing kôd sam zadržao da se za računanje brzine uzima svaka promjena udaljenosti jer je time znatno smanjena stepenastost brzine. Rezultat mjerenja se vidi na slici (Slika *13*).



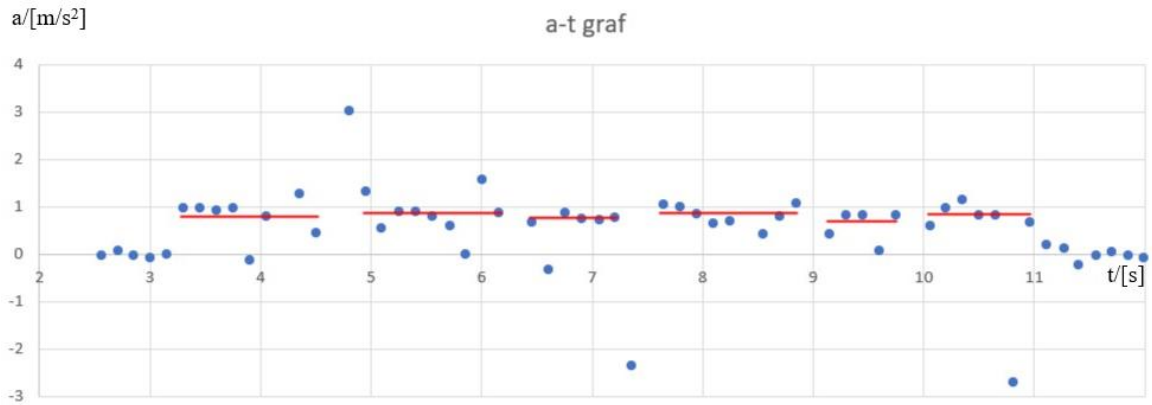
Slika 13 - Rezultati četvrtog mjerenja

S-t graf i dalje lijepo prikazuje stvarnost gibanja. Jedina razlika je povećanje inercije dodavanjem trećeg utega kolicima. To znači da se promjena smjera gibanja događala vrlo sporo te je ultrazvučni senzor dobivao ista očitanja u tom vremenskom intervalu od 2-3 mjerenja što se može uočiti po „spljoštenosti brjegova“ na s-t grafu.

Samim time se odmah vidi da je u v-t grafu brzina približno 0 m/s. Osim toga primjećujem da u nekim trenucima blago narušen oblik pravca koji smo očekivali. Iskrivljenost pravca nastaje zbog skoka u brzini koji se dogodio iz istog razloga kao i u

prošlom mjerenju. Interval je tu 138 ms, što je kraće nego prosječni interval od 150 ms. Također imam neočekivani pad brzine koji se može objasniti time što se brzina računa kao promjena udaljenosti u intervalu vremena. Senzor detektira promjenu udaljenosti sa 7 cm na 3 cm odnosno pomak što je 4 cm. No u tom intervalu su se kolica već odbila od branika i ukupni prijeđeni put je 10 cm. Smanjenjem intervala bi se bolje moglo zabilježiti podatke i izbjeći ovaj problem, no tada se pojavljuju drugi problemi koji se vide u mjerenjima obrađenim kasnije u radu. Razlog zašto u ovom mjerenju opet imam stepenastu brzinu je zato što je dodan još jedan uteg. Povećanjem mase smo povećali inerciju stoga kolicima treba više vremena da promijene smjer gibanja i dobivam očitanja udaljenosti koja se ne mijenjaju. Gruba procjena prosječne akceleracije iz podataka s v-t grafa, od 0 do 0.95 m/s u 1.35 s, daje vrijednost od 0.7 m/s^2.

Što se tiče akceleracije, imam dosad najkonzistentnija očitanja. I dalje postoje neka očitanja koja se ne slažu sa stvarnosti gibanja, no takva očitanja su manje brojna nego u prijašnjim mjerenjima.



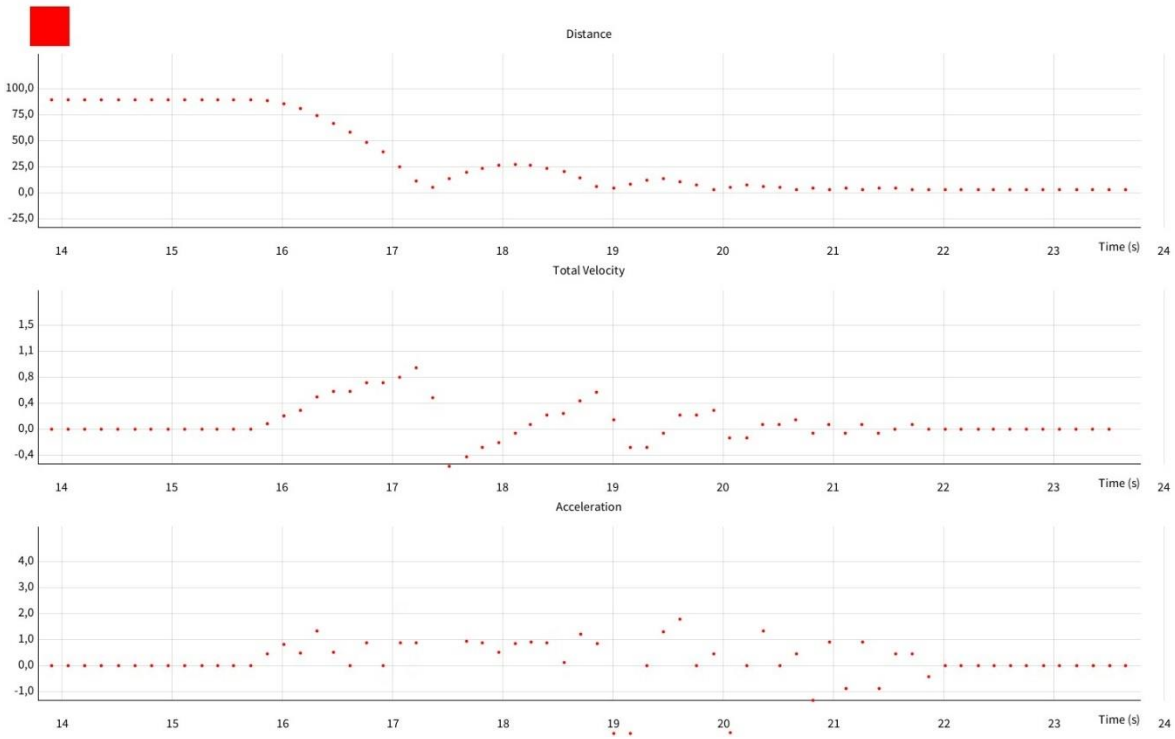
Slika 14 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.78 m/s^2, 0.83 m/s^2, 0.76 m/s^2, 0.84 m/s^2, 0.74 m/s^2, te 0.84 m/s^2

Iz prosjeka nacrtanih na slici (Slika *14*) računam prosječnu akceleraciju tijekom cijelog mjerenja i dobivam 0.79 m/s^2. To je malo veća vrijednost od one koje sam dobio u gruboj procjeni, no dobro se slaže s prosječnim akceleracijama iz prošlih mjerenja.

3.1.5 Peto mjerenje - sve preko udaljenosti

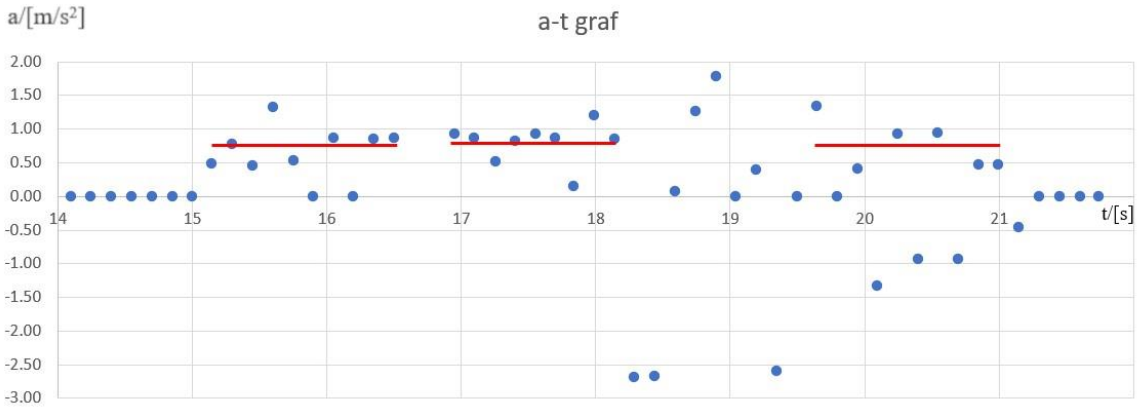
U ovom mjerenju sam uzimao podatke samo s ultrazvučnog senzora za udaljenost, te sam iz podataka o promjeni udaljenosti računao brzinu, kao i prije, no ovaj put sam računao i akceleraciju kako bih prikazao kakve grafove dobivam kada sve podatke uzimam

s jednog senzora. Promjene u Processing kôdu su navedene u dodatku C. Osim toga, maknuo sam utege iz kolica jer smo u prošlom mjerenju vidjeli da je dodatna masa loše utjecala na rezultate mjerenja brzine. Smanjenjem mase kolica dobijemo veće promjene udaljenosti te se nadam da neće biti stepenastih očitanja na v-t grafu te da ću dobiti očekivane pravce. Rezultat mjerenja vidi se na slici (Slika *15*).



Slika 15 - rezultati mjerenja s jednim senzorom

Na s-t grafu, pri prvom spustu, dobivam da je oblik te krivulje bliži pravcu nego paraboli kako sam očekivao od jednolikog ubrzanog gibanja. Ti rezultati odmah povlače da će se brzina sporo povećavati. Isto tako se može uočiti da je nagib prvog pravca na v-t grafu manji nego nagib drugog pravca. Iz toga zaključujem da su kolica brže ubrzavala nakon prvog odbijanja od branika. To mogu i potvrditi računanjem grube procjene akceleracije. Vidim iz prvog spusta kolica, da je iz stanja mirovanja postignuta brzina od 0.95 m/s u intervalu od 1.5 s. Dobijem da je gruba procjena akceleracije 0.63 m/s^2. Pri drugom spustu kolica niz kosinu brzina je išla od -0.5 m/s do 0.6 m/s u intervalu od 1.35 s. Dobijem da je tu akceleracija 0.81 m/s^2, što je znatna razlika od prvog spusta.

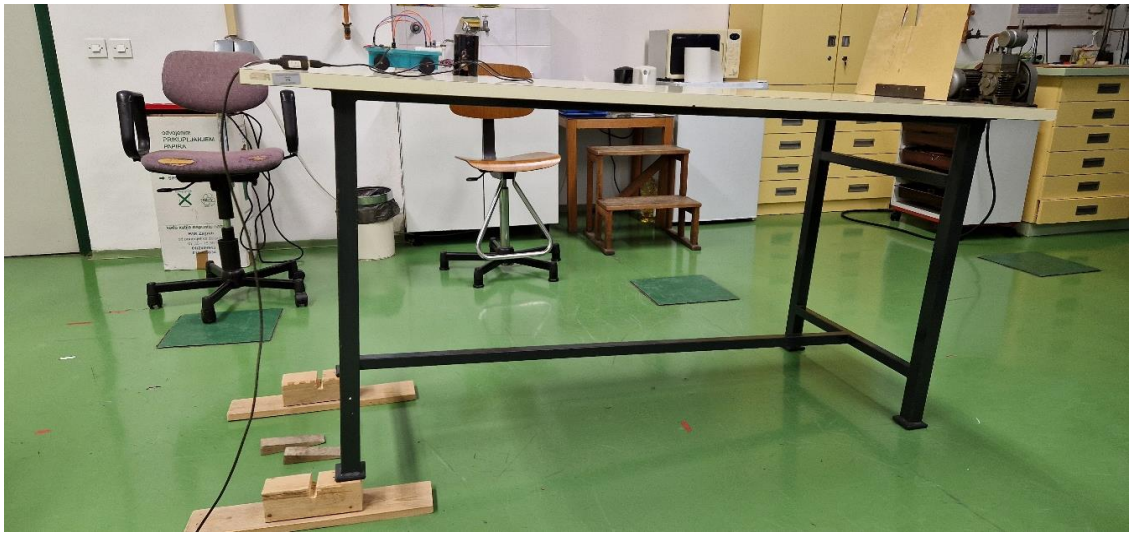


Slika 16 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.77 m/s^2, 0.79 m/s^2, te 0.76 m/s^2

Za zadnji prosjek sam također isključio negativne vrijednosti akceleracije jer znam da ako se gleda težinski po vremenu, odnosno koliko dugo te negativne akceleracije traju, da je to trajanje praktički zanemarivo u usporedbi na trajanje pozitivne akceleracije. Samo u trenutku udara imam negativnu akceleraciju, no kako sam računao akceleraciju preko promjene udaljenosti to se ne vidi na ovom grafu. To je jedna od mana mjerenja samo s jednim senzorom. Prema podacima prikazanim na slici (Slika 16), ukupna prosječna akceleracija je 0.80 m/s^2. Taj rezultat se poklapa s rezultatima iz prošlih mjerenja.

3.2 Drugi pokus

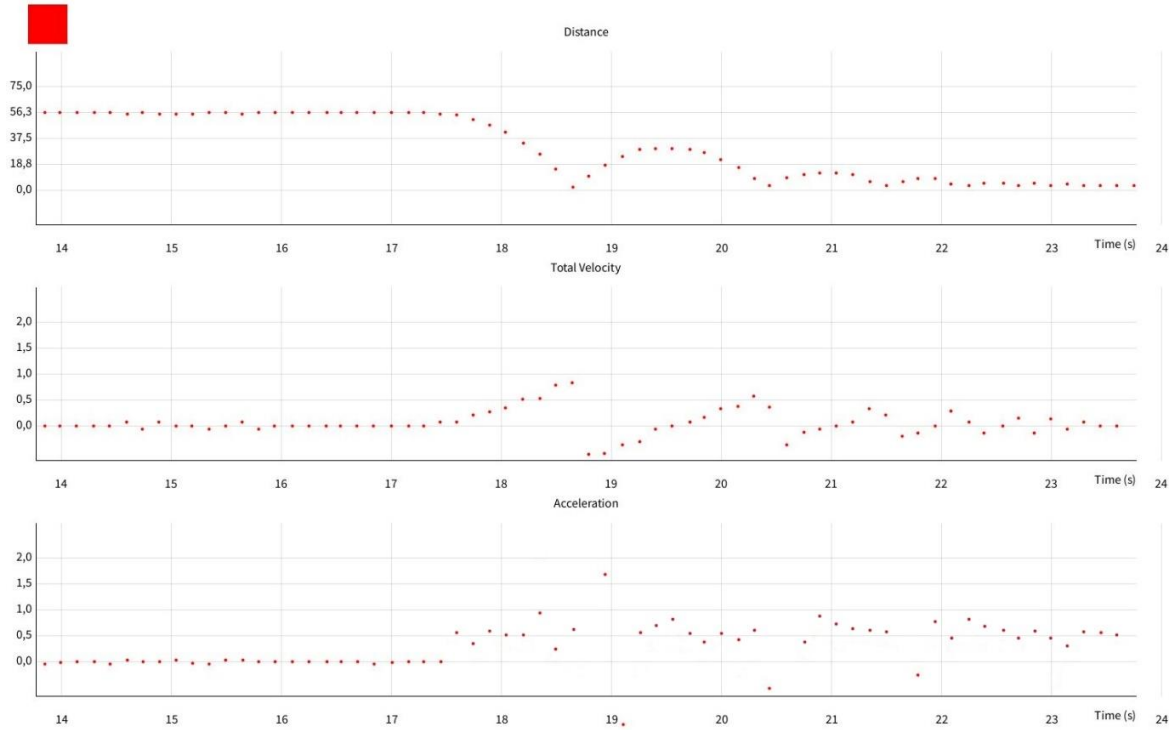
Za razliku od pokusa iz poglavlja 3.1, smanjen je nagib stola kako bi se vidjela razlika u dobivenim akceleracijama. Nagib je sada otprilike 7 stupnjeva. Postav drugog pokusa može se vidjeti na slici (Slika *17*).



Slika 17 - Postav drugog pokusa

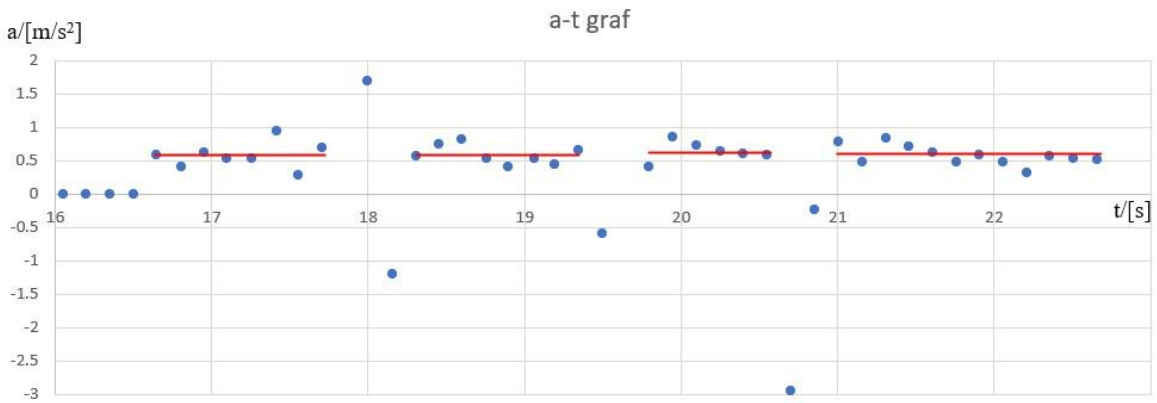
3.2.1 Prvo mjerenje

U ovom mjerenju puštao sam kolica s jednim utegom unutra te sam postavio parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 1.0 jer time dobivam maksimalnu brzinu odziva na promjene. Za Processing sam zadržao da se za računanje brzine detektira bilo kakva promjena udaljenosti jer je time znatno smanjena stepenastost brzine. Rezultat mjerenja se vidi na slici (Slika *18*).



Slika 18 - Rezultati prvog mjerenja drugog pokusa

Na s-t grafu sam dobio očekivanu krivulju, te na v-t grafu imam vrlo uredne pravce. Iz v-t grafa mogu dobiti grubu procjenu akceleracije. Pri prvom spustu kolica su ubrzala od 0 m/s do 0.8 m/s u intervalu od 1.35 s. Time dobivam grubu procjenu akceleracije koja iznosi 0.59 m/s^2. U a-t grafu imam prilično konstantne vrijednosti, no u nekoliko mjerenja dobivam neočekivano ekstremnu vrijednost. Te ekstremne vrijednosti se neće uzimati u obzir pri računanju srednje akceleracije.

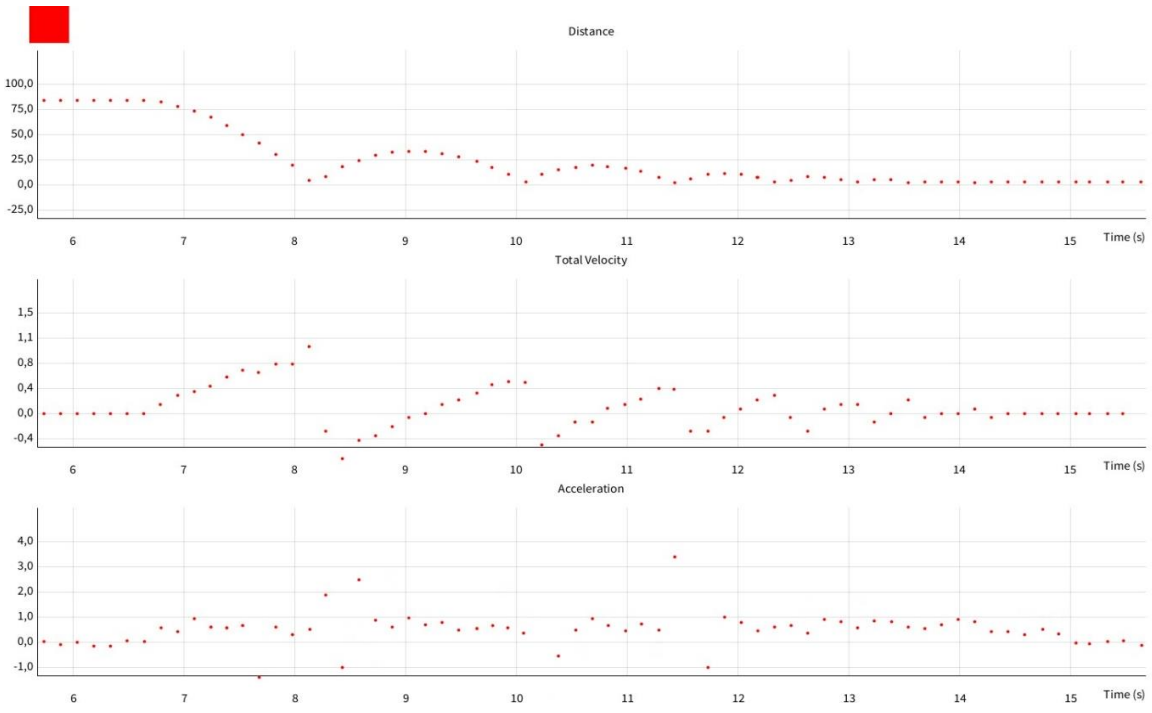


Slika 19 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.57 m/s^2, 0.59 m/s^2, 0.63 m/s^2, te 0.58 m/s^2

Iz rezultata prikazanih na slici (Slika 19) dobijemo da je ukupna srednja akceleracija 0.59 m/s^2. Taj rezultat se poklapa s procjenom akceleracije koju smo računali iz v-t grafa.

3.2.2 Drugo mjerenje

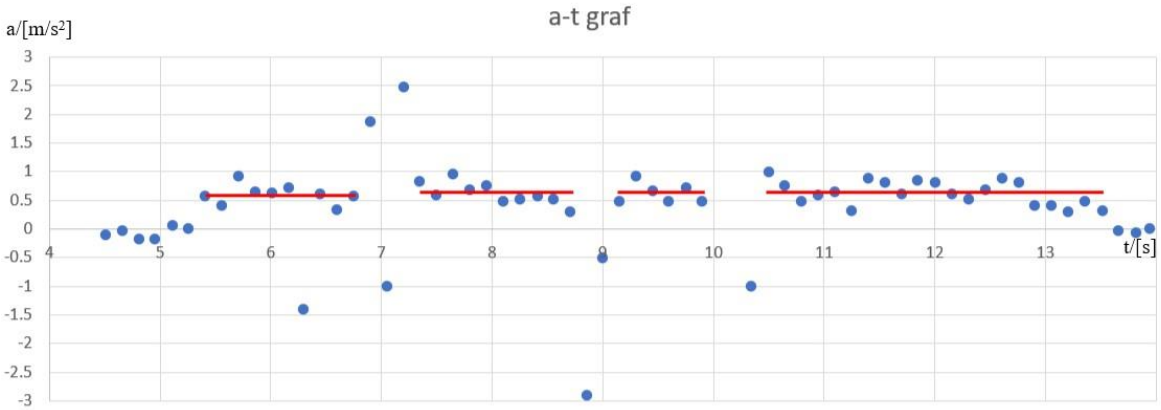
U ovom mjerenju puštao sam kolica s jednim utegom unutra te sam zadržao u Arduino kôdu vrijednost alpha koeficijenta od 1.0. Za Processing sam zadržao da se za računanje brzine detektira bilo kakva promjena udaljenosti. Jedina razlika u odnosu na prošlo mjerenje jest da su kolica bila puštena s veće udaljenosti te da su skale na osima grafovima prilagođene da se bolje vide rezultati mjerenja. Rezultat mjerenja se vidi na slici (Slika 20).



Slika 20 - Rezultati drugog mjerenja drugog pokusa

Baš kao i u prijašnjem pokusu s-t graf odlično opisuje realnost gibanja. Imao sam sreće da su se očitanja dogodila točno u trenutku kada je udaljenost bila najmanja, odnosno u trenutku udarca kolica u branike. Zbog toga sam i dobio ovako uredan v-t graf gdje se lijepo vide pravci koji se periodički ponavljaju. Na s-t grafu još se može vidjeti da sam ispustio kolica s veće početne udaljenosti nego u prvom mjerenju, pa dobivam i veću maksimalnu brzinu. Također se primijetiti da je na prvom pravcu na v-t grafu promjena brzine bila od 0 m/s do 0.8 m/s u otprilike 1.2 sekunde što daje grubu procjenu akceleracije od 0.67 m/s^2.

Na a-t grafu sam dobio iznimno stabilne, konstantne vrijednosti. Izuzev 3-4 točkice na grafu sva ostala mjerenja prikazuju konstantnu vrijednost s minimalnim fluktuacijama.



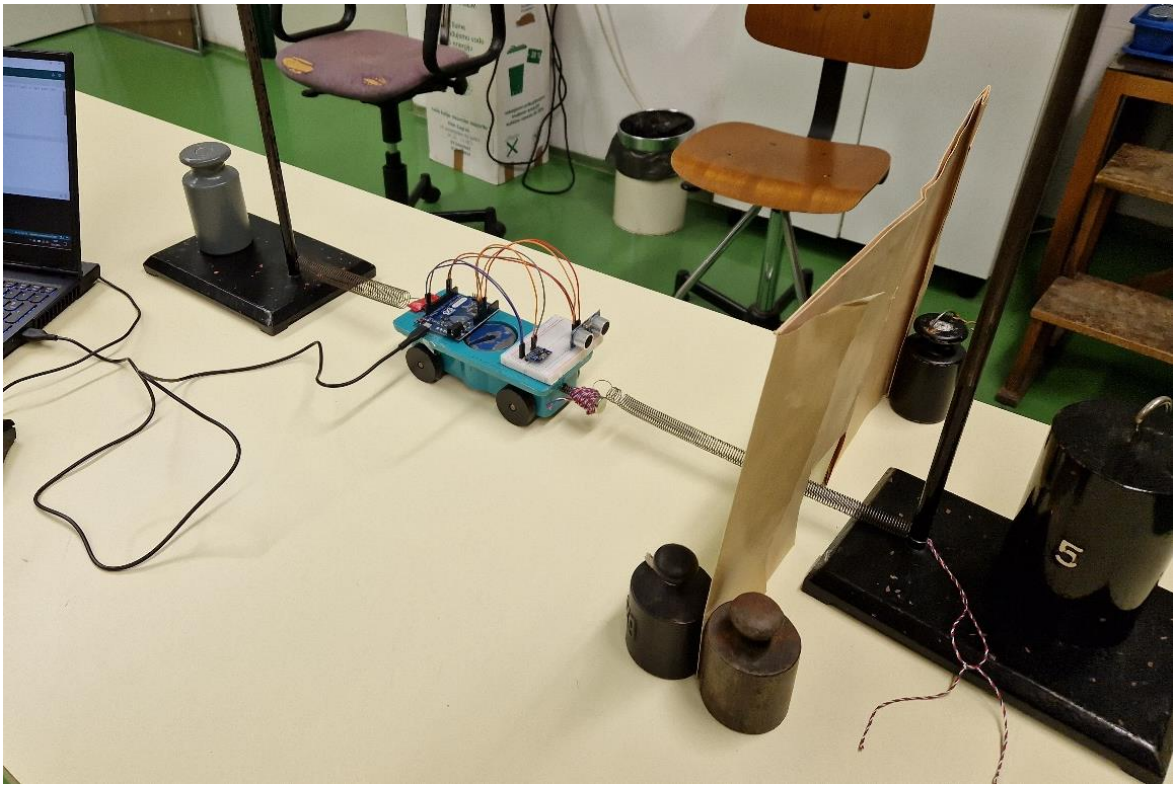
Slika 21 - Graf akceleracije nacrtan u Excelu pomoću podataka u csv datoteci, crvene linije označavaju prosječnu vrijednost akceleracije u tom vremenskom intervalu, prikazane vrijednosti su 0.59 m/s^2, 0.62 m/s^2, 0.62 m/s^2, te 0.62 m/s^2

S pomoću rezultata prikazanih na slici (Slika *21*) računam ukupnu prosječnu akceleraciju i dobivam vrijednost od 0.61 m/s^2. Taj rezultat je nešto manji od grube procjene no dobro se slaže s prošlim mjerenjem. Sa smanjenjem kuta nagiba kosine iz prvog pokusa se očekivano smanjila i akceleracija kolica tijekom gibanja u drugom pokusu.

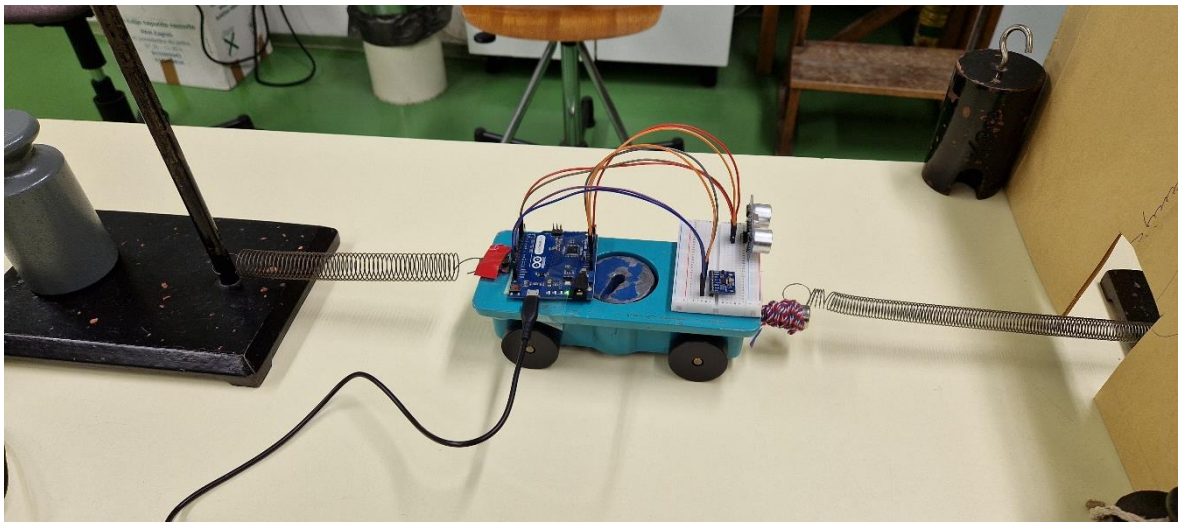
3.3 Treći pokus

Pribor koji je korišten za izvođenje pokusa su pokusna kolica, stol ili neka ravna podloga, karton ili neki drugi zastor, utezi za kolica, dva stalka, te dvostrana ljepljiva traka. Osim toga imam sav pribor za mjerenje: mikrokontroler, senzore, računalo, te žice potrebne za spajanje.

Opis Pokusa: Na kolica su montirani mikrokontroler i senzori baš kao i u prijašnjim pokusima. Kolica su povezana oprugama s dvije strane. Te opruge su pričvršćene na čvrste krajeve, u ovom slučaju na stalke. Ti stalci se mogu proizvoljno razmaknuti kako bi se odredilo koliko će opruge biti rastegnute na početku. Na jedan kraj postavljen je karton koji služi kao pregrada za ultrazvučni senzor. Kolica su zatim maknuta iz ravnotežnog položaja i puštena da slobodno titraju. Postav se može vidjeti na slikama (Slika *22* i Slika *23*).



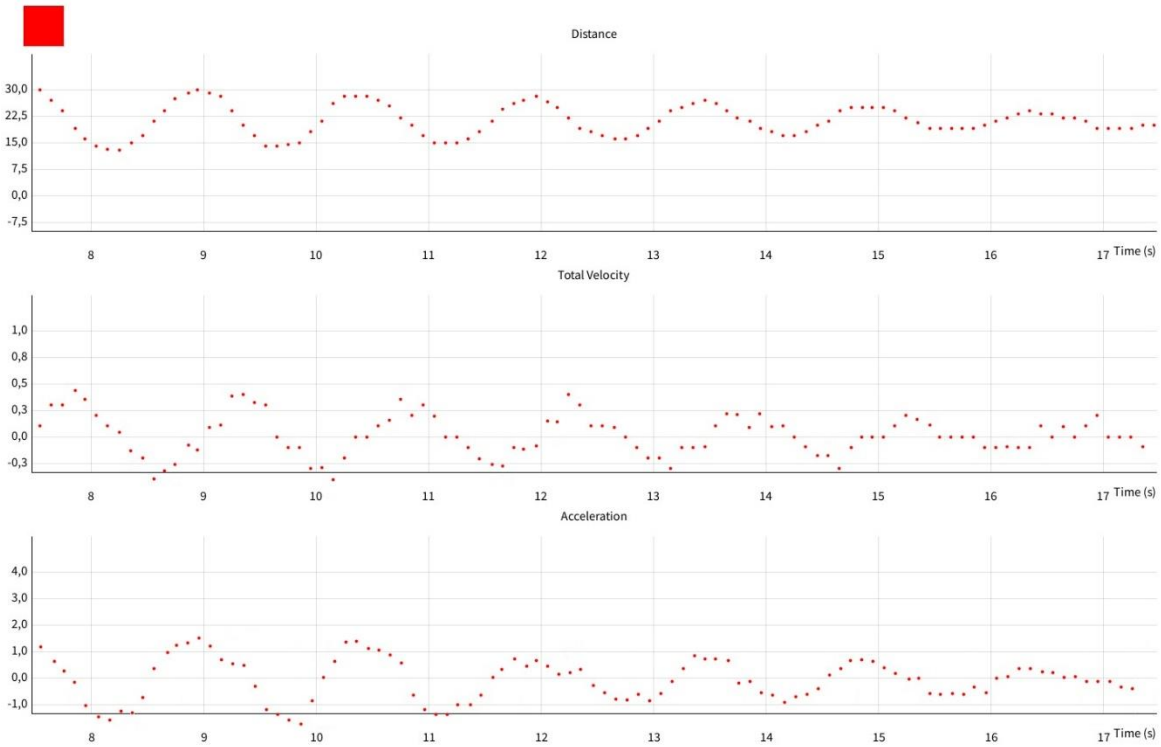
Slika 22 - Postav pokusa s oprugama



Slika 23 - Postav pokusa s oprugama 2

3.3.1 Prvo mjerenje

U ovom mjerenju imam kolica s tri utega unutra dok sam u Arduino kôdu postavio parametre pokusa tako da je vrijednost alpha koeficijenta 0.4. Za Processing kôd sam stavio prag od jednog cm potreban za očitanje promjene udaljenosti pri računanju brzine kao što je bilo u prvom i drugom mjerenju prvog pokusa. Smanjenjem alpha koeficijenta smanjujem brzinu odziva, no kako su očekivane krivulje u ovom pokusu sinusoide, krivulje koje nemaju naglih promjena, očekujem da bi dodatno izglađivanje dobro došlo. Osim toga je promijenjen interval mjerenja na 100 ms. Rezultat mjerenja vidi se na slici (Slika *24*):



Slika 24 - Rezultati prvog mjerenja trećeg pokusa

Na s-t grafu imam sinusoidu, gotovo bez nepravilnosti, te vidim i gušenje titranja uzrokovano disipacijom energije iz sustava zbog trenja. Za ravnotežni položaj odabrana je udaljenost od 20 cm. Vidimo da zbog manjeg intervala između mjerenja, s-t graf ima nekoliko mjerenja gdje izgleda kao da se vrijednost nije promijenila. Sporo gibanje kolica pridonosi tom rezultatu.

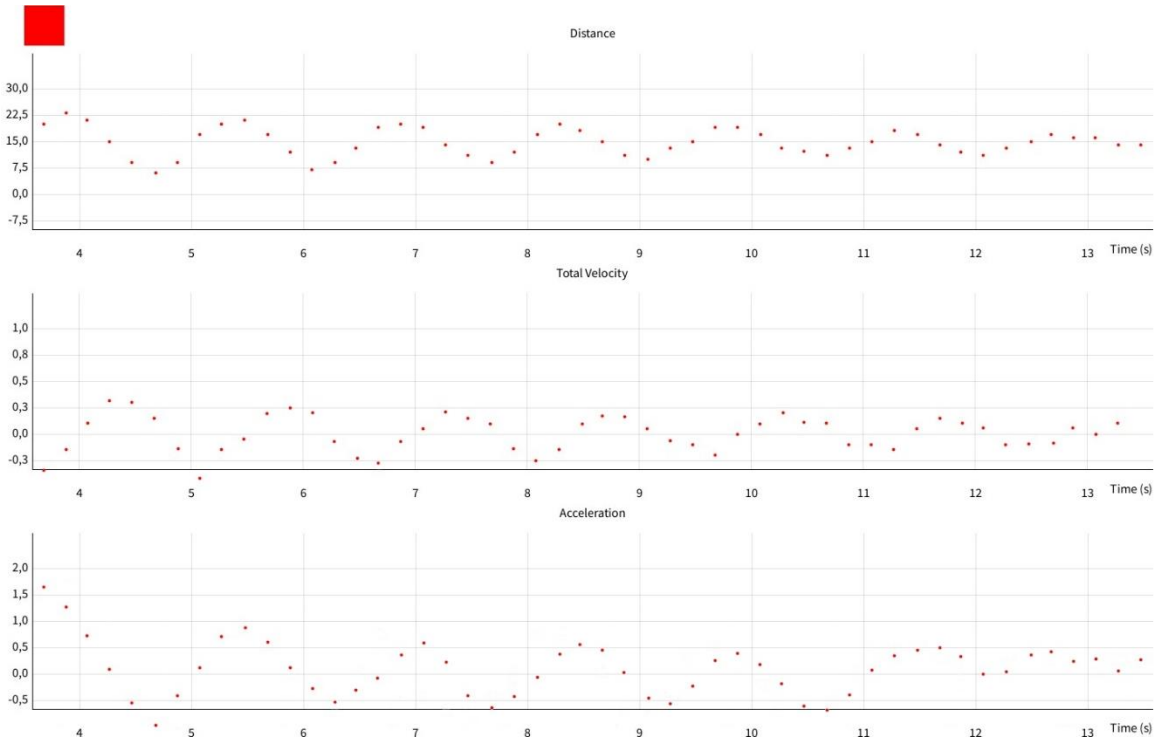
U v-t grafu može se vidjeti grubi oblik sinusoide. Stepenastost rezultata uzrokovana je načinom na koji se brzina računa preko promjene udaljenosti te dobivam dosta točkica koje su na nuli. Glavni uzrok stepenastosti krivulje na v-t grafu je smanjenje intervala između mjerenja. Zbog tog je razloga izbjegavano smanjivanje intervala u prijašnjim pokusima.

Osim toga je u kôdu zadano da se za računanje brzine zanemare promjene udaljenosti manje od 1 cm. To je najveći uzrok ovog stepenastog oblika krivulje u v-t grafu.

U a-t grafu imam lijepu sinusoidu. „Divljanje“ grafa akceleracije je dosta ublaženo odabirom malog alpha koeficijenta koji jako izglađuje dobivenu krivulju. Još jedna stvar koja bi povećala eratično ponašanje akceleracije je smanjenje intervala među mjerenjima no izgleda da su postavke zadane u kôdu odgovarale ovakvom tipu pokusa. Zadnju stvar koju bih spomenuo je da u mjerenjima koja su rađena s malim alpha koeficijentom zna doći do pomaka (eng. „shift“) u rezultatima akceleracije naspram udaljenosti i brzine. To se vrlo lako može popraviti u Processing kôdu u funkciji za crtanje grafa. Ta tehnika je bila primijenjena tijekom crtanja grafova na slikama (Slika *24* i Slika *25*).

3.3.2 Drugo mjerenje

U ovom mjerenju imam kolica s tri utega unutra te sam postavio parametre pokusa u Arduino kôdu tako da je vrijednost alpha koeficijenta 0.4. Za Processing kôd sam maknuo prag od jednog cm za promjenu udaljenosti koja se koristi za računanje brzine kako bih smanjio stepenastost krivulje v-t grafa. Koeficijent alpha je isti kao u prošlom mjerenju jer vidimo da je lijepo izglađivao dobivenu krivulju. Također je promijenjen interval mjerenja na 200 milisekundi kako bi se ublažila stepenastost krivulje u v-t grafu. Rezultat mjerenja vidi se na slici (Slika *25*).



Slika 25 - Rezultati drugog mjerenja trećeg pokusa

Na s-t grafu imam lijepu sinusoidu, te vidimo i gušenje titranja uzrokovano disipacijom energije iz sustava. Za ravnotežni položaj je odabrana udaljenost je 15 cm. Smanjena je brzina kojom se kolica gibaju zbog manje napetosti opruga, no zbog promjene intervala mjerenja vidi se da to nije narušilo urednost mjerenja, te da nemam stepenastost ni u s-t i u v-t grafu. U v-t grafu se mnogo bolje vidi sinusoida nego u prošlom mjerenju. Povećanjem intervala sam uklonio najveći uzrok nepravilnosti u krivulji iz prošlog mjerenja. Pred sam kraj mjerenja se gubi sinusoida zbog prespore brzine kolica. U a-t grafu imamo lijepu sinusoidu. Zbog velikog EMA a-t graf je bio pomaknut u usporedbi s ostala dva. U Processing kôdu sam to ispravio i dobio pravilne podatke u grafovima.

3.4 Ograničenja i potencijalna poboljšanja

U ovom poglavlju razmatram ograničenja opreme, potrebna poboljšanja te eventualne alternative koje bi mogle doprinijeti unapređenju eksperimentalnog postava korištenog u radu te unaprjeđenju i optimizaciji kôda.

Eksperimentalni postav, kao što su stol i pokusna kolica, pokazao je nekoliko ograničenja. Šum, koji je akcelerometar očitavao pri mjerenjima, dolazio je od vibracija koje su uzrokovane nesavršenom podlogom te kolicima čiji kotači su zapinjali i proklizavali po podlozi. Zamjena postava u tračnice te kolica prilagođena za te tračnice, gdje je cilj što više smanjiti vibracije i trenje, značajno bi uklonila šum. Šum je moguće djelomično ukloniti dobro razrađenim kôdom, korištenjem naprednijih tehnika filtriranja i metoda obrade podataka. Pri izradi rada sam razmatrao low-pass filtre za filtriranje vibracija te high-pass filtre za uklanjanje komponente gravitacije te naprednije filtre poput Kalmanovog filtra za bolje rukovanje šumom i poboljšanje preciznosti. Međutim, zbog složenosti implementacije i vremenskog ograničenja, korištene su samo jednostavne metode poput eksponencijalnog pomičnog prosjeka (EMA). Low- i high-pass filtri nisu bili korišteni jer tijekom testiranja nisu davali rezultate koji su se slagali s realnosti gibanja. Ako bi se u budućim verzijama kôda koristili ti filtri, treba dodatno razraditi kôd kako bi se dobili ispravni rezultati.

Nedostatak bežične veze i nezgrapnost postava sa žicama otežava manipulaciju i može utjecati na točnost rezultata. To zahtijeva dodatnu pažnju, te je za izvođenje pokusa potrebno više osoba. Uvođenje bežične veze bi olakšalo korištenje opreme i uklonilo moguće negativne utjecaje koji su se pojavljivali u pokusima u ovom radu, a bili su uzrokovani žicama. Dodavanje Wi-Fi modula bi također omogućilo daljinsko upravljanje i nadzor.

Integracija svih komponenti u kompaktnu kutijicu ili kućište olakšala bi manipulaciju i korištenje postava, povećavajući praktičnost i preciznost. Isto tako je potrebno osigurati što bolje pričvršćivanje breadboarda sa senzorima i mikrokontrolera na sama kolica kako ne bi dolazilo do naknadnih očitanja zbog pomicanja senzora u odnosu na ta kolica. U radu je korištena obostrana ljepljiva traka kako bi se učvrstio breadboard, no u nekim mjerenjima se ta traka pokazala nedostatnom stoga preporučujem traženje boljih i trajnijih alternativa.

4 Motivacija za rad

4.1 Uporaba u nastavi

Izvođenje pokusa na nastavi fizike u osnovnoj i srednjoj školi može postati monotono te učenici brzo mogu izgubiti interes. U osmim razredima osnovne škole te u prvom razredu srednje škole rade se pokusi vezani s gibanjem. Dosad su se pokusi uvijek izvodili s tipkalom i trakicama koje bi učenici vezali za kolica i ovisno o gradivu postavili aparaturu te provodili istraživanja s nekim manjim varijacijama između pokusa. No svi ti pokusi se svode na istu stvar, proučavanje trakica te brojanje točkica koje tipkalo ostavi na trakici. Naravno, tipkalo i trakica su sasvim dostatni, korisni, i brz način pokazivanja i istraživanja različitih vrsta gibanja, no taj jedan te isti način nastavnici i učenici koriste u nastavi više od godinu dana u jednom komadu. Korištenjem mikrokontrolera u nastavu uvodimo dinamiku kojom bismo povećali interes učenika te im dali drugi pogled u način na koji mogu provoditi istraživanja. Pogotovo u današnje vrijeme gdje se djeca u sve ranijoj dobi upoznaju s tehnologijom. Naravno ovi pokusi se mogu provoditi zajedno sa starim metodama pri čemu se mogu istraživati razlike u pristupima izvođenja pokusa.

4.2 Prednosti

Arduino mikrokontroleri, sa senzorima i priborom koji je potreban da se sve spoji na računalo, jeftini su te ih je jednostavno nabaviti preko webshopa. Škole bi se vrlo lako trebale moći opskrbiti svim potrebnim priborom. Komponente su iznenađujuće izdržljive, no i dalje treba biti oprezan s njima, baš kao i s pokusnim kolicima.

Mnogo lakše je uvesti neke pojmove s ovim načinom izvođenja pokusa npr. pojam trenutne brzine. Postupnim smanjivanjem intervala između mjerenja, što se ovom metodom može lako postići, dolazi se do tog pojma. Inače se u udžbenicima fizike kao u [10], pojam trenutne brzine uvodi riječima ili u najboljem slučaju na primjeru jednog grafa. Uporabom platforme za grafičko crtanje kao Processing mogu se nacrtati brojni grafovi istog mjerenja s različitim intervalima crtanja mjerenja. Time učenici dobiju bolju vizualnu reprezentaciju pojma trenutne brzine nego dosadašnjim načinom poučavanja. Osim toga, ako se učenicima kod izvođenja pokusa zada i spajanje senzora na mikrokontroler, što je iznimno jednostavno, možemo ih unaprijed upoznati s nekim pojmovima u elektronici.

4.3 Primjena u cijelom kurikulumu

U ovom radu obrađeni su samo pokusi vezani uz gibanje tijela, no postoji širok raspon senzora koji omogućuju izvođenje pokusa iz gotovo sveg gradiva osnovne i srednje

škole te bi se moglo uvesti i na fakultete. Neki od senzora koji postoje i mogu se koristiti su senzori za vlagu, silu, svjetlost, magnetska polja, brzinu, zvuk, temperaturu, i još mnogi drugi. [11] Jedan od senzora korišten u ovom radu, ADXL345 akcelerometar, se može koristiti i kao žiroskop ako je dovoljno dobro programiran. Kombinacijom svih ovih senzora te dobrim vještinama programiranja mogu se napraviti eksperimentalna mjerenja za bilo koji dio kurikuluma iz fizike za osnovne i srednje škole.

Službena Arduino web stranica ima sekciju, education (obrazovanje), gdje se može odabrati razina koja se želi učiti i vježbati za bilo kakav rad s Arduino mikrokontrolerima. [12]

4.4 Međupredmetna povezanost s informatikom

Programiranje mikrokontrolera može biti zahtjevno no ono je i vrlo poučno. Može se koristiti kao razvoj vještina s programskim jezikom, ili jezicima, koji se koriste za programiranje mikrokontrolera. Sav kôd korišten u ovom radu je najosnovniji dio korištenih programskih jezika te postoji mnogo prostora za unaprjeđenje i optimizaciju.

Nastavnici fizike i nastavnici informatike bi mogli zajednički zadati zadatke gdje bi učenici trebali isprogramirati svoje dijelove koje bi koristili za mjerenja u fizici. Naravno to zahtijeva da su i sami nastavnici upoznati s aparaturom koju koriste. Tu također treba ispravno procijeniti i učeničke sposobnosti. Neki od jednostavnijih projekata koji bi se izvodili mogu biti upoznavanje s dijelovima, kako ti dijelovi rade, te kao što je već spomenuto, učenje nekih pojmova u elektronici.

5 Zaključak

U ovom diplomskom radu ispitivao sam upotrebu ADXL345 akcelerometra i HC-SR04 ultrazvučnog senzora u mjerenju akceleracije i udaljenosti s pomoću Arduino Leonardo mikrokontrolera. Izradom kôda programirao sam mikrokontroler i senzore u Arduino IDE platformi. Korišteni kôd je specijaliziran za izvođenje pokusa fokusiranih za proučavanje pravocrtnog, ubrzanog gibanja. Kroz provedene pokuse sam potvrdio funkcionalnost ADXL345 senzora da mjeri akceleraciju kolica koja se gibaju, s različitim stupnjevima uspješnosti. Isto tako sam potvrdio sposobnost HC-SR04 senzora da precizno mjeri udaljenost objekata, odnosno udaljenost kolica od zvučne barijere. Kombinacijom dobivenih podataka s ta dva senzora grafički sam prikazao gibanje tijekom pokusa preko Processing platforme.

Međutim, naišao sam na nekoliko izazova i ograničenja tijekom izrade kôda te izvođenja pokusa. Najveći izazov je bio naučiti dva programska jezika s kojima prije ovog rada nisam bio upoznat. Posljedično, korišteni kôdovi u pokusima su bazični te ograničeni u spektru korištenja. Kôdovi imaju mnogo prostora za poboljšanje i proširenje svojih mogućnosti. Potencijalna poboljšanja uključuju implementaciju naprednijih tehnika filtriranja kako bi se smanjio šum u podacima i poboljšala preciznost mjerenja. Ograničenja eksperimentalnog postava uključuju nesavršenost opreme, poput vibracija koje su utjecale na točnost mjerenja te izazivale šum. Upotreba Wi-Fi konekcije mogla bi olakšati izvođenje pokusa te pružiti mogućnost daljinskog nadzora i upravljanja.

U budućim istraživanjima i radovima preporučujem dodatno ispitivanje različitih tipova senzora i tehnika obrade podataka kako bi se postigla veća pouzdanost i preciznost rezultata. Korišteni senzor za mjerenje udaljenosti je vrlo dobro služio svrsi. No ima nekoliko ograničenja, kao što su maksimalna preciznost te udaljenost. Za izvođenje pokusa s većim udaljenostima i eventualno većom preciznosti, treba naći pogodnu alternativu. Korišteni akcelerometar je jako osjetljiv senzor te je izazivao probleme uključujući jaki šum u podacima pri izvođenju mjerenja. Nalaženje modela koji bolje odgovara potrebama ovog pokusa bi unaprijedio točnost dobivenih rezultata. Smatram da bi korišteni model akcelerometra bio pogodniji za izvođenje pokusa, kao što su pokusi 1 i 2 iz ovog rada, kada bi se koristili napredniji algoritmi za smanjenje šuma.

Unatoč tim izazovima, smatram da uvođenjem upotrebe mikrokontrolera te pripadajuće opreme znatno se može poboljšati nastava, ne samo fizike, nego i ostalih znanosti.

Dodaci

Dodatak A – Arduino kôd

```
// biblioteke koje su potrebne za pravilno funkcioniranje kôda
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <NewPing.h>

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);
NewPing sonar(4, 5, 300);  // Trigger pin, Echo pin, maksimalna udaljenost (u centimetrima), brojevi 4 i 5 označavaju digitalne pinove 4 i 5 na mikrokontroleru

// deklariranje varijabli
const int numSamples = 500;  // broj neobrađenih podataka za kalibraciju
const int numReadings = 20;   // broj očitanja za pomični prosjek
const float alpha = 0.6;  // EMA faktor izglađivanja


float meanX, meanY, meanZ;
float readingsX[numReadings];
float readingsY[numReadings];
float readingsZ[numReadings];

float emaX = 0.0;
float emaY = 0.0;
float emaZ = 0.0;

unsigned long previousMillis = 0;
const long interval = 200;  // Interval između printanja podataka, u milisekundama


// funkcija koja služi za početno postavljanje kôda
void setup() {
  Serial.begin(115200);
  // čekanje uspostave serijske veze prije nastavka
  while (!Serial);
  Serial.println("kod se izvršava");

  if (!accel.begin()) {
```

```
    Serial.println("ADXL345 nije detektiran, provjeri konekciju!");
    while (1);
  }

  // prilagođavanje načina rada akcelerometra, frekvencija mjerenja neobrađenih podataka te osjetljivost
i raspon mjerenja
  accel.writeRegister(ADXL345_REG_BW_RATE, ADXL345_DATARATE_800_HZ);
  accel.writeRegister(ADXL345_REG_DATA_FORMAT, ADXL345_RANGE_2_G);

  Serial.println("Kalibracija...");
  calibrateAccelerometer();

  // printanje početnih vrijednosti kalibracije
  Serial.println("Kalibracija završena.");
  Serial.print("Srednja akceleracija (X, Y, Z): ");
  Serial.print(meanX);
  Serial.print(" m/s^2, ");
  Serial.print(meanY);
  Serial.print(" m/s^2, ");
  Serial.print(meanZ);
  Serial.println(" m/s^2");
  Serial.println("AkceleracijaX, AkceleracijaY, AkceleracijaZ, UkupnaAkceleracija, Udaljenost");
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis <= interval) {
    // ažuriranje nizova za računanje pomičnog prosjeka
    for (int i = numReadings - 1; i > 0; --i) {
      readingsX[i] = readingsX[i - 1];
      readingsY[i] = readingsY[i - 1];
      readingsZ[i] = readingsZ[i - 1];
    }
    // prikupljanje očitanja akcelerometra
    sensors_event_t event;
    accel.getEvent(&event);
    readingsX[0] = event.acceleration.x - meanX;
    readingsY[0] = event.acceleration.y - meanY;
```

```
    readingsZ[0] = event.acceleration.z - meanZ;
}
else{
previousMillis = currentMillis;
    // Izračun eksponencijalnog pomičnog prosjeka (EMA) ubrzanja
    emaX = alpha * readingsX[0] + (1 - alpha) * emaX;
    emaY = alpha * readingsY[0] + (1 - alpha) * emaY;
    emaZ = alpha * readingsZ[0] + (1 - alpha) * emaZ;
    float totalAcceleration = sqrt(pow(emaX, 2) + pow(emaY, 2) + pow(emaZ, 2));

    // prikupljanje podataka s ultrazvučnog senzora
    unsigned int distance = sonar.ping_cm();
    // ispis obrađenih podataka
    Serial.print(emaX);
    Serial.print(",");
    Serial.print(emaY);
    Serial.print(",");
    Serial.print(emaZ);
    Serial.print(",");
    Serial.print(totalAcceleration);
    Serial.print(",");
    Serial.println(distance);
  }
}
// funkcija koja izvršava početnu kalibraciju akcelerometra, postavljena da početne vrijednosti za sve osi
budu 0.0 m/s^2
void calibrateAccelerometer() {
  float sumX = 0.0, sumY = 0.0, sumZ = 0.0;

  for (int i = 0; i < numSamples; ++i) {
    sensors_event_t event;
    accel.getEvent(&event);
    sumX += event.acceleration.x;
    sumY += event.acceleration.y;
    sumZ += event.acceleration.z;
  }

  meanX = sumX / numSamples;
```

```
  meanY = sumY / numSamples;
  meanZ = sumZ / numSamples;
}
```

Dodatak B – Processing kôd

```
import processing.serial.*;
import java.util.ArrayList;

// Inicijalizacija serijske komunikacije s Arduinom
Serial serialPort;
// Liste za pohranu podataka
ArrayList<Float> selectedAccelerationData = new ArrayList<Float>();
ArrayList<Float> distanceData = new ArrayList<Float>();
ArrayList<Integer> timeData = new ArrayList<Integer>();
ArrayList<Float> totalVelocityData;

int startTime;
boolean drawGraph = true;
float totalDistance = 0;
// Pozicija i dimenzije gumba za zaustavljanje
int stopRectX = 40;
int stopRectY = 10;
int rectSize = 50;

boolean isDrawing = true;
// PrintWriter za zapisivanje podataka u CSV file
PrintWriter output;
// Odabrana os ubrzanja (0 za X os, 1 za Y os, 2 za Z os)
int accelerationAxis = 1;
// Postavljanje početnih parametara
void setup() {
  size(1500, 950);
  smooth();
  // Postavljanje serijske komunikacije
  serialPort = new Serial(this, "COM5", 115200);
  serialPort.bufferUntil(&apos;\n&apos;);
  // Postavljanje putanje za spremanje CSV file-a
  String filePath = "file_path";
  output = createWriter(filePath);
  // Naslovi stupaca u CSV file-u
```

```
  output.println("Vrijeme : Udaljenost : Ubrzanje : Brzina");
}

// Funkcija za crtanje prozora
void draw() {
  background(255);
  // Čitanje podataka iz serijske komunikacije
  while (serialPort.available() > 0) {
    String data = serialPort.readStringUntil(&apos;\n&apos;);
    if (data != null) {
      processData(data);
    }
  }
  // Provjera i crtanje grafova
  if (drawGraph && isDrawing) {
    stroke(255, 0, 0);
    // Crtanje grafova, argumenti su (x pozicija, y pozicija, naslovi osi, koji podaci se crtaju u taj graf, skala y osi, koliko se x os treba pomaknuti da grafovi budu vremenski usklađeni)
    drawGraph(50, 670, "Ubrzanje", "Vrijeme (s)", selectedAccelerationData, 2.0, 0);
    drawGraph(50, 70, "Udaljenost", "Vrijeme (s)", distanceData, 30.0, 1);
    totalVelocityData = calculateTotalVelocity(distanceData, timeData);
    drawGraph(50, 370, "Ukupna Brzina", "Vrijeme (s)", totalVelocityData, 1.0, 1);
    // Crtanje gumba za zasutavljanje
    fill(255, 0, 0);
    rect(stopRectX, stopRectY, rectSize, rectSize);
  }
}
// Funkcija za procesiranje pristiglih podataka
void processData(String data) {
  String[] values = splitTokens(data, ",");
  if (values.length == 5) {
    float accelerationX = float(values[0]);
    float accelerationY = float(values[1]);
    float accelerationZ = float(values[2]);
    float selectedAcceleration = 0.0;
    // Odabir odgovarajućih podataka za akceleraciju na temelju odabrane osi
    if (accelerationAxis == 0) {
      selectedAcceleration = accelerationX;
    } else if (accelerationAxis == 1) {
      selectedAcceleration = accelerationY;
```

```
    } else if (accelerationAxis == 2) {
      selectedAcceleration = accelerationZ;
    }

    selectedAccelerationData.add(selectedAcceleration);

    float distance = float(values[4]);
    int currentTime = millis();
    float elapsedTimeInSeconds = (currentTime - startTime) / 1000.0;
    distanceData.add(distance);
    timeData.add(currentTime);
    if (startTime == 0) {
      startTime = currentTime;
    }

    float velocity = getVelocityFromGraph();
    totalDistance = distance;

    // Zapisivanje podataka u CSV file
    output.print(String.format("%.3f", elapsedTimeInSeconds).replace(&apos;,&apos;, &apos;.&apos;) + ":" +
                 String.format("%.0f", distance).replace(&apos;,&apos;, &apos;.&apos;) + ":" +
                 String.format("%.2f", selectedAcceleration).replace(&apos;,&apos;, &apos;.&apos;) + ":" +
                 String.format("%.2f", velocity).replace(&apos;,&apos;, &apos;.&apos;));
    output.println();
    output.flush();
  }
}

// Funkcija za dohvaćanje posljednje vrijednosti brzine iz grafa za zapis u csv file
float getVelocityFromGraph() {
  if (totalVelocityData.size() > 0) {
    return totalVelocityData.get(totalVelocityData.size() - 1);
  } else {
    return 0.0;
  }
}

// Funkcija za izračun ukupne brzine preko udaljenosti, računa se trenutna brzina u m/s
ArrayList<Float> calculateTotalVelocity(ArrayList<Float> distanceData, ArrayList<Integer> timeData) {
  ArrayList<Float> velocityData = new ArrayList<Float>();
```

```
  for (int i = 1; i < distanceData.size(); i++) {
    float timeDifference = (timeData.get(i) - timeData.get(i - 1)) / 1000.0;
    float distanceChange = distanceData.get(i - 1) - distanceData.get(i);

    // Zanemari male promjene udaljenosti u cm
    if (abs(distanceChange) > 1.0) {
      float velocity = (distanceChange / timeDifference) / 100.0;
      velocityData.add(velocity);
    } else {
      velocityData.add(0.0);
    }
  }
  return velocityData;
}

// Funkcija za crtanje grafa
void drawGraph(int x, int y, String label, String timeLabel, ArrayList<Float> data, float maxY, int shift) {
  int plotWidth = width - 2 * x;
  int plotHeight = 220;

  if (data.size() == 0 || timeData.size() == 0) {
    return;
  }
  fill(0);
  textSize(16);
  textAlign(CENTER);
  text(label, x + plotWidth / 2, y - 20);
  textAlign(RIGHT);
  // Naslov x osi
  text(timeLabel, x + plotWidth, y + plotHeight + 30);
  // Crtanje osi grafa
  stroke(0);
  line(x, y + plotHeight, x + plotWidth, y + plotHeight);
  line(x, y, x, y + plotHeight);

  // Crtanje mreže linija (eng. "gridlines")
  for (int i = -2; i <= 3; i++) {
    float yPos = map(i, -2, 2, y + plotHeight * 0.2, y + plotHeight * 0.8);
    stroke(0, 30);
```

```
  line(x, yPos, x + plotWidth, yPos);
}

// Crtanje grafova
noFill();
beginShape();
int currentTime = millis();
stroke(255, 0, 0);
for (int i = 0; i < data.size(); i++) {
  if (i < timeData.size()) {
    float timeDifference = currentTime - timeData.get(i);

    if (i < timeData.size() && timeDifference <= 10000) {
      float xPos = map(timeData.get(i), currentTime - 10000, currentTime, x, x + plotWidth);
      if (i > shift) {
        float yPos = map(data.get(i - shift), 0, maxY, y + plotHeight * 0.8, y + plotHeight * 0.2);
        strokeWeight(4);
        point(xPos, yPos);
        strokeWeight(1);
      }
    }
  }
}
endShape();

// Crtanje oznaka na y osi
for (int i = -1; i <= 4; i++) {
  float yPos = map(i, 4, 0, y + plotHeight * 0.2, y + plotHeight * 0.8);
  textAlign(RIGHT);
  float scaledValue = i * maxY / 4;
  text(String.format("%.1f", scaledValue), x - 5, yPos + 4);
}

// Crtanje oznaka na x osi
int numSeconds = 100;
for (int i = 0; i <= numSeconds; i++) {
  float xPos = map(i * 1000, currentTime - 10000, currentTime, x, x + plotWidth);
  textAlign(CENTER);
  text(Integer.toString(i), xPos, y + plotHeight + 35);
  stroke(0, 30);
```

```
    line(xPos, y, xPos, y + plotHeight);
  }
}

// Funkcija koja pokreće i zasutavlja crtanje kada je stisnut gumb za zaustavljanje
void mousePressed() {
  int mouseXPosition = mouseX;
  int mouseYPosition = mouseY;

  // Provjera pritiska na gumb za zaustavljanje
  if (mouseXPosition > stopRectX && mouseXPosition < stopRectX + rectSize && mouseYPosition >
stopRectY && mouseYPosition < stopRectY + rectSize) {
    isDrawing = !isDrawing;
    if (!isDrawing) {
      noLoop();
    } else {
      loop();
    }
  }
}
```

Dodatak C – Izmjena Processing kôda za korištenje podataka jednog senzora

```
// dodati array list za akceleraciju koja se računa
ArrayList<Float> accelerationData = new ArrayList<Float>();

// ovaj blok kôda dodati u processData funkciju, na kraju odmah prije output.print funkcije

    float velocityChange = 0.0;
    if (totalVelocityData.size() >= 2) {
      velocityChange = totalVelocityData.get(totalVelocityData.size() - 1) -
totalVelocityData.get(totalVelocityData.size() - 2);
    }

    if (abs(velocityChange) < 0.05) {
      selectedAccelerationData.add(0.0);
    } else {
      accelerationData.add(calculateAcceleration(totalVelocityData));
      selectedAccelerationData.add(accelerationData.get(accelerationData.size() - 1));
    }

    if (accelerationData.size() > 0) {
```

```
                // tu sad ide output.print cijeli dio
                }
// nakon processData funckije dodati sljedeći blok kôda, NE u processData funkciju

float calculateAcceleration(ArrayList<Float> velocityData) {
  float acceleration = 0.0;
  if (velocityData.size() >= 2) {
    float velocityChange = velocityData.get(velocityData.size() - 1) - velocityData.get(velocityData.size() - 2);
    float timeDifference = (timeData.get(timeData.size() - 1) - timeData.get(timeData.size() - 2)) / 1000.0;

    // Izbjegni djeljenje s nulom
    if (timeDifference > 0.0) {
      acceleration = velocityChange / timeDifference;
    }
  }
  return acceleration;
}
```

Literatura

[1] Izvor slike Arduino Leonardo, (datum pristupa web stranici 7.2.2024.), https://store-usa.arduino.cc/products/arduino-leonardo-with-headers

[2] Arduino Leonardo tehnička dokumentacija (datum pristupa web stranici 7.2.2024.), https://docs.arduino.cc/hardware/leonardo/

[3] Izvor slike akcelerometra ADXL345, (datum pristupa web stranici 7.2.2024.), https://learn.adafruit.com/assets/6359

[4] Tehnička dokumentacija ADXL345 akcelerometra, (datum pristupa web stranici 7.2.2024.), https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[5] Izvor slike HC-SR04 ultrazvučnog senzora, (datum pristupa web stranici 7.2.2024.), https://www.sparkfun.com/products/15569

[6] Tehnička dokumentacija HC-SR04 ultrazvučnog senzora, (datum pristupa web stranici 7.2.2024.), https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[7] Arduino forum za pomoć i podršku, (datum pristupa web stranici 7.2.2024.), https://forum.arduino.cc/

[8] Arduino lista referenci, (datum pristupa web stranici 7.2.2024.), https://www.arduino.cc/reference/en/

[9] Processing lista referenci, (datum pristupa web stranici 7.2.2024.), https://processing.org/reference/

[10] Vladimir Paar, udžbenik za 1. razred gimnazije, Školska knjiga 2003.g., (datum pristupa dokumentu 12.2.2024.), https://www.scribd.com/document/390261300/Fizika-1-Vladimir-Paar?doc_id=390261300

[11] Popis senzora kompatibilnih za korištenje s Arduino mikrokontrolerima, (datum pristupa web stranici 7.2.2024.), https://store.arduino.cc/collections/sensors

[12] Arduino edukacija, (datum pristupa web stranici 7.2.2024.), https://www.arduino.cc/education/