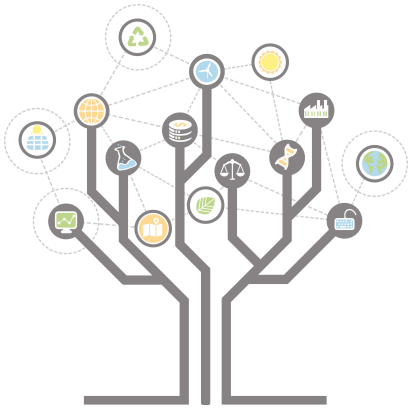
Arduino platforma za mjerenje vibracija

Višnić, Saša



Arduino platforma za mjerenje vibracija

Višnić, Saša

SVEUČILIŠTE U ZAGREBU

GEOTEHNIČKI FAKULTET

SAŠA VIŠNIĆ

ARDUINO PLATFORMA ZA MJERENJE VIBRACIJA

ZAVRŠNI RAD

VARAŽDIN, 2018.

SVEUČILIŠTE U ZAGREBU

GEOTEHNIČKI FAKULTET

ZAVRŠNI RAD

ARDUINO PLATFORMA ZA MJERENJE VIBRACIJA

KANDIDAT:

SAŠA VIŠNIĆ

MENTOR:

doc.dr.sc. MARIO GAZDEK

NEPOSREDNI VODITELJ:

dr.sc. MARKO PETRIC

VARAŽDIN, 2018.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je završni rad pod naslovom

ARDUINO PLATFORMA ZA MJERENJE VIBRACIJA

rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi te je izrađen pod mentorstvom doc.dr.sc. Marija Gazdeka i dr.sc. Marka Petrica.

Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

U Varaždinu, _____.

ZAHVALE

Zahvale na uloženom trudu, sugestijama i vodstvu, upućujem mentoru doc.dr.sc. Mariju Gazdeku te naročito neposrednom voditelju dr.sc. Marku Petricu. Posebne zahvale na pomoći u polju programiranja upućujem asistentu Filipu Dodigoviću, bez kojeg konačni rezultat rada ne bi bio moguć. Zahvaljujem i asistentu Davoru Stanku na pomoći pri izvođenju terenskih pokusa i pri analizi rezultata.

Zahvale upućujem i Mihaelu Buhinu i Mihaelu Gregurašu (Elektrostrojarska škola Varaždin) na vrhunskoj izradi kućišta za akcelometar koje je bilo potrebno pri ispitavanjima unutar i van građevine.

Neizmjerne zahvale na podršci i strpljenju upućujem svojoj obitelji i djevojci, kojima posvećujem ovaj rad.

U Varaždinu, lipanj 2018.

Saša Višnić

SAŽETAK RADA

Tema rada jest istraživanje i ispitivanje mogućnosti primjene open-source Arduino platforme u polju geoinženjerstva okoliša. Na primjerima mjerenja vlažnosti zraka i temperature te mjerenja vibracija u usporedbi s profesionalnim uređajem pokazuje se uspješnost i razina primjene platforme u te svrhe. Rad opisuje osnovne karakteristike Arduina i kroz primjere uvodi u njegovo funkcioniranje. Ukratko se opisuju teorijske osnove vibracija i mehaničkih valova, a zatim se prikazuju rezultati mjerenja vibracija u zatvorenom i otvorenom prostoru. S obzirom na razlike u cijeni između sustava temeljenog na Arduinu i profesionalnog uređaja, rezultati mjerenja su vrlo ohrabrujući i pretpostavljaju mogućnost daljnjeg napretka platforme.

KLJUČNE RIJEČI

Arduino, vibracije, open-source platforme, temperatura, vlažnost zraka, geoinženjerstvo okoliša

Sadržaj

1. Uvod 1
2. Arduino 2
2.1. Nastanak i razvoj Arduino platforme 2
2.2. Opis Arduino platforme 3
2.3. Osnove korištenja Arduino platforme 5
2.4. Mjerenje vlažnosti zraka i temperature pomoću Arduino platforme 11
3. Titranje, valovi, vibracije u okolišu 16
3.1. Titranje 16
3.2. Valovi 18
3.3. Vibracije u okolišu 21
4. Mjerenje vibracija pomoću Arduino platforme 22
4.1. Oprema korištena pri ispitivanju 22
4.2. Mjerenja u zatvorenom prostoru 24
4.3. Mjerenja u otvorenom prostoru 26
5. Zaključak 28
Popis literature 29
Popis slika 31
Prilozi 32

1. Uvod

Open-source platforme doživljavaju značajan rast broja korisnika i mogućnosti koje pružaju. Njihove prednosti su dostupnost, jednostavnost, veliki broj online radionica i uputa te niska cijena. Mogućnosti kao da su ograničene samo maštom. Jedna od tih platformi je i Arduino, čija primjena u geoinženjerstvu okoliša ima veliki potencijal. Od nastanka Arduina do danas, na njemu je napravljeno više tisuća projekata iz raznih polja djelatnosti, od elektronike i elektrotehnike do poljoprivrede. Unutar polja inženjerstva okoliša, Arduino pruža mogućnosti praćenja stanja atmosfere (temperatura, vlažnost zraka, tlak zraka, koncentracije plinova i sl.), mjerenja zvučnog i svjetlosnog onečišćenja, mjerenja elektromagnetske interferencije, mjerenja vibracija, analize električne vodljivosti vode te brojna druga mjerenja i analize. Široki spektar mogućnosti, veliki potencijal platforme i pristupačnost korisnicima, razlozi su nastajanja ovog rada. U prvom dijelu rada opisana je Arduino platforma, njene komponente i princip rada kroz primjer mjerenja temperature i vlažnosti zraka. Drugi dio rada posvećen je teorijskom opisu i karakteristikama titranja, svojstvima mehaničkih valova i vibracija u okolišu te primjeni Arduina za njihovo mjerenje. Cilj rada je pokazati kvalitetu mjerenja dobivenih preko Arduino platforme te na taj način dokazati njen potencijal u inženjerstvu okoliša.

2. Arduino

U ovom je poglavlju opisana Arduino platforma, njen nastanak, dijelovi i princip rada. Također se kroz primjer mjerenja temperature i vlažnosti zraka u periodu od 5 sati, prikazuje jedan od načina prikupljanja i prezentiranja dobivenih podataka.

2.1. Nastanak i razvoj Arduino platforme

Ideja o nastanku Arduino platforme začeta je 2005. godine u gradiću Ivrei u predgrađu Torina, kao rezultat želje za jeftinijom elektroničkom platformom u usporedbi s ostalima iz tog vremena, a koja bi služila kao uređaj za kontroliranje interaktivnih studentskih projekata. Suosnivači Arduina su Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino i David Mellis. Od svog začetka Arduino koristi programski jezik baziran na Wiring programskom jeziku, koji je sličan C++ programskom jeziku, ali je djelomično pojednostavljen i izmijenjen. Cijela platforma temeljena je na projektu Hernanda Barragana kojem je bio cilj ispreplesti Arduino kao hardversku jezgru, Wiring kao programski jezik i Processing kao softversku osnovu, na način da su međusobno zavisni te tvore jedinstvenu platformu za razvoj projekata. [1,2,3]

Danas postoji preko 200 distributera Arduino proizvoda, među kojima je najveće poduzeće Adafruit Industries iz New Yorka. Prema njihovim procjenama, do 2011. godine proizvedeno je i plasirano oko 300 000 Arduino proizvoda, a potražnja za njima raste te je prema riječima Davida Cuartiellesa, suosnivača Arduina, 2013. godine bilo registrirano oko 700 000 korisnika („...Arduino has registered over 700,000 official boards, but we have estimated that there is at least one derivative or clone board per every official one. Our server statistics seem to indicate so...“). [4,5]

Izvorni Arduino kontroler nazvan je Arduino Uno (Slika 1.), a do danas je razvijen niz različitih dizajna pločica koji uključuju Arduino Leonardo, Arduino 101, Arduino Mega, Arduino Zero, Arduino Due i ostale. Također su nastale brojne pločice s mikrokontrolerima koje su kompatibilne s Arduino softverom i jezikom, a jedna od njih je i Croduino Basic2 (Slika 1.), pomoću koje su izvedena sva mjerenja u ovom radu.

Slika 1. Arduino Uno i Croduino Basic2 [1,5]

2.2. Opis Arduino platforme

Arduino je naziv za open-source elektroničku platformu koja putem jednostavnog hardvera (senzori i sustav za upravljanje) i pojednostavljenog softverskog jezika omogućuje primanje vanjskog podražaja (input-a) te njegovog pretvaranja u digitalni signal (output). Osnovne sastavnice platforme jesu elektronički mikrokontroler čije komponente ovisi o vrsti modela, Arduino programski jezik temeljen na Wiring programskom okviru za upravljanje mikrokontrolerima te Arduino softver temljen na Processing softveru. Wiring i Processing su, kao i Arduino, open source, što znači da su dostupni besplatno, zajedno sa svim projektima i doprinosima koje su korisnici napravili na njima. [1,2]

S obzirom da su sva mjerenja u ovom radu napravljena s Croduino Basic2 pločicom, slijedi njen opis i dijelovi. Croduino Basic2 bazirana je na Atmelovom mikrokontroleru Atmega328 i Silabsovom USB konventeru CP2102. Sastoji se od 22 Input-Output pina, od čega je 14 digitalnih (D0-D13), a 8 analognih (A0-A7), Atmelovog Atmega328 mikrokontrolera (32kB flash, 2kB RAM), pinova za napajanje, LED-ica (Slika 2. označene plavom bojom), mini-b USB priključka (Slika 2. označen ljubičasto s lijeve strane) pomoću kojeg se povezuje s računalom, senzora i aktuatora te reset tipke (Slika 2. označena ljubičasto s desne strane). [7,8]

Pinovi (Slika 2. označeni crvenom bojom) služe za povezivanje komponenata, senzora i aktuatora te ukopčavanje žica. Razlikuju se digitalni, analogni i ostali pinovi. Digitalni pinovi (Slika 2. označeni s digital i brojevima 0-13) očitavaju podražaje na način da podražaja ima ili nema, odnosno 1 ili 0. Tako na primjer digitalni pin može služiti za spajanje LED dioda pomoću kojih se lampica može paliti i gasiti.

Analogni pinovi (Slika 2. obilježeni s „analog“ i brojevima 0-7) mogu osim samog postojanja podražaja, očitati i intenzitet ili kontrolirati napon koji prolazi kroz pin i u trošilo, dakle nešto su sofisticiraniji. Pomoću analognog pina može se očitavati promjena temperature, intenziteta svjetlosti i slično. Ostali pinovi su:

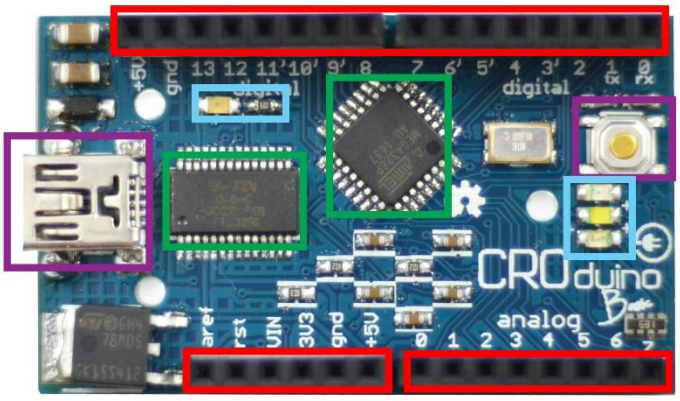
- +5V – izvor istosmjernog napajanja napona 5V, daje malu struju (400mA)
- gnd – (ground) negativni pol istosmjernog napona
- 3V3 – izvor istosmjernog napajanja napon 3V, daje izrazito malu struju
- VIN – ulaz istosmjernog napona 7V do 24V
- rst – povezan s reset tipkom
- aref – referentni analogni napon

Atmel Atmega328 mikrokontroler (Slika 2. označen zelenom bojom s desne strane) upravlja cijelom pločicom. Na njega se učitava kod s računala, kontrolira što se događa s pinovima, očitava signale koji dolaze i slično.

USB konverter (Slika 2. označen zelenom bojom s lijeve strane) služi za komunikaciju pločice s računalom, preko njega se uploada kod.

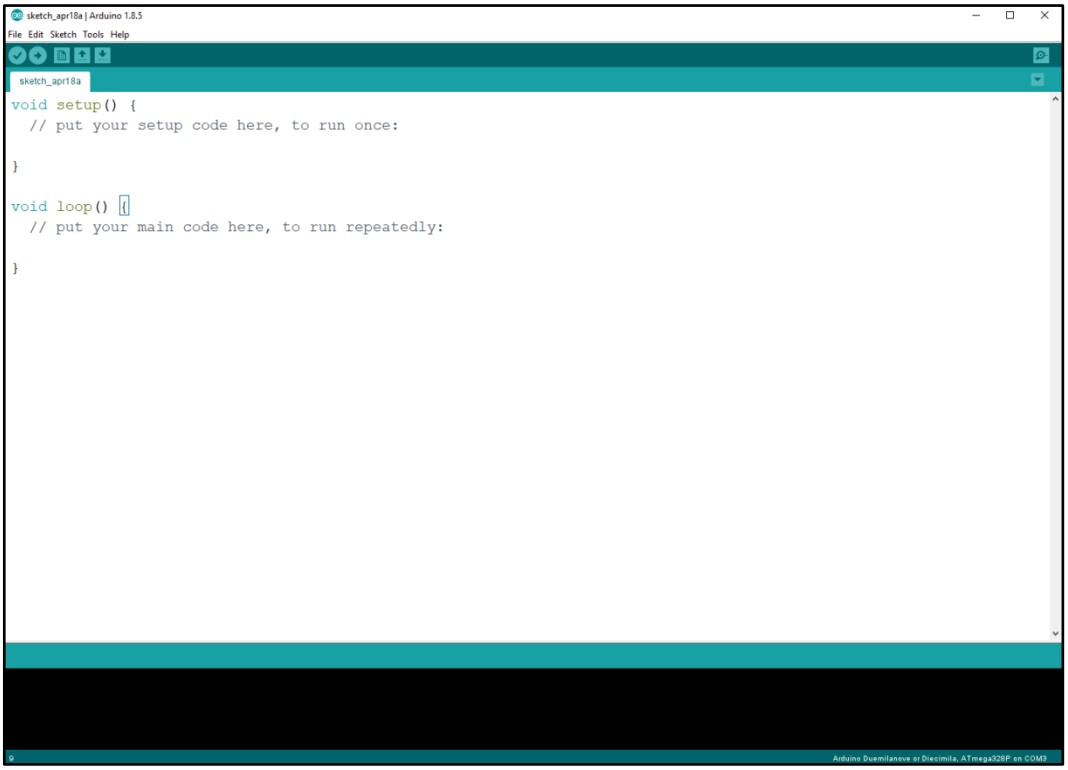
Senzori se spajaju na pločicu uglavnom preko eksperimentalne pločice povezivanjem na pinove sa žicama. Vrste i raspon mogućnosti senzora su vrlo široki, od mjerenja temperature i tlaka do očitavanja položaja i vibracija.

Aktuatori su dijelovi uređaja koji su odgovorni za obavljanje neke radnje, npr. pretvaranja električne energije u svjetlosnu. [7,8]



Slika 2. Komponente Croduino Basic pločice [7]

Croduino Basic2 kompatibilan je s Arduino softverom za učitavanje koda na pločicu. Arduino softver dostupan je besplatno na službenoj web stranici Arduina.



Slika 3. Arduino softver s osnovnim funkcijama *void setup* i *void loop*, temeljnim naredbama svakog programa

2.3. Osnove korištenja Arduino platforme

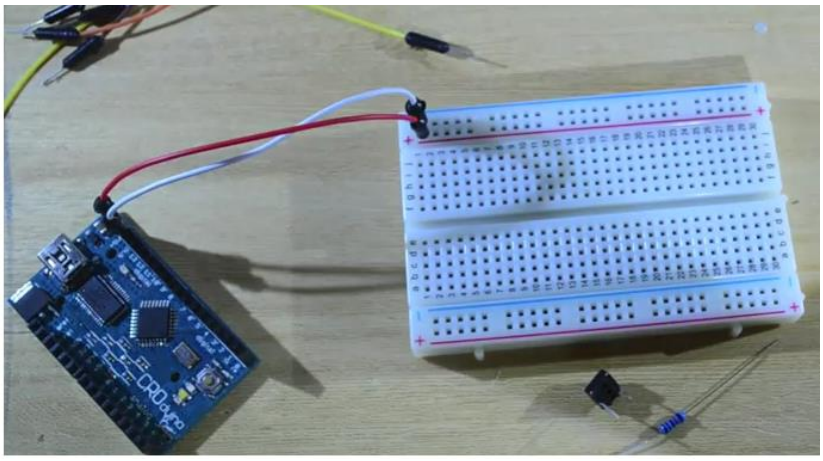
Kao osnovni primjer korištenja Arduino platforme opisan je spoj Croduino Basic2 kontrolne pločice s LED lampicom preko eksperimentalne pločice.

Za ovaj primjer korišteni su:

- Croduino Basic2
- mini eksperimentalna pločica
- žice za povezivanje
- LED lampica
- 2 otpornika (330Ω i 10000Ω)
- prekidač
- mini-b USB kabel
- Arduino softver i osobno računalo.

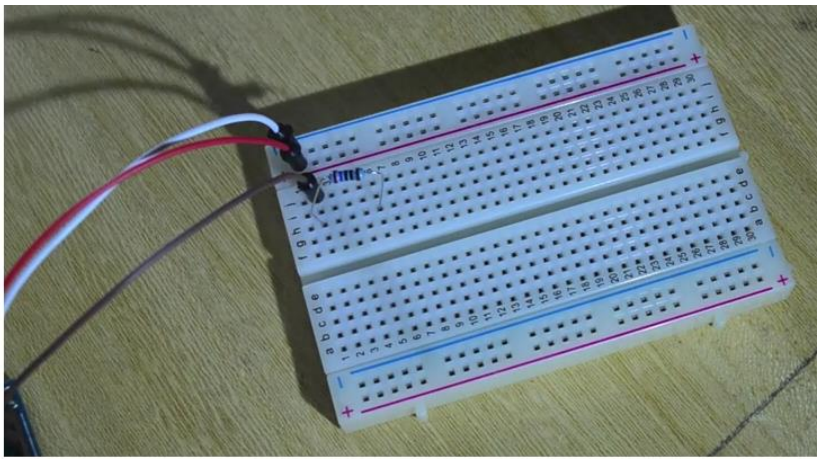
Spajanje komponenti (Slika 4. – Slika 7.)

i) Prvi korak pri spajanju Croduino Basic2 kontrolne pločice i eksperimentalne pločice jest povezivanje izvora napajanja +5V s eksperimentalnom pločicom na pozitivan pol te uzemljenja gnd na negativan pol, pomoću žica za povezivanje. (Slika 4.)



Slika 4. Povezivanje +5V i gnd pinova s eksperimentalnom pločicom [8]

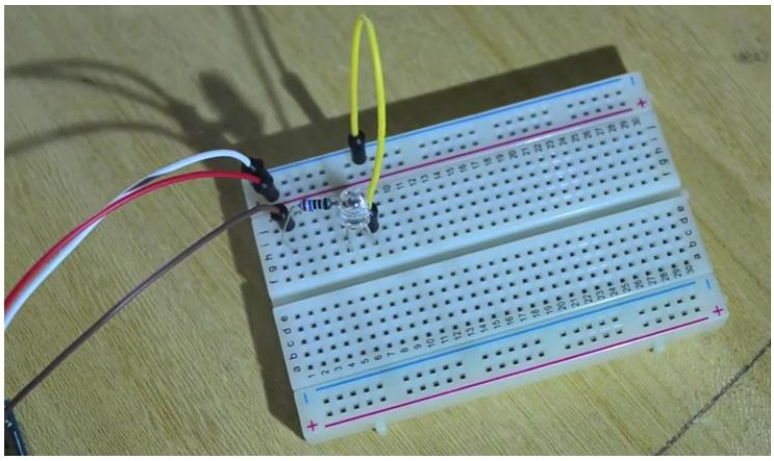
ii) Sljedeći korak je spajanje digitalnog pina (u ovom slučaju D13) s proizvoljnim redom na eksperimentalnoj pločici. Da bi se spriječilo pregaranje LED diode uslijed prevelike struje, između dovoda napajanja i trošila (LED diode) umeće se otpornik od 330Ω na način da je jedna nožica otpornika spojena u isti red kao i digitalni pin, a druga u proizvoljni različiti red. (Slika 5.)



Slika 5. Spoj digitalnog pina i otpornika na eksperimentalnoj pločici [8]

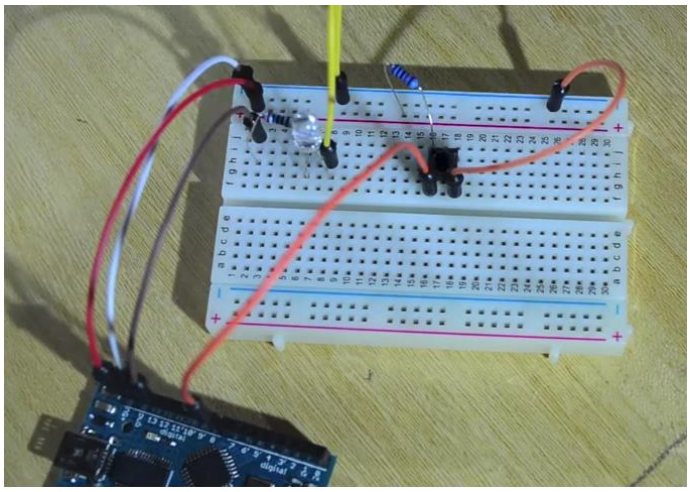
LED dioda ima 2 nožice, dužu i kraću. Duža nožica predstavlja pozitivan pol, a kraća negativan, na što treba obratiti pozornost pri povezivanju. U ovome slučaju,

pozitivna nožica LED diode bit će spojena u red u kojem je druga nožica otpornika, a kraća u prazan red. Kraća nožica se zatim pomoću (žute) žice povezuje s negativnim polom na eksperimentalnoj pločici kako bi se uzemljila. (Slika 6.)



Slika 6. Povezivanje LED diode na eksperimentalnoj nožici [8]

iii) Da bi se LED dioda mogla proizvoljno paliti i gasiti, na proizvoljni red umeće se prekidač koji može propustiti ili spriječiti dovod napajanja te se povezuje s pozitivnim polom na eksperimentalnoj pločici pomoću otpornika od 10000Ω. Kada je tipka pritisnuta, dovod struje je spriječen. U istom redu u kojem je prekidač spojen s pozitivnim polom, prekidač se pomoću žice povezuje s digitalnim pinom (u ovom slučaju D8) na Croduino kontrolnoj pločici. Drugi kraj prekidača se povezuje s negativnim polom na eksperimentalnoj pločici. Kada je prekidač uključen on propušta napajanje do LED diode preko otpornika te lampica svijetli, a kad nije uključen napajanje ne prolazi kroz prekidač te lampica ne svijetli. (opisano prema e-radionici [8]) (Slika 7.)



Slika 7. Povezivanje prekidača s eksperimentalnom i kontrolnom pločicom [8]

Spajanjem prekidača hardverski dio je završen te preostaje napisati i učitati program na kontrolnu pločicu.

Pisanje programa (Slika 8.)

i) Pisanje svakog Arduino programa započinje s unošenjem dvije naredbe, *void setup* i *void loop* iza kojih dolaze otvorena i zatvorena zagrada te otvorena i zatvorena vitičasta zagrada unutar kojih se nalazi dio programa koji se odnosi na određenu naredbu, (Slika 3.). Naredba *void setup* izvodi se samo jednom na početku, a *void loop* se neprestano ponavlja dok postoji napajanje.

ii) Potrebno je definirati komponente koje su spojene na digitalne pinove. Komponente se definiraju s naredbom *int*, imenom koji se daje komponenti, znakom jednakosti i brojem digitalnog pina na koji je komponenta spojena. Svaka riječ i znak moraju biti odvojeni razmakom, a na kraju reda dolazi točka zarez. Definicije se upisuju iznad *void setup* i *void loop* naredbi. U ovome primjeru spojene su dvije komponente te se definiraju kao:

int tipka = 8;

int LED = 13;

iii) Osim dvije digitalne komponente mora se definirati i varijabla koja će očitavati stanje tipke, odnosno da li je ona pritisnuta ili nije. Iza te definirane varijable ne upisuje se znak jednakosti i broj pina, čime se programu govori da je ta varijabla brojčana vrijednost, zasad nedefinirana. Varijabli se dalo ime „vrijednost“:

int vrijednost;

iv) Unutar vitičastih zagrada u *void setup* naredbi opišemo digitalne komponente pomoću naredbe *pinMode* na način da za svaku komponentu odredimo odnosi li se na input ili output, to jest šalje li signal ili ga prima. Opisivanje se radi unutar zagrada iza naredbe *pinMode* tako da se upiše ime komponente, zatim odvoji zarezom i razmakom te zatim upiše INPUT ili OUTPUT. U ovom primjeru dvije komponente opisane su na sljedeći način:

pinMode(tipka, INPUT);

pinMode(LED, OUTPUT);

v) Unutar naredbe *void loop*, koja se neprestano ponavlja, definira se da je varijabla „vrijednost“ ono što digitalni pin očita tamo gdje je povezana tipka. To se radi pomoću naredbe *digitalRead* iza koje slijedi zagrada unutar koje se upisuje na koji digitalni pin se naredba odnosi.

vrijednost = digitalRead(tipka);

vi) Sada je kontrolnoj pločici zadano da očitava je li tipka pritisnuta ili nije, odnosno propušta li struju ili ne. Prema tome može se zadati da u jednom slučaju pločica propušta struju LED diodi, a u drugome ne propušta, dakle da ju pali ili gasi prema stanju tipke. To se radi pomoću naredbi *if* i *else* što znači da kontrolna pločica radi neku radnju ukoliko su zadovoljeni određeni uvjeti. U ovome slučaju definiramo:

if (vrijednost == HIGH){

digitalWrite(LED, LOW);

}

Iza naredbe *if* dolazi zagrada unutar koje se definira uvjet. U ovom slučaju uvjet je da zadana brojčana varijabla vrijednost mora biti HIGH odnosno da tipka propušta struju što znači da nije pritisnuta. Dva znaka jednakosti predstavljaju uvjet „ako i samo ako“ koji govori da je uvjet zadovoljen samo u slučaju da je vrijednost HIGH. Nakon zatvorene zagrade, otvara se vitičasta zagrada unutar koje se zadaje što se treba izvršiti ukoliko je uvjet zadovoljen. Naredba *digitalWrite* dovori da se na određeni pin šalje digitalni signal koji prenosi

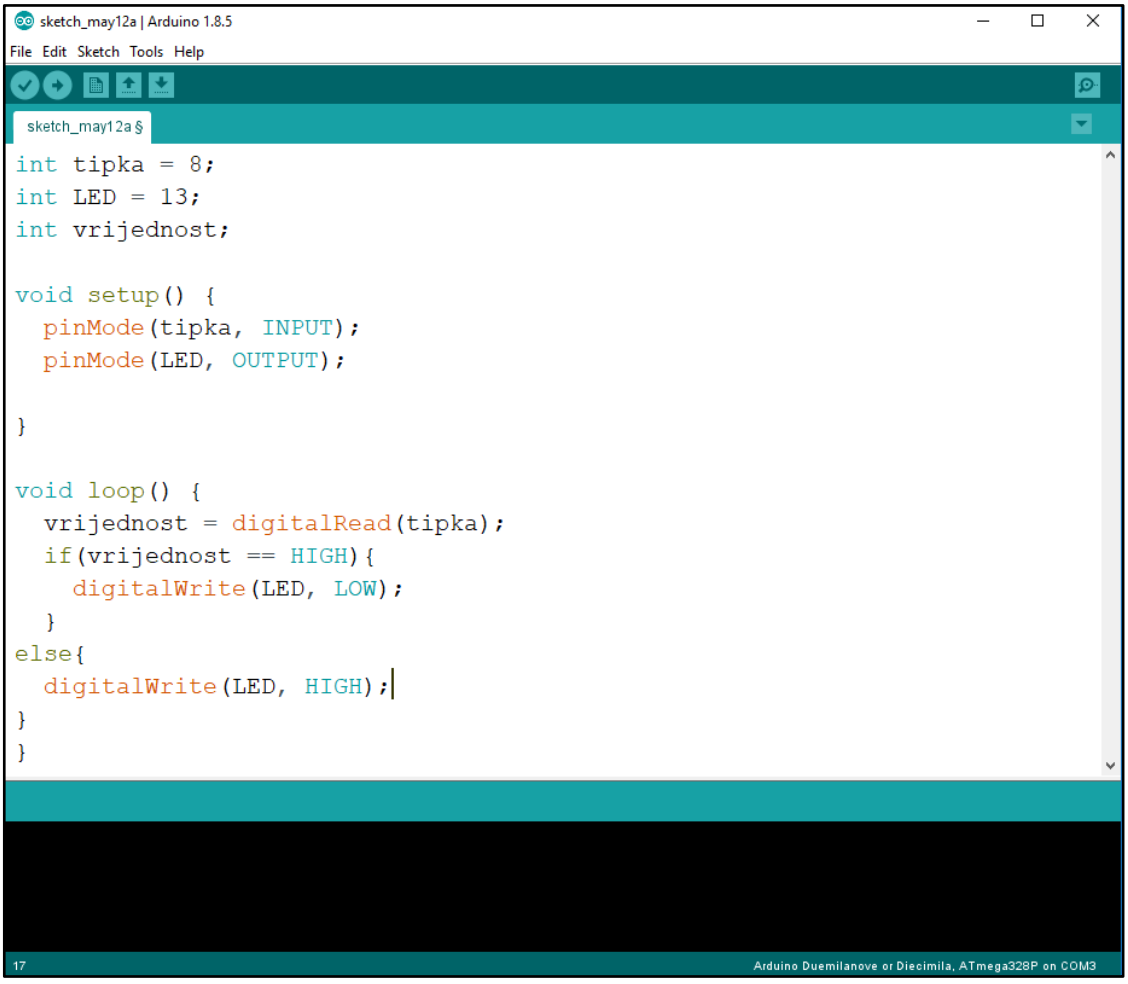
informaciju o odrađivanju neke radnje. Ovdje je radnja da se LED lampica na digitalnom pinu ugasi ukoliko tipka propušta struju što se događa kad tipka nije pritisnuta.

Za slučaj kada uvjet nije ispunjen upisuje se naredba *else* koja govori što pločica u tom slučaju radi. U ovom primjeru to znači da tipka ima vrijednost LOW, odnosno da je pritisnuta i ne propušta struju te se upisuje:

else{

digitalWrite(LED, HIGH);

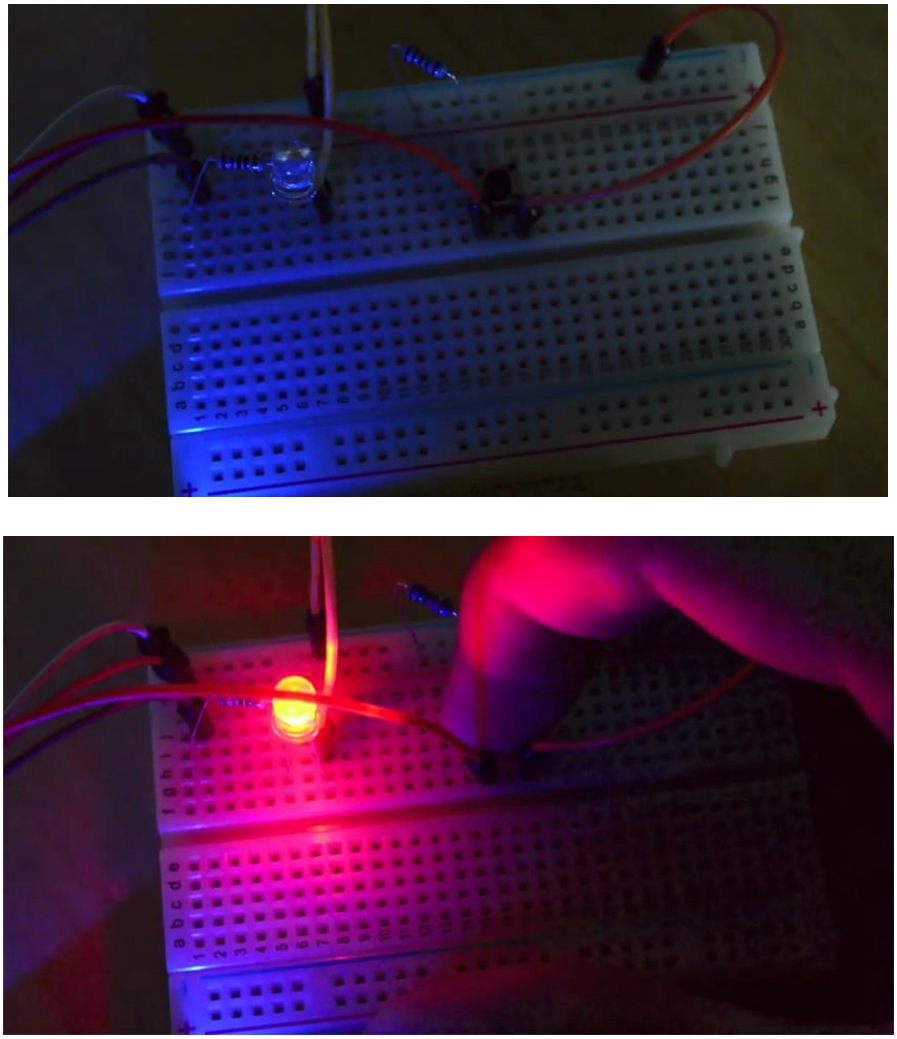
}



Slika 8. Završen program za paljenje i gašenje LED diode pomoću tipke

Sada je program kompletiran i potrebno ga je učitati na kontrolnu pločicu pomoću mini-b USB priključka. Programom je definirano da digitalni pin 13 propušta struju LED diodi i pali je u slučaju kada digitalni pin 8 ne dobiva signal od prekidača, to jest kada je

tipka pritisnuta. U suprotnom slučaju, kada tipka nije pritisnuta LED diodi se ne propušta struja i ona je ugašena. (Slika 9.)



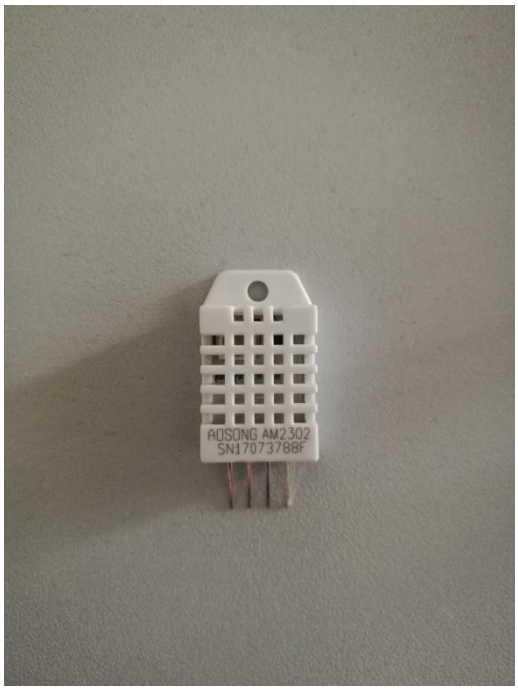
Slika 9. Tipka nije pritisnuta - lampica ne svijetli, tipka je pritisnuta – lampica svijetli [8]

2.4. Mjerenje vlažnosti zraka i temperature pomoću Arduino platforme

Jedna od brojnih mogućnosti koje Arduino platforma pruža je i mjerenje parametara okoliša, kao što su temperatura, tlak zraka, vlažnost zraka, koncentracija plinova, radijacija i slično. U ovom poglavlju prikazano je mjerenje temperature i vlažnosti zraka pomoću senzora DHT22, u trajanju od pet sati. Dobiveni podaci mjerenja su uspoređeni s podacima Državnog Hidrometeorološkog Zavoda.

DHT22 (Slika 10.) je senzor za mjerenje temperature i vlažnosti zraka pomoću kapacitivnog senzora vlage i termootpornika. Podaci se šalju na kontrolnu pločicu

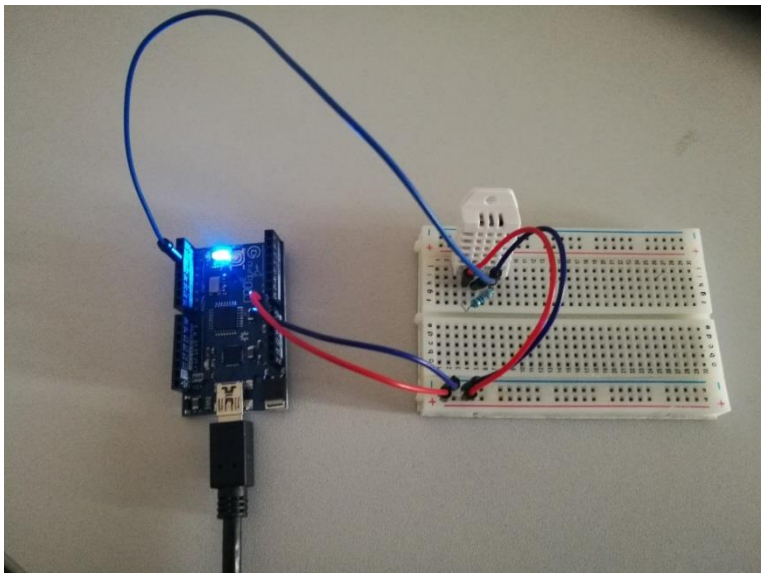
pomoću digitalnog signala. Ima 4 pina od kojih se jedan spaja na napajanje 3-5V, drugi se spaja na digitalni pin te služi za prijenos podataka, treći služi za uzemljenje, a četvrti nije potrebno spajati. Raspon mjerenja temperature je od -40°C do 80°C, a vlažnosti od 0 do 100%. Temperaturu očitava s preciznošću ±0,5°C, a vlažnost zraka s ±2 - 5%. [9]



Slika 10. DHT22 senzor za mjerenje temperature i vlažnosti zraka

Za spajanje DHT22 senzora (Slika 11.) i očitavanje temperature te njihov grafički prikaz, potrebni su:

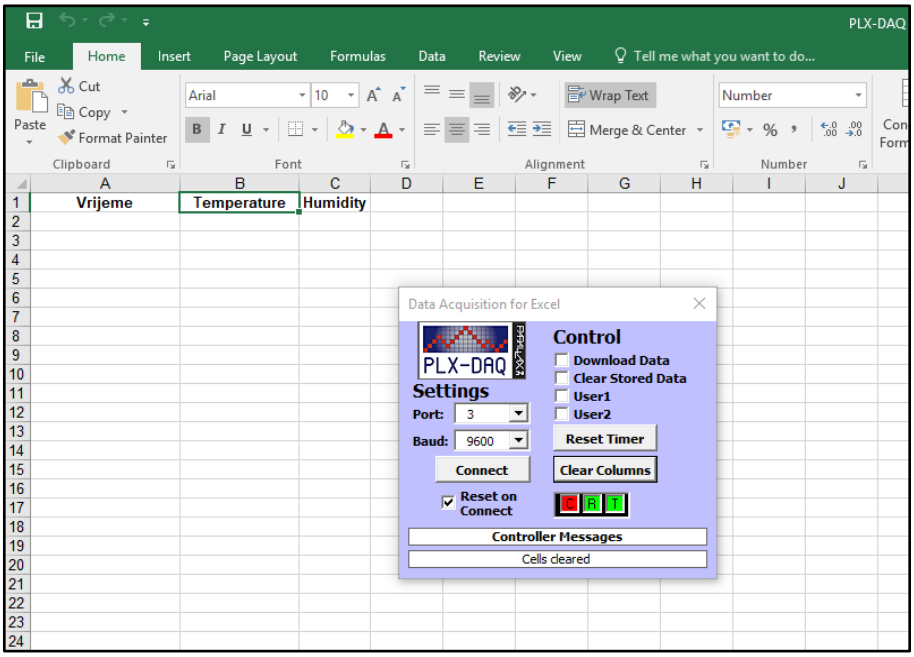
- DHT22 senzor
- Croduino Basic2 kontrolna pločica
- eksperimentalna pločica
- žice za povezivanje
- otpornik od 10Ω
- osobno računalo
- kabel za povezivanje s osobnim računalom
- Arduino softver
- PLX-DAQ softver za prijenos podataka mjerenja u Microsoft Excel



Slika 11. Spoj DHT22 senzora s Croduino Basic2 kontrolnom pločicom

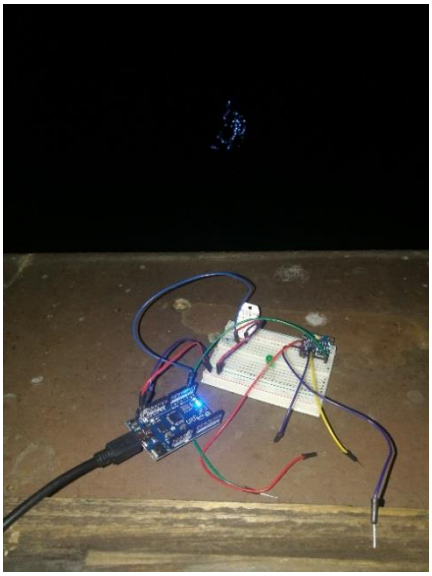
Arduino platforma je open-source što znači da korisnici svoje programe mogu dati na raspolaganje drugim korisnicima besplatno. Tako Arduino softver sadrži knjižnicu gotovih programa za određene senzore i druge elektroničke spojeve, među kojima je i DHT22. [10,11] Svi su programi napisani tako da se očitavanja mogu dobiti samo unutar Arduino softvera. Ako se mjerenja žele zapisati izravno u Microsoft Excel tablicu (Prilog 1.), kod programa je potrebno preraditi da bi se mogao povezati s PLX-DAQ softverom (Slika 12.).

Kada se mjerenja zapisuju u Microsoft Excel, moguće ih je kontinuirano grafički prikazivati. Arduino softver ne pruža tu mogućnost, stoga je nužno uz Arduino koristiti i druge programe za prikazivanje podataka mjerenja.



Slika 12. PLX-DAQ softver u spoju s Microsoft Excelom

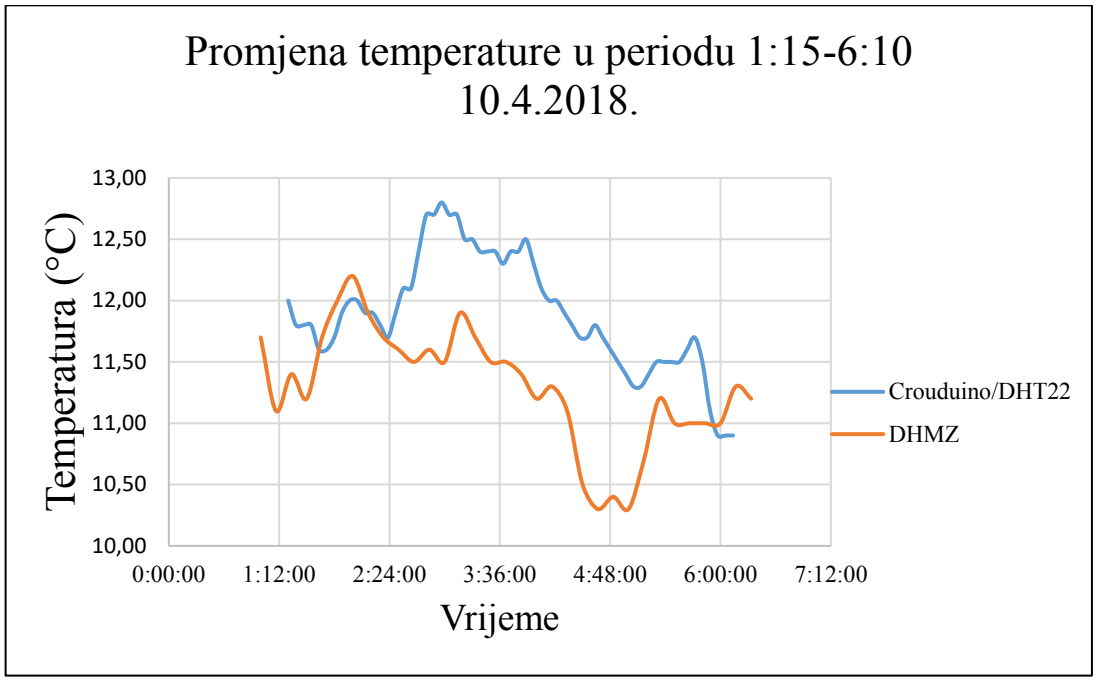
Mjerenje za potrebe ovog rada obavljeno je 10.4.2018. u periodu od 1:15 do 6:10 sati. Senzor je postavljen s vanjske strane prozora na visini od otprilike 10 metara. (Slika 13 a.)



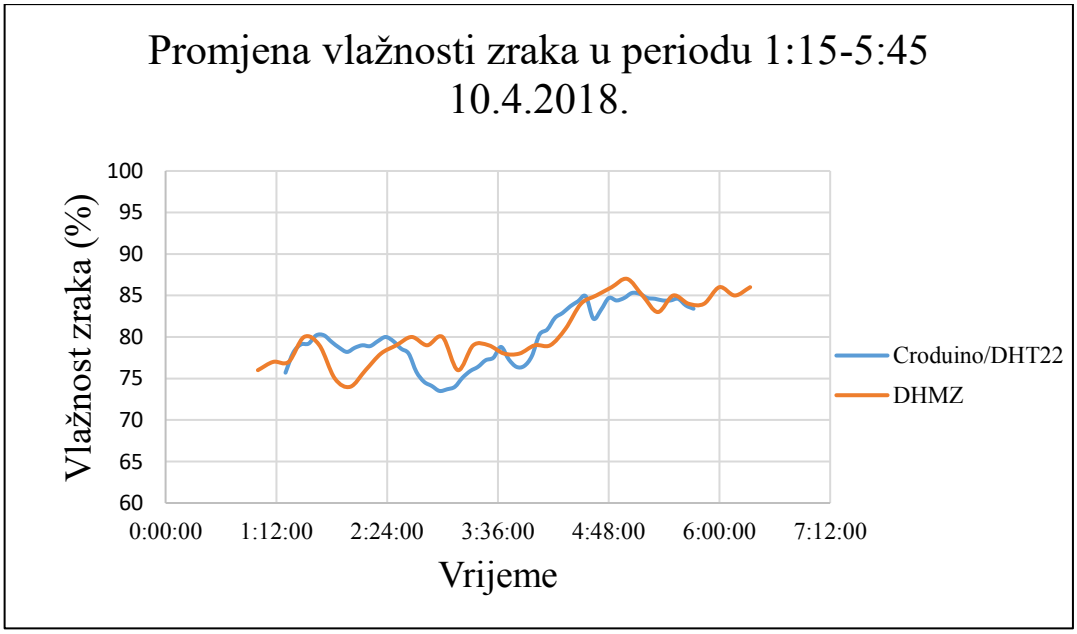
Slika 13 a. Postavljanje DHT22 senzora na vanjski dio prozora

Program je napravljen tako da bilježi temperaturu i vlažnost zraka svakih 5 minuta te su zabilježeni rezultati mjerenja prikazani u usporedbi s podacima zaprimljenih od Državnog hidrometeorološkog zavoda. Iz rezultata mjerenja je vidljivo da grafički prikazane vrijednosti prate jedna drugu, odnosno da postoje jednaki trendovi u

rezultatima snimanja Arduino platformom i službenim mjernim uređajima. U mjerenju temperature postoje odstupanja veća nego što je predviđeno u specifikacijama senzora, ali to se može pripisati lokaciji mjerenja. Rezultati mjerenja vlažnosti su u granicama odstupanja 2-5% koliko je i predviđeno, stoga se eksperiment može smatrati uspješnim.



Slika 13 b. Promjena temperature zabilježena DHT22 senzorom u usporedbi s podacima DHMZ-a



Slika 13 c. Promjena vlažnosti zraka zabilježena DHT22 senzorom u usporedbi s podacima DHMZ-a

3. Titranje, valovi, vibracije u okolišu

U ovom poglavlju obrazložene su teoretske osnove titranja i valova bez kojih nije moguće razumijevanje pojava vibracija u okolišu te njihovo interpretiranje.

3.1. Titranje

Titranje predstavlja svako gibanje koje se ponavlja u nekom vremenskom intervalu. Može biti cikličko ili periodično, a za svako gibanje mehaničkih sustava potrebna je vanjska periodična sila ili otklon iz ravnotežnog položaja. Razlika između vibracije i titranja jest u veličini otklona od ravnotežnog položaja, s obzirom na veličinu mehaničkog sustava. Kada se promatraju mali otkloni visoke frekvencije, uglavom govorimo o vibracijama. Primjer titranja jest matematičko njihalo, a primjer vibracije jesu vibracije vratila ili osovinskog voda pri prijenosu snage ili gibanja. Prilikom svakog pojedninog titraja dolazi do pretvorbe potencijalne energije u kinetičku i obrnuto. Kad ne bi postojale sile prigušenja, jednom pokrenuti mehanički sustav vibrirao bi bez prestanka. Uslijed trenja, otpora zraka i drugih sila prigušenja, dolazi do pretvorbe kinetičke energije u toplinsku, mehaničku ili neku drugu energiju, postepenog smanjenja amplitude i vraćanja sustava u ravnotežni položaj i stanje mirovanja. [12,13,14]

Ukoliko se sustav otkloni iz ravotežnog položaja i prepusti gibanju, govorimo o slobodnim prigušenim ili neprigušenim vibracijama. U slučaju kada na sustav djeluje periodična uzbudna sila, riječ je o prisilnim vibracijama. Prisilne vibracije koriste se u strojevima za usitnjavanje materijala, kao što su drobilice, kod mehaničkih sita i dr. U konstrukcijama, vibracije predstavljaju dinamičko opterećenje te mogu dovesti do sloma dijelova sustava. [13,14]

Najjednostavniji oblik vibracijskog sustava jest masa na opruzi, (Slika 14.). Jednom kada se masa otkloni iz ravnotežnog položaja, sila u opruzi, koja potječe iz konstante elastičnosti opruge, djelovat će na masu te će sustav oscilirati sve dok sile prigušenja ne zaustave gibanje, odnosno dok se sva kinetička energija ne pretvori u druge oblike energije. Praćenjem gibanja sustava masa-opruga kroz vrijeme može se primijetiti da je gibanje harmonijsko te da se može opisati sinusoidom (ukoliko se zanemare sile prigušenja). (Slika 15.)

Jednadžba takvog harmonijskog oscilatora glasi:

$$\frac{d^2x(t)}{dt^2} + \frac{K}{m} * x(t) = 0 \text{ [16]}$$

Gdje je:

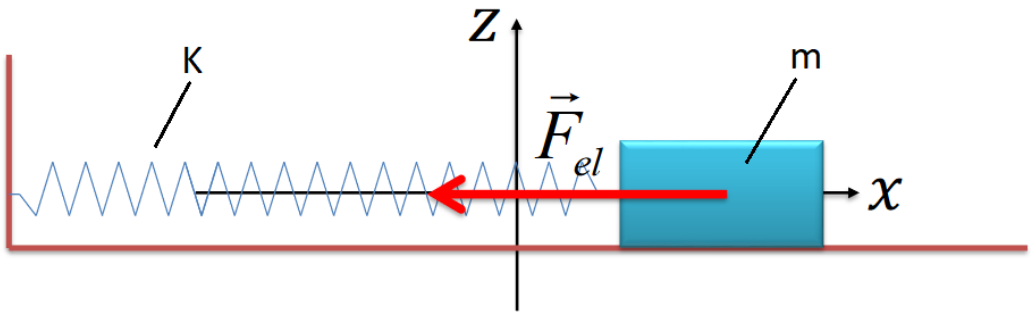
- K – konstanta elastičnosti opruge [N/m]
- m – masa na opruzi [kg]
- x(t) – otkon mase u ovisnosti o vremenu [m]
- t – vrijeme [s]

Rješenje te diferencijalne jednadžbe i njen uobičajen zapis u fizici glasi:

$$x(t) = A\ sin(\omega t + \varphi_0) \text{ [16]} \tag{3.1}$$

Gdje je:

- A – amplituda [m]
- ω – kružna frekvencija [rad/s]
- t – vrijeme [s]
- φ_0 – početna faza [rad]

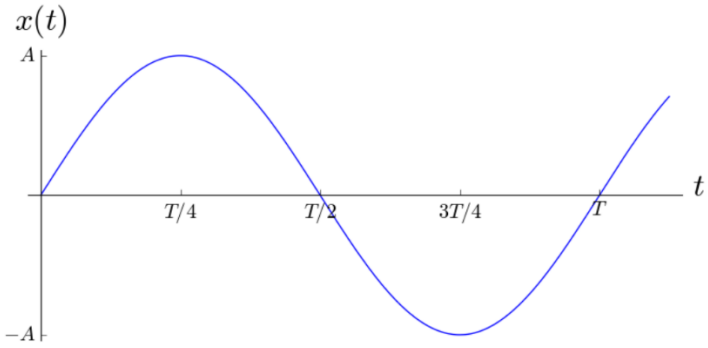


Slika 14. Harmonijski oscilator na idealno glatkoj podlozi [16]

Kružna frekvencija ovisi o masi na opruzi i konstanti elastičnosti opruge te se može izračunati izrazom:

$$\omega = \sqrt{\frac{K}{m}} = \frac{2\pi}{T} \left[\frac{rad}{s}\right] \tag{3.2}$$

gdje je T period titranja, odnosno vrijeme između dva brijega ili dva dola sinusoide.



Slika 15. Graf kretanja mase na opruzi u vremenu [16]

Prvom derivacijom otklona čestice po vremenu (3.1) dobiva se brzina titranja čestice:

$$v(t) = \omega A \cos(\omega t + \varphi_0) \qquad [16] \tag{3.3.}$$

3.2. Valovi

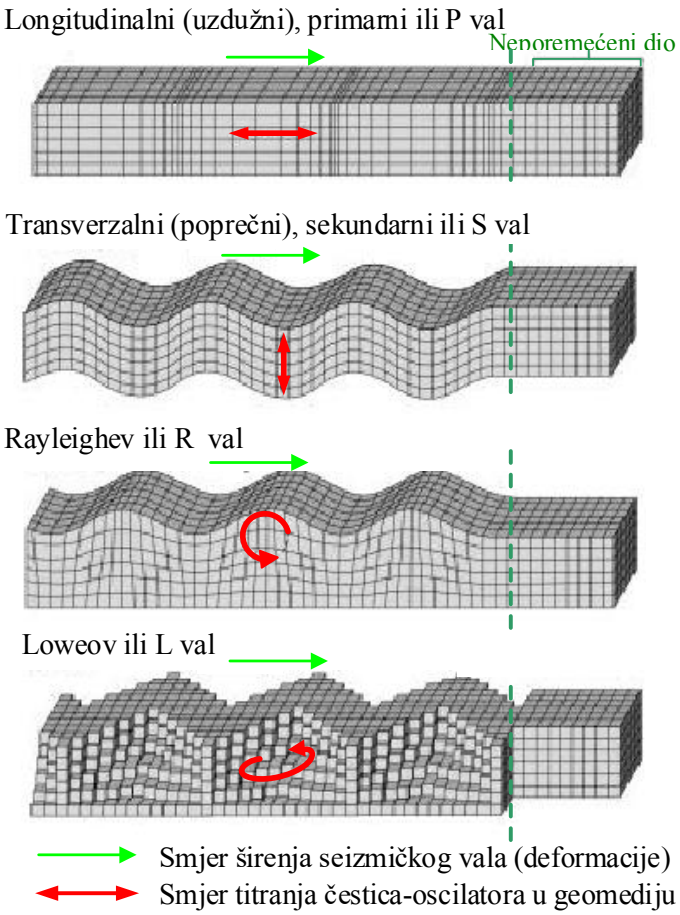
Mehanički valovi predstavljaju širenje poremećaja kroz neki medij, pri čemu čestice medija titraju oko ravnotežnog položaja ili proces zahvaćanja titranjem sve udaljenijih čestica-oscilatora. Mehaničkim valovima ne prenosi se materija, već samo energija. Medij u kojem se šire valovi može biti čvrsti, tekući ili plinoviti. Kroz čvrsti medij valovi se šire u obliku elastičnih deformacija. U ravnotežnom stanju, elementarni oscilatori raspolažu elastičnom potencijalnom i kinetičkom energijom što znači da pri širenju elastičnih deformacija dolazi i do širenja mehaničke energije pohranjene u deformacijama. [15, 16]

Brzina širenja procesa titranja s jednog elementarnog oscilatora na drugi tj. brzina širenja elastične deformacije i energije pohranjene u njoj, naziva se fazna brzina i treba je strogo razlikovati od brzine titranja elementarnog oscilatora oko ravnotežnog položaja.

S obzirom na smjer titanja elementarnih oscilatora, odnosno njihovog položaja u odnosu na smjer širanja vala, razlikuju se longitudinalni (uzdužni) valovi i transverzalni (poprečni) valovi. U longitudinalnim valovima čestice titraju u smjeru širenja vala, dok u transverzalnim titraju okomito na širenje vala te je za njihovo širenje potrebno postojanje posmične povratne sile, stoga njihovo postojanje nije moguće u plinovima i tekućinama.

Prema broju dimenzija kontinuuma u kojem se val širi, razlikuju se 1D, 2D i 3D valovi. Primjer 1D vala je val na žici, primjer 2D vala je val na površini vode, a primjer 3D vala je zvuk koji se širi u prostoru. [15,16]

Kada se teoretski promatraju tlo ili stijene, pretpostavlja se da je riječ o homogenom, elastičnom i izotropnom mediju. Izazivanjem dinamičkog naprezanja, primjerice u nekoj točki blizu površine, u takvom mediju nastaju 3 vrste valova. Prvi val naziva se i primarnim ili P valom te je s obzirom na smjer titranja elementarnih oscilatora longitudinalan. On u mediju uzrokuje promjenu volumena, stezanje i rastezanje. Drugi val naziva se sekundarnim ili S valom te je on po smjeru titranja čestica transverzalan. On ne uzrokuje promjene volumena, ali uzrokuje promjene oblika elementarnih kvadratića u rombove. Treći val je površinski i naziva se Rayleighevim ili R valom te se rasprostire uz slobodnu površinu elastične čvrste mase, jer mu amplituda naglo pada s dubinom. R val se smatra kompozicijom P i S vala u rubnim uvjetima, odnosno u blizini površine. Moguća je i pojava Loweovog vala, iako on nije karakterističan za seizmičke pokuse. (Slika 16.) [15,17]



Slika 16. Longitudinalni, transverzalni, Rayleighev i Loweov val [17]

Seizmički valovi se opisuju su pomoću valne duljine λ, brzine širenja V i frekvencije ν. Valna duljina je najmanja udaljenost između oscilatora koji imaju isti otklon ili udaljenost između dva susjedna brijega (rastezanja) ili dva susjedna dola (stezanja). Frekvencija je broj titraja harmonijskog oscilatora u jednoj sekundi, a izražava se u hertzima (Hz). Izražava se omjerom 1/T, gdje je T period, odnosno vrijeme između dva susjedna brijega ili dva susjedna dola. Ove vrijednosti međusobno su zavisne te se mogu izraziti formulom:

$$\lambda = \frac{V}{\nu}\ [m] \quad [15] \tag{3.3}$$

Brzine primarnih i sekundarnih valova u čvrstom mediju ovise o elastičnim konstantama (Youngov modul elastičnosti E i Poissonov koeficijent σ) i gustoći ρ te se mogu izraziti formulama:

$$Vp = \sqrt{\frac{E}{\rho} * \frac{1-\sigma}{2(1-2\sigma)(1+\sigma)}}\ [\frac{m}{s}] \quad [15] \tag{3.4}$$

$$Vs = \sqrt{\frac{E}{\rho} * \frac{1}{2(1+\sigma)}}\ [\frac{m}{s}] \quad [15] \tag{3.5}$$

Omjer brzine primarnih valova i brzine sekundarnih valova ovisi samo o Poissonovom koeficijentu:

$$(\frac{Vp}{Vs})^2 = \frac{2-2\sigma}{1-2\sigma}\ [15]$$

Uzmemo li za primjer beton, koji ima Poissonov koeficijent σ = 0,2, dobije se da je

$$Vp = 1{,}634\ Vs.$$

Površinski valovi su složeniji od primarnih i sekundarnih te su nešto sporiji. Njihova brzina širenja jest oko 0,9Vs. U seizmičkim istraživanjima njihova je pojava nepoželjna te se prikazuje kao smetnja ili šum, osim ako se ne radi o metodama koje koriste upravo površinske valove. [15]

3.3. Vibracije u okolišu

Vibracije u okolišu promatraju se kao negativna pojava, jer mogu značajno utjecati na zdravlje ljudi, društvenu i gospodarsku aktivnost i nastanak konstruktivnih i estetskih šteta na građevinama. Problemi uzrokovani vibracijama najčešće su vezani uz učestale pojave vibracija malih amplituda, pojave vibracija velikih amplituda (potresi) ili diferencijalna slijeganja uzrokovana prepraspodjelom čestica tla. Primjer učestalih vibracija malih amplituda je promet koji može izazvati konstruktivna oštećenja na objektima u području mikrotremora, a popraćen je i zvučnim onečišćenjem što može negativno utecati na zdravlje i razdražljivost ljudi. [18] Prema ISO standardima definirana su pravila o mogućem ljudskom izlaganju vibracijama i šoku, vibracijama u vozilima, vibracijama vezanim za posebne uređaje i strojeve te vibracija u građevinama. Unatoč tome, nedostaje konzistentnih načina definiranja vibracija, njihove evaluacije i prikaza. Ovisno o aspektu vibracija koji im je najvažniji, različiti autori različito kategoriziraju vibracije kao npr. prema mjestu nastanka, dostupnoj opremi za njihovo mjerenje, vezi između vršne brzine titranja, prigušenja i energije, utjecaju na ljude i objekte te prema zakonski uspostavljenim graničnim kriterijima. [18]

Ovisno o učestalosti pojavljivanja, vibracije se mogu kategorizirati kao kontinuirane (neprekidne) ili tranzijentne (kratkotrajne, prijelazne). Kategorizira se na temelju usporedbe trajanja gibanja tla τ_r s vremenskom konstantom koja je definirana kao:

$$\tau_f = \frac{1}{2\pi\xi_r f_r} \qquad [18] \qquad (3.6.)$$

gdje je ξ_r koeficijent prigušenja određene vrste vibracija, a f_r rezonantna frekvencija pridužena toj vrsti vibracija. U slučaju kada trajanje gibanja tla prelazi $5\tau_f$ riječ je o kontinuiranim vibracijama, u suprotnom su one tranzijentne. Moguća je pojava i treće kategorije, odnosno intermitiranih vibracija koje nastaju kao periodično ponavljanje tranzijentnih podražaja. Primjer kontinuirane vibracije jest promet, tranzijentne miniranje, a intermitirane zabijanje pilota. [18]

Vibracije se mjerenje sustavima koji se sastoje od geofona (senzora za mjerenje brzine vibracija) i/ili akcelerometra (senzora za mjerenje ubrzanje), pojačala, filtara i sustava za snimanje. Ovisno o karakteru promatrane vibracije, potrebno je odrediti frekvenciju uzorkovanja. Mjerena vibracija prikazuje se kao krivulja frekventne reakcije u kojoj je promjena brzine ili akceleracije (ovisno o mjernom uređaju) funkcija vremena.

4. Mjerenje vibracija pomoću Arduino platforme

Konačni cilj ovog rada jest izmjeriti vibracije pomoću senzora kompatibilnog s Arduino platformom i prikazati dobivene rezultate u usporedbi s rezultatima mjerenja profesionalnim uređajem. Izvedeno je nekoliko serija mjerenja, unutar i izvan građevine (otvoreni i zatvoreni prostor), pri čemu su se registrirala podrhtavanja, odnosno vibracije namjerno izazvane dinamičkim impulsom, udarom čekića mase 6 kg.

4.1. Oprema korištena pri ispitivanju

Sustav za mjerenje na Arduino platformi korišten u ovom ispitivanju sastojao se od:

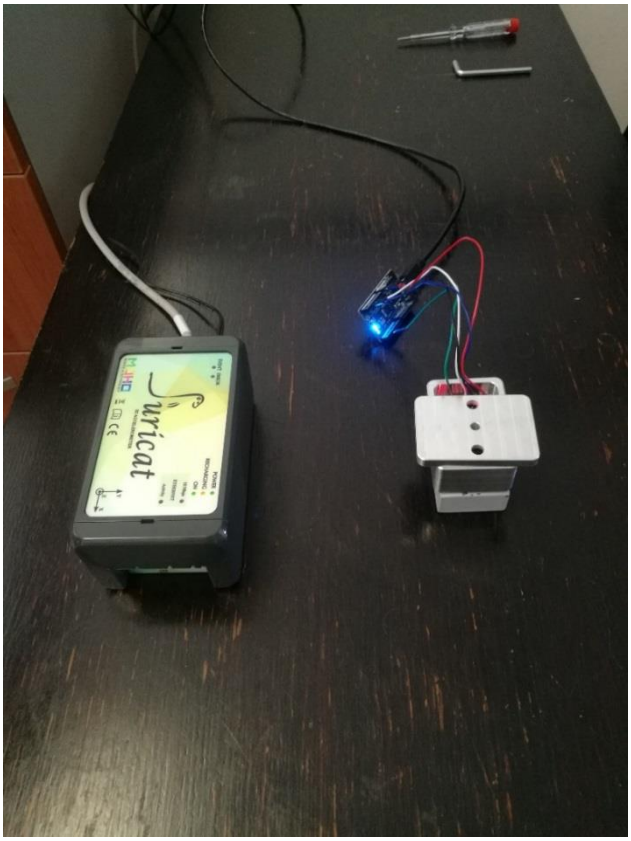
- Croduino Basic2 pločice
- MPU6050 akcelerometra
- kućišta za akcelerometar
- Arduino programa za akcelerometar (Prilog 2.)
- programa za snimanje (izrađen u C Sharpu)

Slika 17. MPU6050 akcelerometar fiksiran u kućištu te spojen na Croduino Basic2 pločicu

MPU6050 akcelerometar koristi napon 3-5 V i ima raspon: ±2g, ±4g, ±8g, ±16 g ovisno o tome kako je zadano u programu. [19] Spojen je na Croduino Basic2 pločicu preko 5 izlaza na sljedeći način:

- VCC na 3,3 V
- GND na GND
- SDA na A4 (analogni pin 4)
- SCL na A5 (analogni pin 5)
- INT na D2 (digitalni pin 2)

Dobiveni rezultati mjerenja usporedili su se s podacima dobivenim uređajem Suricat. Suricat je 3D akcelerometar za kratkotrajan ili dugotrajan monitoring. Namijenjen je potrebama u inženjerstvu za registriranje poremećaja prilikom potresa ili drugih izvora vibracija. [20]



Slika 18. 3D akcelerometar Suricat i Croduino Basic2 povezan s MPU6050 senzorom (unutar kućišta)

4.2. Mjerenja unutar građevine (u zatvorenom prostoru)

Sljedeća mjerenja izvedena su u kabinetu na Geotehničkom fakultetu u Varaždinu. Mjerni uređaji (MPU6050 + Croduino Basic2 i Suricat) postavljeni su tako da su im x osi paralelne. (Slika 18.) Pri mjerenju, stol na kojem su postavljeni uređaji udaren je tri puta te su paralelno praćeni rezultati mjerenja oba uređaja. (Slika 19.) Prema oblicima dobivenih krivulja promjene akceleracije, odmah je bila vidljiva podudarnost rezultata Arduino sustava i Suricata.

Interval mjerenja Arduino platformom bio je oko 15 sekundi, a snimanje Suricat akcelometrom je trajalo sat vremena te je stoga bilo potrebno iz jednosatnog zapisa izvući točno onaj isječak koji je u istom vremenu kao i snimanje Arduinom te uskladiti krivulje kako bi se intervali poklapali. Očitanja dobivena Arduino platformom bilo je potrebno transformirati da bi imala standardni zapis. Pretpostavilo se da u stanju mirovanja akcelerometra, Arduino očitava vrijednost -7595. Arduino koristi 16bitni zapis uz osjetljivost ±8g te je prema tome potrebno transformirati zapis na sljedeći način:

2^{16} (16 bitni zapis) = 65 536

65536 / 8 = 8192 (osjetljivost 8g) => 1 g = 8192

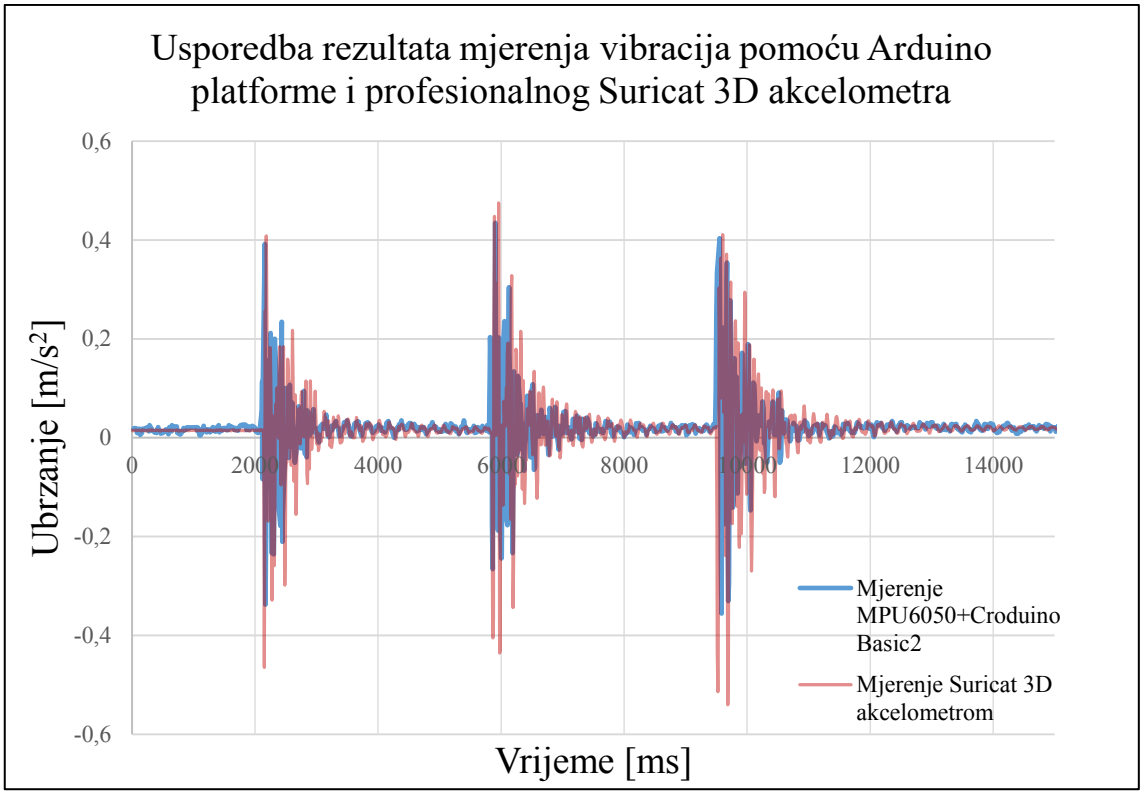
0 m/s^2 = (-7595/8192) * 9,81 m/s^2 + (7595/8192) * 9,81 m/s^2 (kalibracija senzora)

Pomoću programa za snimanje rezultata, zapisane su vrijednosti očitanja Arduino platforme u .txt datoteci, a u Microsoft Excelu su pretvorene u vrijednosti u m/s^2 te je napravljen grafički prikaz.

Suricat 3D akcelerometar očitava vrijednosti ubrzanja u mg (mili g), stoga je njegova očitanja bilo potrebno podijeliti s 1000 i pomnožiti s 9,81 da bi vrijednosti bile u m/s^2.

Grafički prikaz rezultata potvrđuje prvobitni dojam da su rezultati mjerenja s oba uređaja slični. Iako postoje odstupanja u krivuljama, ona su relativno mala, a maksimalne amplitude gotovo se preklapaju za pozitivne vrijednosti, dok su odstupanja negativnih vrijednosti nešto veća. To se može pripisati različitim osjetljivostima senzora i različitim frekvencijama snimanja. Arduino platforma postigla je frekvenciju snimanja 100 Hz, a Suricat uređaj ima frekvenciju 128 Hz što dovodi do detaljnijeg i preciznijeg

zapisa. Također je uočljivo da Suricat 3D akcelerometar dulje i osjetljivije registrira vibracije, dok Arduino platforma ima nešto kraće intervale titranja.



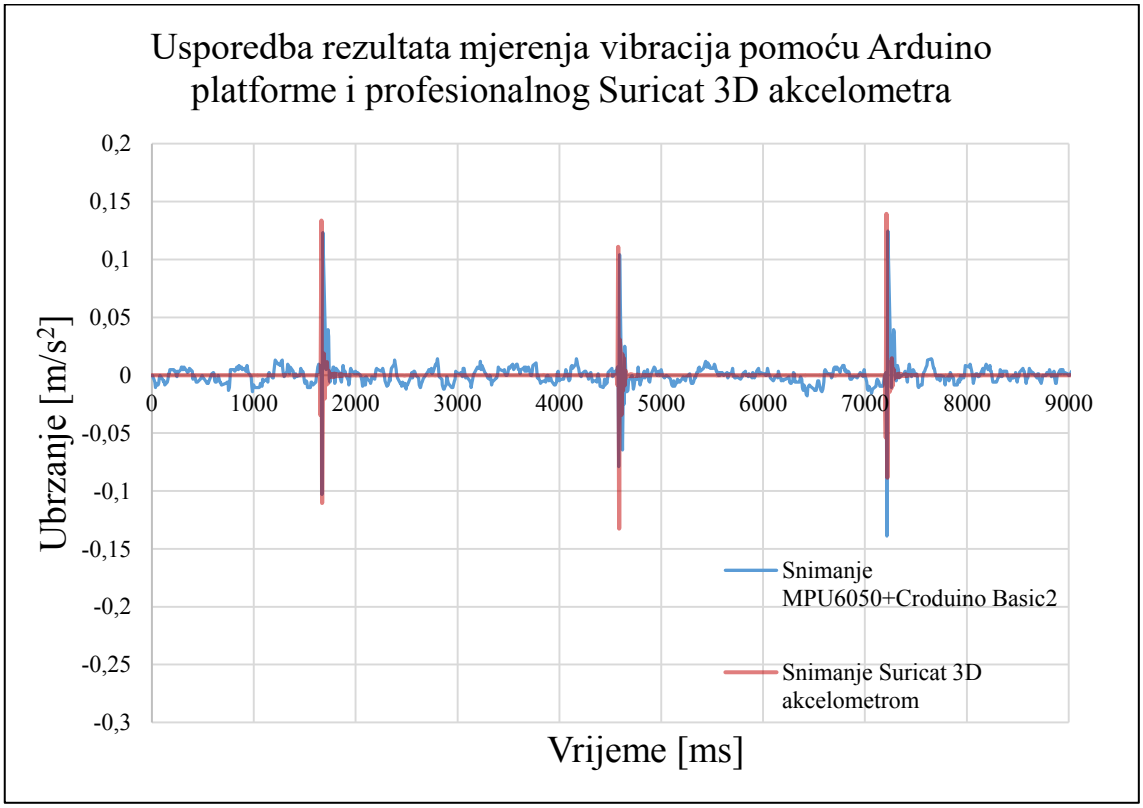
Slika 19. a. Usporedba rezultata mjerenja vibracija u zatvorenom prostoru pomoću Arduino platforme i Suricat 3D akcelometra

Slika 19. b. Praćenje rezultata snimanja, lijevo Suricat 3D akcelerometar (prati se gornja os x), desno Arduino platforma

4.3. Mjerenja izvan građevine (u otvorenom prostoru)

Mjerenja u otvorenom prostoru izvedena su sa zapadne strane Geotehničkog fakulteta uz samu zgradu fakulteta. (Slika 21.) Arduino sustav i Suricat 3D akcelerometar postavljeni su paralelno i horizontalno na tlo, a vibracije su izazvane udarcima maljem po betonskom zidu (dubina temeljenja 1,5 m) koji je prenosio vibracije u tlo. Primijećena je nešto manja korelacija među krivuljama nego u zatvorenom prostoru. Razlog tome je daleko veći šum prisutan kod Arduino platforme te dulji interval vibriranja. Maksimalne amplitude približno su jednake, iako u slučaju mjerenja u otvorenom prostoru, za razliku od unutarnjeg prostora, Arduino platforma ponekad prikazuje i veće vrijednosti. Općenito su iznosi ubrzanja nekoliko puta manji nego u zatvorenom prostoru, a razlog tome je sposobnost tla da prigušuje vibracije.

Prilikom transformacije dobivenih rezultata koristila se ista metoda kao i u prethodnom poglavlju. Rezultate je bilo nešto lakše preklopiti, budući da je interval snimanja Suricat akcelometrom bio 10 minuta, umjesto jedan sat.



Slika 20. Usporedba rezultata mjerenja vibracija u otvorenom prostoru pomoću Arduino sustava i Suricat 3D akcelometra

Slika 21. Lokacija mjerenja i postava sustava za mjerenje vibracija u otvorenom prostoru

5. Zaključak

Cilj ovog rada je bio dokazivanje potencijala primjene Arduino platforme za mjerenja u inženjerstvu okoliša. Nakon pripremljenih, izvedenih i analiziranih testnih mjerenja, može se zaključiti da rezultati ispitivanja kvalitetom znatno nadilaze očekivanja. Ustanovljeno je također da Arduino platforma predstavlja koncept koji u budućnosti može postati solidna i pouzdana alternativa etabiliranim mjernim uređajima, posebno za mjerenja na temelju kojih se trebaju donositi odluke o daljnjem razvoju projekata i/ili mjernih sustava u inženjerstvu okoliša.

S obzirom na višestruko manju cijenu kompletnog Arduino sustava za određena mjerenja, jasna je njegova prednost u odnosu na profesionalne uređaje. Međutim, za ispravno i pouzdano funkcioniranje Arduino sustava, potrebno je uložiti mnogo truda u učenje programiranja i njegovo razumijevanje. Kontinuiran i pouzdan rad cjelokupnog sustava može predstavljati problem, jer se nerijetko događa da se kontrolna pločica treba resetirati ili program ponovno pokrenuti. Dio rješenja vjerojatno leži u korištenju kvalitetnijih komponenti. Za potrebe ovog rada korištene su komponente iz Croduino seta za početnike koji predstavlja samo uvod ili prvi korak u cjelokupnu platformu.

Nakon višemjesečnog rada na Arduino platformi i istraživanja njenih mogućnosti te uspješno izvedenih ispitivanja, ustanovio sam da se isplati ulagati napore u razvoj i primjenu platforme. Platforma omogućava praktičan i interaktivan način učenja te omogućava brzo razumijevanje odabranih područja zanimanja/interesa, a to ju čini idealnom za primjenu na visokim učilištima. Uvjeren sam da će zbog svojih mogućnosti u bliskoj budućnosti Arduino platforma postati vrlo raširena i u polju inženjerstva okoliša.

Popis literature

1. *Arduino – Introduction.* Dostupno na: https://www.arduino.cc/en/guide/introduction. Datum pristupa: 12.4.2018.
2. History of Arduino. Dostupno na: http://creativityprojects.blogspot.hr/2013/03/history-of-arduino_4195.html. Datum pristupa: 13.4.2018.
3. The making of Arduino. Dostupno na: https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino. Datum pristupa: 13.4.2018.
4. How many Arduinos are „in the wild“?. Dostupno na: https://blog.adafruit.com/2011/05/15/how-many-arduinos-are-in-the-wild-about-300000/. Datum pristupa: 13.4.2018.
5. Arduino FAQ. Dostupno na: http://medea.mah.se/2013/04/arduino-faq/. Datum pristupa: 13.4.2018.
6. O e-radionici. Dostupno na: https://e-radionica.com/hr/about-us/. Datum pristupa: 13.4.2018.
7. Novi Croduino je tu – Croduino Basic2. Dostupno na: https://e-radionica.com/hr/blog/2015/10/11/novi-croduino-je-tu-croduino-basic2/. Datum pristupa: 15.4.2018.
8. Što je Arduino, a što Croduino?. Dostupno na: https://e-radionica.com/hr/blog/2015/10/08/sto-je-arduino-i-croduino/. Datum pristupa: 15.4.2018.
9. DHT22 Temperature-Humidity sensor. Dostupno na: https://www.adafruit.com/product/385. Datum pristupa 20.4.2018.
10. GitHub-adafruit/Adafruit_sensor_DHT22. Dostupno na: https://github.com/adafruit/Adafruit_Sensor. Datum pristupa: 21.4.2018.
11. GitHub-adafruit/DHT22_sensor_library. Dostupno na: https://github.com/adafruit/DHT-sensor-library. Datum pristupa: 21.4.2018.
12. vibracije. Dostupno na: http://www.enciklopedija.hr/Natuknica.aspx?ID=64462. Datum pristupa: 17.5.2018.
13. Vibracije mehaničkih sustava. Dostupno na: http://e.math.hr/vibracije/index.html. Datum pristupa: 17.5.2018.

14. Grubišić, R.: „Teorija konstrukcija“, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, Zagreb, 2002.
15. Šumanovac, F: „Osnove geofizičkih istraživanja“, Sveučilište u Zagrebu, Rudarsko-geološko-naftni fakultet, Zagreb, 2012.
16. Hip, I.: „Predavanja iz Fizike 2: Titranje i valovi“, Sveučilište u Zagrebu, Geotehnički fakultet, Varaždin, 2016.
17. Gazdek, M: „Predavanja iz geofizike“, Sveučilište u Zagrebu, Geotehnički fakultet, Varaždin 2016.
18. Chameau, J.L., Rix, G.J., Empie L. Measurment and Analysis of Civil Engineering Vibrations 1998. Missouri Institute of Science and Technology
19. 3-osni žiroskop + akcelerometar MPU6050. Dostupno na: https://e-radionica.com/hr/3-osni-ziroskop-acelerometar-mpu-6050.html. Datum pristupa: 13.6.2018.
20. MoHo s. rl. – Instruments for Geophysics and Civil Engineering. Dostupno na: http://moho.world/en/. Datum pristupa: 13.6.2018.
21. Arduino _MPU6050: libraries. Dostupo na: https://drive.google.com/file/d/0B7o8xcgWngX9ZVdOaFc2TzhXaTA/view. Datum pristupa: 13.6.2018.

Popis slika

1. Arduino Uno i Croduino Basic2 [1,5]

2. Komponente Croduino Basic pločice [7]

3. Arduino softver s osnovnim funkcijama *void setup* i *void loop*, temeljnim komponentama svakog programa

4. Povezivanje +5V i gnd pinova s eksperimentalnom pločicom [8]

5. Spoj digitalnog pina i otpornika na eksperimentalnoj pločici [8]

6. Povezivanje LED diode na eksperimentalnoj nožici [8]

7. Povezivanje prekidača s eksperimentalnom i kontrolnom pločicom [8]

8. Završen program za paljenje i gašenje LED diode pomoću tipke

9. Tipka nije pritisnuta - lampica ne svijetli, tipka je pritisnuta – lampica svijetli [8]

10. DHT22 senzor za mjerenje temperature i vlažnosti zraka

11. Spoj DHT22 senzora s Croduino Basic2 kontrolnom pločicom

12. PLX-DAQ softver u spoju s Microsoft Excelom

13. Postavljanje DHT22 senzora na vanjski dio prozora

14. Harmonijski oscilator na idealno glatkoj podlozi [16]

15. Graf kretanja mase na opruzi u vremenu [16]

16. Primarni, sekundarni, Loweov i Rayleighev val [17]

17. MPU6050 akcelerometar fiksiran u kućištu te spojen na Croduino Basic2 pločicu

18. 3D akcelerometar Suricat i Croduino Basic2 povezan s MPU6050 senzorom (unutar kućišta)

19. Praćenje rezultata snimanja, lijevo Suricat 3D akcelerometar (prati se gornja os x), desno Arduino platforma

20. Lokacija mjerenja i postava sustava za mjerenje vibracija u otvorenom prostoru

Prilozi

Prilog 1.

**Program za mjerenje temperature i vlage pomoću DHT22 senzora i Arduina uz prikaz podataka u Microsoft Excelu pomoću PLX-DAQ softvera*

```
// DHT Temperature & Humidity Sensor
// Unified Sensor Library Example
// Written by Tony DiCola for Adafruit Industries
// Released under an MIT license.

// Depends on the following Arduino libraries:
// - Adafruit Unified Sensor Library: https://github.com/adafruit/Adafruit_Sensor
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
int lampica = 8;
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN            2        // Pin which is connected to the DHT sensor.

// Uncomment the type of sensor in use:
//#define DHTTYPE           DHT11     // DHT 11
#define DHTTYPE           DHT22     // DHT 22 (AM2302)
//#define DHTTYPE           DHT21     // DHT 21 (AM2301)

// See guide for details on sensor wiring and usage:
//   https://learn.adafruit.com/dht/overview

DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Vrijeme,Temperature,Humidity,");
  Serial.println("RESETTIMER");
  // Initialize device.
  dht.begin();
  Serial.println("DHTxx Unified Sensor Example");
  // Print temperature sensor details.
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.println("Temperature");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println("
*C");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println("
*C");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println("
*C");
  Serial.println("------------------------------------");
  // Print humidity sensor details.
  dht.humidity().getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.println("Humidity");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value);
Serial.println("%");
```

```
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value);
Serial.println("%");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution);
Serial.println("%");
  Serial.println("------------------------------------");
  // Set delay between sensor readings based on sensor details.
  delayMS = sensor.min_delay / 300000;
}

void loop() {
  // Delay between measurements.
  delay(delayMS);
  // Get temperature event and print its value.
  sensors_event_t event;
  dht.temperature().getEvent(&event);
Serial.print("DATA,TIME,");
    Serial.print(event.temperature);

 Serial.print(", ");
  // Get humidity event and print its value.
 dht.humidity().getEvent(&event);

    Serial.println(event.relative_humidity);
 delay (3000);
}
```

Prilog 2.

**Program za očitavanje ubrzanja u smjeru x-osi akcelometra MPU6050 [21]*

```
// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
   #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

//resetiranje
//int ResetPin = 7;

// uncomment "OUTPUT_READABLE_WORLDACCEL" if you want to see acceleration
// components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no magnetometer
// is present in this case). Could be quite handy in some cases.
```

```
#define OUTPUT_READABLE_WORLDACCEL

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
```

```
//
================================================================
=======
        // ===              INTERRUPT DETECTION ROUTINE              ===
//
================================================================
=======


volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has
gone high
void dmpDataReady() {
   mpuInterrupt = true;
}


//
================================================================
=======
// ===                   INITIAL SETUP                    ===
//
================================================================
=======


void setup() {
   // join I2C bus (I2Cdev library doesn't do this automatically)
   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
      //Wire.setClock(4000000); // 400kHz I2C clock. Comment this line if
having compilation difficulties
   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
   #endif



   //resetiranje
  // digitalWrite(ResetPin, HIGH);
```

```
  // pinMode(ResetPin, OUTPUT);


   // initialize serial communication
   // (115200 chosen because it is required for Teapot Demo output, but it's
   // really up to you depending on your project)
   Serial.begin(115200);


   while (!Serial); // wait for Leonardo enumeration, others continue
immediately


   // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or
Arduino
   // Pro Mini running at 3.3V, cannot handle this baud rate reliably due to
   // the baud timing being too misaligned with processor ticks. You must use
   // 38400 or slower in these cases, or use some kind of external separate
   // crystal solution for the UART timer.


   // initialize device
   //Serial.println(F("Initializing I2C devices..."));
   mpu.initialize();
   pinMode(INTERRUPT_PIN, INPUT);


     // load and configure the DMP
   //Serial.println(F("Initializing DMP..."));
   devStatus = mpu.dmpInitialize();


   // supply your own gyro offsets here, scaled for min sensitivity
   mpu.setXAccelOffset(-914); // 1688 factory default for my test chip
   mpu.setYAccelOffset(-3972);
   mpu.setZAccelOffset(1492);
   mpu.setXGyroOffset(54);
   mpu.setYGyroOffset(-37);
   mpu.setZGyroOffset(-11);
```

```
  // za odredivanje osjetljivosti mjerenja
    mpu.setFullScaleAccelRange(0); //0 = +/- 2g | 1 = +/- 4g | 2 = +/- 8g | 3 =  +/- 16g



    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // turn on the DMP, now that it's ready
        //Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);


        // enable Arduino interrupt detection
        //Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();


        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        //Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;


        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
```

```
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}



// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================


void loop() {
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize) {
        // other program behavior stuff here
        // .
        // .
        // .
        // if you are really paranoid you can frequently test in between other
        // stuff to see if mpuInterrupt is true, and if so, "break;" from the
        // while() loop to immediately process the MPU data
        // .
        // .
        // .
    }
```

```
    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too
inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        //Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should happen
frequently)
    } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;



        #ifdef OUTPUT_READABLE_WORLDACCEL
            // display initial world-frame acceleration, adjusted to remove gravity
            // and rotated based on known orientation from quaternion
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetAccel(&aa, fifoBuffer);
```

```
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
            mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
            Serial.print("*+*");
            //Serial.print(aa.x);

            //Serial.print(aa.y);
            //Serial.print("*3*");
            Serial.print(aa.x);

            Serial.println("*-*");
            //Serial.println("*4*");
        #endif

 //       blinkState = !blinkState;
//       digitalWrite(LED_PIN, blinkState);
   }
   //resetiranje
  // if(Serial.available()>0){

  //   String dolaznaPoruka = Serial.readString();

    // if(dolaznaPoruka == "resetiraj"){

      //   digitalWrite(ResetPin, LOW);

        // delay(100);
     //}
   }
//}
```