Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE53Lovcevic

VIZUALIZACIJA RADA AKCELEROMETRA POMOĆU QT I ANDROID APLIKACIJE UPOTREBOM KLIJENT-SERVER ARHITEKTURE

ACCELEROMETER VISUALIZATION USING QT AND ANDROID APPLICATION AND CLIENT-SERVER ARCHITECTURE

Milan Lovčević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U okviru ovog rada prikazane su realizacije Qt i Android aplikacija koje omogućavaju prikupljanje podataka merenja akcelerometra ADXL345. Opisani su i način skladištenja podataka u SQLite bazu podataka i vizualizacija u realnom vremenu, primenom klijent-server arhitekture.*

Ključne reči: *Akcelerometar, Raspberry Pi, Qt, Android, klijent-server arhitektura*

Abstract – *Within this paper, the realizations of Qt and Android applications are presented, which enable the collection of measurement data of the ADXL345 accelerometer. The method of data storage in SQLite database and real-time visualization, using client-server architecture, are also described.*

Keywords: *Accelerometer, Raspberry Pi, Qt, Android, client-server architecture.*

1. UVOD

Jedna od reči koja se poslednjih par godina sve intenzivnije koristi u indrustriji jeste termin *IoT*, što predstavlja skraćenicu od „*Internet of Things*" ili kako se na našem jeziku prevodi Internet stvari ili Internet integrisanih uređaja.

Ovaj pojam se odnosi na mrežu fizičkih uređaja, vozila, zgrada i drugih predmete koji u sebi sadrže ugrađen softver, senzor i pristup internetu preko čega ti objekti mogu da prikupljaju i razmenjuju podatke. *IoT* često uključuje obradu velikih količina podataka prikupljenih radi dobijanja korisnih informacija, koje mogu dosta da pomognu prilikom donošenja nekih strateških odluka.

Na primer, u nekim zemljama su postavljeni merači količine padavina tokom cele godine, zatim se prikupljeni podaci analiziraju i koriste kako bi zemlja bolje reagovala prilikom velikih padavina ili poplava [1].

U daljem tekstu biće dat jedan od primera upotrebe *IoT* tehnologije, korišćenjem ADXL345 senzora, zatim TCP/IP protokola zasnovanim na krijent-server arhitekturi, zaduženim za razmenu podataka putem interneta, kao i Qt i *Android Studio* aplikacija, pomoću kojih je kreiran grafički korisnički interfejs.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Mezei, vanr. prof.

2. OPIS PODSISTEMA UREĐAJA

Komponenete koje su korišćene u realizaciji ovog master rada su Raspberry Pi 3 model B+ (skraćeno RPi) i ADXL345 model akcelerometra. *Android Studio* i *Qt Creator* su korišćeni za realizaciju server aplikacije vidljive korisniku i na kraju, za razmenu podataka između Raspberry Pi i jedne od server aplikacija korišćena je klijent-server arhitektura.

Akcelerometar je komponenta koja se koristi za merenje vibracija ili ubrzanja kretanja objekata. Svoju primenu je pronašao u mnogim granama industrije, kao što je automobilska industrija (npr. pri sudaru, za aktiviranje vazdušnih jastuka), robotika (npr. za određivanje pozicije objekta), u avio industriji, itd. Akcelerometar koji je korišćen za realizaciju ovog projekta je ADXL345, koji predstavlja akcelerometar male snage i visoke rezolucije (13-bita). Poseduje mogućnost merenja statičkog ubrzanja (gravitaciona sila), kao i dinamičkog ubrzanje, koje predstavlja rezultat kretanja ili vibriranja. Zbog svoje visoke rezolucije ima mogućnost merenja promene nagiba manjeg od 1.0° [2, 3].

Qt predstavlja besplatni softver za kreiranje grafičkog korisničkog interfejsa kao i višeplatformskih aplikacija koje rade na različitim softverskim i hardverskim platformama kao što je Windows, macOS, Android ili na embeded sistemima, sa malo ili bez izmena u osnovi koda i bez ograničavanja funkcionalnosti ili brzine aplikacije. Signali i slotovi predstavljaju jednu od najvažniji Qt karakteristika koji omogućavaju komunikaciju između Qt objekata [4, 5].

Android Studio predstavlja razvojno okruženje koje je namenjeno za razvoj Android mobilnih aplikacija, zasnovano je na *IntelliJ IDEA* softveru. Glavne komponente koje se koriste u razvoju Android aplikacije su aktivnosti i servisi [5, 6].

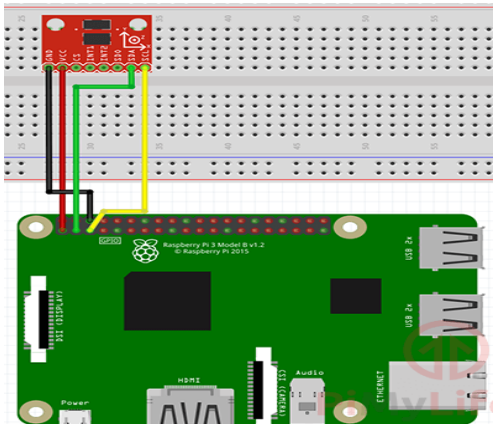
Klijent-server arhitektura predstavlja komunikaciju između dve aplikacije koje se nalaze na odvojenim uređajima . Aplikacija koja se zove klijent ima zadatak da inicira komunikaciju sa drugom aplikacijom koja se zove server. Dok server ne mora da zna nikakvu informaciju o klijentu, klijent mora da poseduje osnovne informacije o serveru, kao što je IP adresa i port na koji može da se poveže. Da bi se uspostavila komunikacija, potrebno je da server i klijent kreiraju sopstveni *socket*, nakon uspostavljanje komunikacije moguća je međusobna razmena podataka (*full duplex*). Velika prednost klijent-server arhitekture je što dozvoljava razmenu podataka sa

velike udaljenosti, pošto akcelerometar ne mora fizički biti povezan sa uređajem na kojem je server aplikacija implementirana, odnosno podaci se prenose putem interneta ili lokalne mreže [4].

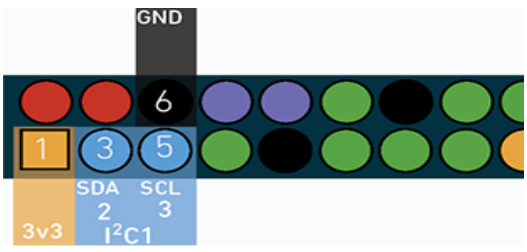
Kao što je bilo rečeno, implementacija projekta je zasnovana na klijent-server arhitekturi koja razdvaja rad na dve celine. Prvi deo predstavlja povezivanje Raspberry Pi mikroračunara sa odgovarajućim akcelereometrom i kreiranje aplikacije u Python-u, koja će predstavljati klijenta. Drugi deo predstavlja kreiranje Qt i Android server aplikacije, čija će funkcija biti da nakon što klijent inicira komunikaciju počne sa prikupljanjem podataka poslatih od klijenta, zatim da ih vizualizuje (grafički prikaz) i skladišti u bazu podataka. Za skladištenje podataka u bazu korišćena je *SQLite* biblioteka.

3. REALIZACIJA RPI KLIJENT APLIKACIJE

Klijent aplikacija je realizovan u Python-u i nalazi se na Raspberry Pi uredjaju koji je direktno povezan sa akcelerometrom pomoću GPIO pinova, kao što je prikazano na slici 1. Komunikacija između RPi i akcelerometra je sihrona serijska komunikacija zasnovana na I2C (eng. *Inter-Integrated Circuit*) serijskom interfejsu, koji predstavlja jedan od najčešće korišćenih načina komunikacije između uređaja na kratkim razdaljinama (rastojanje treba da bude manje od 3 metra). Da bi se ostvario ovaj vid komunikacije potrebne su nam pored napona i uzemljenja samo još dve žice, čiji položaj na RPi ploči je dat na slici 2.



Slika 1. *RPI i akcelerometar povezivanje [7]*



Slika 2. *Prikaz korišćenih pinova [7]*

Kako bi klijent uspeo da ostvari komunikaciju sa serverom potrebno je da zna IP adresu servera i port na koji može da se poveže. U Python skripti to je definisano upotrebom dve promenjive „HOST“ i „PORT“, gde se promenjivoj „HOST“ dodeljuje IP adresa servera, a promenjivoj „PORT” broj porta servera na koji klijent planira da se poveže. Sa druge strane, potrebno je takođe pripremiti komunikaciju sa akcelerometrom, za tu svrhu koristimo „*busio*” i „*board*” biblioteke. Prva biblioteka u sebi sadrži mnoštvo drugih biblioteka koje omogućavaju rukovanje sa različitim serijskim protokolima. Prilikom realizacije ovog projketa ona je iskorišćena kako bi se ostvarila komunikacija sa akcelerometrom putem I2C serijskog interfejsa. Druga biblioteka je potrebna kako bi se na lakši način odredili položaj SDA i SCL pinova na RPi ploči.

Instanciranje različitih klasa, priprema pinova, definisanje porta i IP adrese je potrebno uraditi samo jednom, na početku izvršavanja programa. Nakon uspešnog povezivanja sa serverom, klijent prvo čita podatke sa senzora, pakuje ih i šalje serveru. Ovaj proces je postavljen u beskonačnu petlju, sa pauzom od jedne sekunde između slanja dva uzastopna podatka. Da bismo prestali sa slanjem podataka potrebno je prekinuti konekciju sa serverom.

4. IMPLEMENTACIJA BAZE PODATAKA

Baza koja je zadužena za skladištenje podataka primljenih od klijenta je realizovana upotrebom aplikacionog formata podataka koji se zove *SQlite*. Identična implementacija baze podataka, upotrebom *SQlite* biblioteke je urađena za potrebe obe server aplikacije. Podaci koji se čuvaju u bazi su *x*, *y* i *z* kordinate, koje server dobija do klijenta, kao i vremenski trenutak i datum kada je ta vrednost zabeležena. Primer baze podataka je dat na slici 3.

X_POSITION	Y_POSITION	Z_POSITION	DATE	TIME
5.0602314	7.1784677999...	5.766310199...	09.08.2020	10:56:18
5.0210048	7.1784677999...	5.6486304	09.08.2020	10:56:19
5.0602314	7.1784677999...	5.6094038	09.08.2020	10:56:20
5.0210048	7.2176943999...	5.687856999...	09.08.2020	10:56:21
5.0602314	7.1392411999...	5.727083599...	09.08.2020	10:56:22

Slika 3. *Baza podataka*

Da bi se kreirala baza potrebno je navesti putanju gde će se ta baza čuvati, tip baze podataka, naziv tabele i nazive kolona koje treba da se nalaze u tabeli. Baza podataka se kreira samo u slučaju ako na navedenoj putanji ne postoji baza sa identičnim imenom, ukoliko postoji, proces kreiranja se preskače. Prilikom izlaska iz aplikacije sama baza podataka kao i podaci koji se nalaze u njoj se ne brišu, tako da prilikom narednog pokretanja aplikacije, proces kreiranja baze podataka se preskače i nastavlje se upis podataka u tabelu bez gubitka prethodno unetih podataka.

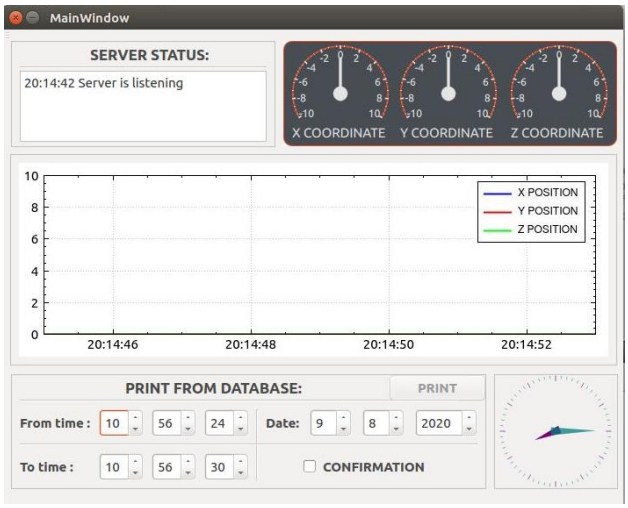
5. REALIZACIJA QT SERVER APLIKACIJE

Server koji je realizovan za potrebe Qt aplikacije pripada onom delu funkcionalnih blokova čije se izvršavanje dešava u pozadini aplikacije, odnosno nije vidljivo korisniku. Jedine informacije koje korisnik može da dobije u vezi servera su prikazane u polju za ispis teksa (eng. *textbox*), koji se nalazi u gornjem levom uglu aplikacije i to su samo osnovne informacije o trenutnom

statusu servera i klijenta. U polju za ispis teksta moguće je dobiti tri različite poruke, uz sve tri poruke ispisuje se i vreme kada se određena akcija desila:

- *„Server is listening“*, koja označava da je server aktivan i da čeka na klijenta.
- *„Client connected“*, klijent se konektovao i počinje sa slanjem podataka.
- *„Client disconnected“*, komunikacija između servera i klijenta je prekinuta.

Qt sever je realizovan upotrebom *QTcpServer* klase, koja obezbeđuje osnovne TCP server funkcionalnosti. Prilikom pokretanja aplikacije server se automatski aktivira i čeka da se neki klijent poveže, što takođe potvrđuje i poruka u polju za ispis teksta. Slika 4 predstavlja izgled aplikacije realizovanog u Qt-u.



Slika 4. *Izgled Qt aplikacije*

Vizualizacija rada akceleromtera u realnom vremenu je postignuta upotrebom *„QcustomPlot”* biblioteke. Glavni deo programa zauzima grafik koji služi za vizuelni prikaz podataka u realnom vremenu koje je server primio od klijenta. Grafik se sastoji od dve ose *x* i *y*, *y* osa je iskorišćena za ispis vremenskog trenutka, u 24-oro časovnom režimu, u kojem su pristigle vrednosti zabeležene, dok *x* osa se koristi za prikaz vrednosti koje je klijent poslao. Početni opseg x ose je od 0 do 10 ali on može dinamički da se menja. Npr. ako klijent pošalje podatak čija je vrednost manja od 0 ili veća od 10, grafik će proširiti svoj opseg kako bi mogao da prikaže i tu vrednost.

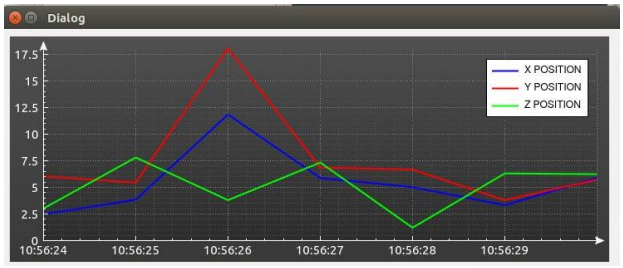
Kazaljke u gornjem desnog uglu su realizovane upotrebom QML jezika. QML je deklarativni jezik dizajniran da opiše korisnički interfejs programa, odnosno kako korisnički interfejs treba da izgleda i kako treba da se ponaša.

Pošto klijent svake sekunde šalje po tri podatka serveru, u gornjem desnom uglu programa nalaze se tri kazaljke, gde svaka od njih treba da prikaže po jednu od te tri vrednosti.

Opseg sve tri kazaljke je od -10 do 10, i ne može dinamički da se menja, odnosno ako klijent pošalje vrednosti koja je veća od 10, kazaljka će pokazati na vrednost 10 ili u slučaju da je potrebno prikazati vrednosti manju od -10 kazaljka će biti postavljena na vrednosti -10.

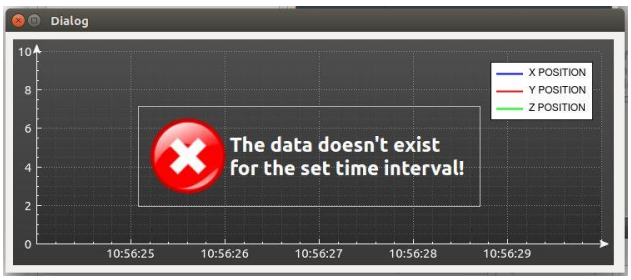
U donjem desnom uglu aplikacije nalazi se sat, koji je preuzet iz primera koji se dobijaju uz Qt instalaciju. Njegov doprinos ovoj aplikaciji je samo estecki.

Pošto se podaci koje je klijent poslao čuvaju u bazu podataka u tabelarnoj formi, koja često može da bude prilično naporna prilikom analize merenja akcelerometra, u samu aplikaciju ugrađen je i mehanizam koji nam omogućuva da podatke pročitamo iz baze za neki zadati vremenski interval i da ih iscrtamo na grafiku, kako bismo olakšali proces analize. Mogućnost iscrtavanja podataka iz baze je ostvarena pomoću opcija koje zauzimaju donji deo aplikacije. Kako bismo grafički prikazali podatke iz baze podataka za neki željeni interval vremena prvo je potrebno da unesemo početno i krajnje vreme, odnosno od kojeg do kojeg trenutka želimo da prikažemo podatke i datum. Nakon što se unese željeni vremenski interval i označi polje za potvrdu (eng. *checkbox)* dobija se prozor kao na slici 5.



Slika 5. *Grafički prikaz podataka za zadati vremenski interval*

Postoji i druga mogućnost, da je korisnik pravilno uneo vremenski interval ali da za taj interval podaci ne postoje u bazi podataka, u tom slučaju otvoriće se novi prozor sa grafikom ali umesto iscrtanih podataka biće ispisana poruka da podaci ne postoje za zadati vremenski interval, kao što je prikazano na slici 6.

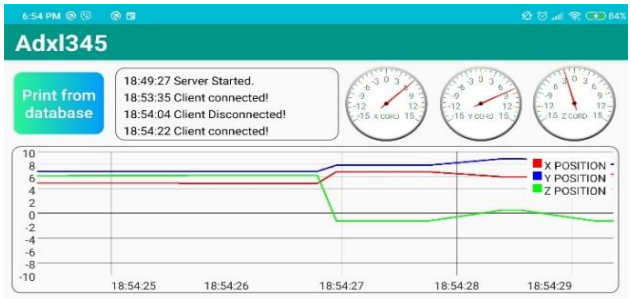


Slika 6. *Poruka koja se ispisuje kada podaci ne postoje za zadati vremenski interval*

6. REALIZACIJA ANDROID APLIKACIJE

Kao što je bio slučaj i kod prethodno objašnjene server aplikacije i ova aplikacija ima polje za ispis teksta u kojem se ispisuju samo osnovne informacija o trenutnom stanju klijenta i servera. Prilikom pokretanja aplikacije server se automatski aktivira i čeka da se neki klijent poveže, što takođe potvrđuje i poruka u polju za ispis teksta. Pošto postoji mogućnost da se poruke u polju za ispis teksa nagomilaju i da neke budu slučajno prepisane, kako bi se izbeglo to i kako bi korisnik u svakom trenutku imao pristup svim informacijama koje su bile ispisane u polju od trenutka pokretanja aplikacije, ugrađen je mehanizam pomeranja na gore i dole (eng. *scroll up and*

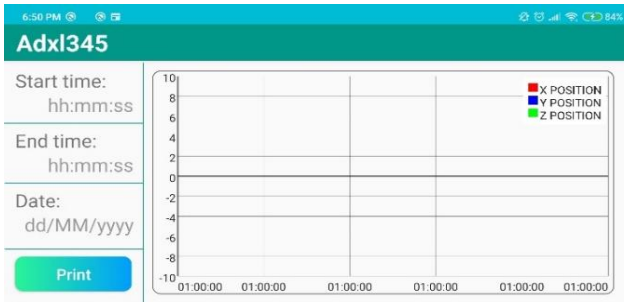
down) kroz polje. Na slici 7 je prikazan izgled server aplikacije realizovane u *Android Studio* programu.



Slika 7. *Izgled Android aplikacije*

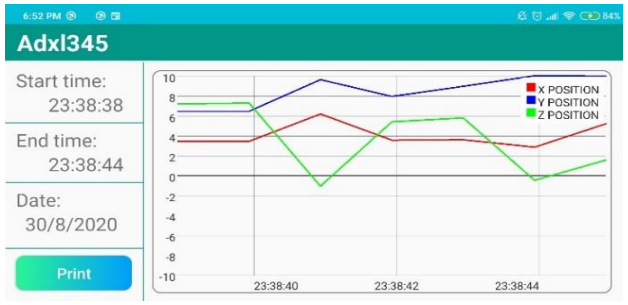
Najveći deo aplikacije zauzima grafik koji u realnom vremenu ispisuje podatke dobijene od klijenta. Za razliku od Qt aplikacije, početni opseg ovog grafika je od -10 do 10 i može dinamički da se proširi ako postoji potreba za tim. Gornji deo prozora aplikacije uglavnom ima istu funkciju kao i u aplikaciji realizovanoj u Qt-u. Tri kazaljke prikazuju vrednosti dobijene od klijenta za sve tri kordinate, a polje za ispis teksta ima zadatak da nas informiše o trenutnom statusu servera i klijenta.

Najveća razlika između ove aplikacije i one realizovane u Qt-u je u dugmetu „*Print from database*" koje kada se pritisne otvara identičan prozor kao na slici 8.



Slika 8. *Prozor za ispis podataka iz baze podataka*

Kako bi se iscrtali podaci za željeni interval vremena potrebno je uneti početno, krajnje vreme i datum, odnosno potrebno je pritisnuti na svako od tri polja koja se nalaze na levoj strani prozora. Nakon što su svi neophodni podaci uneseni potrebno je pritisnuti dugme „*Print*". Ukoliko je korisnik pravilno uneo parametre i podaci postoje u bazi za zadati interval vremena, oni će se iscrtati na grafiku kao što je prikazano na slici 9. U slučaju da podaci ne postoje u bazi grafik će ostati prazan.



Slika 9. *Ispis podataka za zadati vremenski interval*

Najveća prednosti ove aplikacije u odnosu na aplikaciju realizovanu u Qt-u je mogućnost zumiranja (eng. *Zoom*) i pomeranja levo i desno kroz grafik, što u velikoj meri olakšava analizu podataka za veći vremenski interval.

7. ZAKLJUČAK

U prethodnim poglavljima objašnjena je jedna od mogućih upotreba *IoT* tehnologije, koja ima za zadatak da vrednosti merenja senzora vizualizuje i skladišti u bazu podataka. Za prikupljanje podataka iskorišćen je ADXL345 model akcelerometra koji je povezan sa Raspberry Pi mikroračunarom, koji zajedno predstavljaju jednu stranu prilikom razmene podataka. Drugu stranu čini uređaj koji poseduje instaliranu Android ili Qt server aplikaciju, koje su realizovane upotrebom C++, Java, QML i XML programskog jezika.

Komunikacija između mikroračunara i aplikacija postignuta je upotrebom TCP/IP protokola koji je zasnovan na klijent-server arhitekturi. Mogućnosti za proširenje opsega funkcionalnosti i unapređenja projekta su velike. Projekat dalje može da se razvija u smeru upotrebe više od jednog klijenta istovremeno ili upotrebom različitih senzora.

8. LITERATURA

[1] W.-M. Lee, „Introduction to IoT Using the Raspberry Pi", Code magazine. https://www.codemag.com/article/1607071/Introduction-to-IoT-Using-the-Raspberry-Pi (pristupljeno u oktobru 2020.)

[2] Analog Devices, „Digital Accelerometer". https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf (pristupljeno u avgustu 2020.)

[3] Automatika, „Akcelerometri". https://www.automatika.rs/baza-znanja/senzori/akcelerometri.html (pristupljeno u avgustu 2020.)

[4] I. Mezei, „Računarska elektronika (deo skripti)", Fakultet tehničkih nauka, Novi Sad, 2018. https://www.elektronika.ftn.uns.ac.rs/ (pristupljeno u septembru 2020.)

[5] https://www.rt-rk.uns.ac.rs/sites/default/files/materijali/lab/Vezba02%20-%20Upoznavanje%20sa%20Android%20Studio%20okru%C5%BEenjem.pdf (pristupljeno u avgustu 2020.)

[6] http://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/MOS%20Predavanje%205%202018.pdf (pristupljeno u julu 2020.)

[7] Emmet, „Raspberry Pi Accelerometer using the ADXL345", Pi My Life Up, 24 May 2019. https://pimylifeup.com/raspberry-pi-accelerometer-adxl345/ (pristupljeno u septembru 2020.)

Kratka biografija:

Milan Lovčević rođen je u Sremskoj Mitrovici 1995. god. Završio je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, na Departmanu za energetiku, elektroniku i telekomunikacije, 2019 godine. Master rad iz oblasti Embeded sistemi i algoritmi odbranio je 2020.godine.