Automatizacija u industriji pića: Razvoj softvera za automatske punilice pića

Zdunić, Marin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I
INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Marin Zdunić

Automatizacija u industriji pića: Razvoj softvera za automatske punilice pića

ZAVRŠNI RAD

Osijek, 2024. godina

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Marin Zdunić
Studij, smjer:	Stručni prijediplomski studij Elektrotehnika, smjer Automatika
Mat. br. pristupnika, god.	A4720, 27.07.2021.
JMBAG:	0165089922
Mentor:	prof. dr. sc. Drago Žagar
Sumentor:	Josip Spišić, univ. mag. ing. comp.
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Grgić
Član Povjerenstva 1:	prof. dr. sc. Drago Žagar
Član Povjerenstva 2:	izv. prof. dr. sc. Višnja Križanović
Naslov završnog rada:	Automatizacija u industriji pića: Razvoj softvera za automatske punilice pića
Znanstvena grana završnog rada:	**Automatizacija i robotika (zn. polje elektrotehnika)**
Zadatak završnog rada:	Sumentor: Josip Spišić, mag.ing.comp. Zadatak rada je istražiti razvoj naprednog softverskog rješenja za automatizirane sustave punjenja pića. U okviru rada potrebno je analizirati rješenje i razviti programsko rješenje u Arduino IDE sučelju koji upravlja i optimizira proces punjenja pića, uključujući integraciju sa senzorima i kontrolnim jedinicama. Cilj je postići veću efikasnost, točnost i brzinu u proizvodnom procesu. Upravljanje automatskom punilicom treba riješiti putem mobilne aplikacije.

Datum ocjene pismenog dijela završnog rada od strane mentora:	15.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	30.09.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Marin Zdunić
Studij:	Stručni prijediplomski studij Elektrotehnika, smjer Automatika
Mat. br. Pristupnika, godina upisa:	A4720, 27.07.2021.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatizacija u industriji pića: Razvoj softvera za automatske punilice pića**

izrađen pod vodstvom mentora prof. dr. sc. Drago Žagar

i sumentora Josip Spišić, univ. mag. ing. comp.

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Punjenje se definira kao metoda u kojoj je tekućina pakirana u bocu, ta tekućina može biti voda ili druga pića [1]. Proces punjenja moguće je automatizirati pomoć PLC-a (engl. *Programmable Logic Controller*) ili, u ovom slučaju, mikrokontrolera. Od ovakvih procesa se u industriji očekuje velika brzina i točnost. Često su se ova dva zahtjeva kosila jedan s drugim, ali s napretkom tehnologije, to je sve rjeđe slučaj.

U ovo vrijeme, kada je industrija nikad moćnija i naprednija, svakodnevno se puni veliki broj boca u tisućama tvornica diljem svijeta. To je postala svakodnevica koju mnogi potrošači ni ne primjećuju, štoviše zahtjevi potrošača su sve veći. Kako bi se ispunili ti zahtjevi, industrija teži većoj i naprednijoj automatizaciji u takvim procesima, a to podrazumijeva veću brzinu i preciznost punjenja, izdržljivost svih komponenti procesa u nepovoljnim radnim uvjetima te razmjenjivanje informacija između uređaja koji sudjeluju u procesu radi povećane efikasnosti.

U radu je, na što jednostavniji i napredniji način, razvijeno softversko rješenje za sustave automatiziranog punjenja upravljane mobilnom aplikacijom. Za razvoj softvera koristilo se Arduino IDE sučelje, a za izradu mobilne aplikacije Blynk IoT platforma. Pri radu sustava testirana je potrošnja istoga kako bi se dobio uvid u efikasnost komponenti i samoga softvera.

1.1. Zadatak završnog rada

Zadatak rada je istražiti razvoj naprednog softverskog rješenja za automatizirane sustave punjenja pića. U okviru rada potrebno je analizirati rješenje i razviti programsko rješenje u Arduino IDE sučelju koji upravlja i optimizira proces punjenja pića, uključujući integraciju sa senzorima i kontrolnim jedinicama. Cilj je postići veću efikasnost, točnost i brzinu u proizvodnom procesu. Upravljanje automatskom punilicom treba riješiti putem mobilne aplikacije.

Punilica se sastoji od četiri aktuatora – koračnog motora te tri podvodne pumpe, tri ultrazvučna senzora za mjerenje razine u spremnicima tekućine, tri senzora protoka, jednog infracrvenog senzora za detekciju nadolazeće boce, jednog *displaya* za praćenje stanja procesa te *encodera* za odabiranje sučelja na *displayu*. Primanjem određenih podataka i naredbi s aplikacije, mikrokontroler upravlja aktuatorima kao što su pumpe i koračni motor. Svi senzori šalju informaciju o trenutnom stanju dijela procesa za koji su namijenjeni te su neophodni za uspješno obavljanje zadaće punjenja.

U radu će se objasniti značenje IoT-a (Intrnet of Things), cilj ovoga rada, Arduino IDE softverska platforma te izrada samog softvera punilice.

Cilj ovog rada jest prikazati razvoj i optimizaciju softvera za automatizirane punilice pića koje su upravljane mikrokontrolerima.

2. TEORIJSKA PODLOGA RJEŠENJA

U današnjoj automatiziranoj industriji češće se koriste PLC-ovi u odnosu na mikrokontrolere zbog svoje jednostavnosti programiranja, izdržljivosti u nepovoljnim uvjetima i brzini. U ovom slučaju, za jednostavniju automatiziranu punilicu, cjenovno je pristupačniji mikrokontroler, ali i kompleksniji za programiranje. Osvrnut ćemo se na softverske probleme koji se pojavljuju pri automatizaciji i kako ih ukloniti ili ublažiti.

2.1. Teorijski osvrt na problem i rješenje projekta

Miješanjem različitih ulaznih tekućina dobivamo jednu izlaznu tekućinu koja se sastoji od odabranog omjera ulaznih tekućina. U industriji su to pomno izračunati omjeri koji ovise o određenoj recepturi kako bi se dobila željena izlazna tekućina.

U ovom prototipu automatske punilice operater ima mogućnost doziranja tri tekućine u predodređenom omjeru i unošenja željenog broja boca koje se trebaju napuniti. Nakon unošenja traženih vrijednosti u aplikaciju, podaci se šalju u mikrokontroler putem Wi-Fi-a i u njemu ostaju spremljeni do kraja ciklusa ili sve dok ih operater ne promijeni. Od punilice se zahtjeva optimizirano vrijeme punjenja te velika preciznost iz razloga što boce koje se pune, volumena od 100 ml, imaju uska grla. S obzirom da su periferni sklopovi hardverski povezani s mikrokontrolerom, latencija je minimalna, odnosno prijenos signala između mikrokontrolera, senzora i aktuatora se odvija uz minimalno vrijeme kašnjenja. Današnji hardver radi na visokim frekvencijama, stoga ne predstavlja ograničenje brzom prijenosu signala. Iz tog razloga naglasak je na razvoju i optimizaciji softvera.

Cilj je na što jednostavniji način izraditi softver koji će u dovoljno kratkom vremenu zaustaviti bocu, koja je spremna za punjenje, na predviđeno mjesto te to ponoviti što više puta s minimalnom greškom zaustavljanja. Nakon što je boca zaustavljena, započinje proces punjenja prema omjeru zadanom u aplikaciji, u ovom slučaju izrazito je važno da konačan omjer u boci, nakon punjenja, bude što reprezentativniji onome u aplikaciji kako boca ne bi bila djelomično napunjena, s krivim omjerom ili u najgorem slučaju, presipana. Za vrijeme procesa punjenja u određenim vremenskim trenucima senzori razine spremnika šalju mikrokontroleru informaciju o razini tekućine u spremniku, stoga, kada se tekućina nalazi na kritičnoj razini, potrebno je zaustaviti proces kako ne

bi došlo do oštećenja pumpe u spremniku gdje je detektirana kritična razina te obavijestiti operatera o tome.

2.2. Primjena IoT tehnologije u automatizaciji

IoT (Intrnet of Things) ili internet stvari mreža je međusobno povezanih uređaja koji se povezuju i razmjenjuju podatke s drugim IoT uređajima i cloud poslužiteljem [2]. U ovakve uređaje najčešće je ugrađena tehnologija kao što su senzori i softver. Sve veći broj organizacija u raznim industrijama koriste IoT za učinkovitije poslovanje i napredovanje. Također, ova tehnologija je sve prisutnija i u domaćinstvima.

IoT sustav se sastoji od pametnih uređaja koji imaju mogućnost povezivanja na internet te imaju ugrađene podsustave kao što su procesori, senzori i komunikacijski hardver za prikupljanje, slanje i djelovanje nad podacima koje primaju iz svoje okoline. Obrada podataka moguća je na lokalnoj razini (engl. *edge computing*) ili u oblaku (engl. *cloud computing*). Obrada podataka na lokalnoj razini smanjuje količinu podataka poslanih u oblak, što ubrzava samu obradu podataka [3].

Osnovna ideja ove tehnologije je omogućiti jednostavnije povezivanje i umrežavanje uređaja u svrhu automatizacije željenih procesa bez potrebe za žičanim vezama, što u konačnici ima za rezultat manje troškove izgradnje infrastrukture te manje troškove rada [4].

Nedostatak ove tehnologije, s industrijskog stajališta, jest sigurnost i privatnost podataka s kojima IoT raspolaže. Ti podaci su od iznimne važnosti organizacijama koje koriste IoT te se isti ne smiju zloupotrijebiti ili koristiti bez dopuštenja organizacije. Na ovakvim mrežama može biti povezan velik broj različitih uređaja što omogućava jednostavniju krađu podataka koja organizacije može trajno oštetiti. Kako bi se ovakvi slučajevi izbjegli u industriji se koristi inačica IoT-a pod nazivom IIoT (engl. *Industrial Intrnet of Things*) koji ima gotovo identičnu ideju automatiziranja kao i IoT, ali je prilagođen industriji i ima veću sigurnost i privatnost podataka.

2.3. Usporedba primjene PLC-a i mikrokontrolera

Kao što je i ranije rečeno, ova dva tipa uređaja se koriste u svrhu automatizacije i kontrole procesa. Iako imaju istu svrhu, postoje značajnije razlike u njihovoj uporabi, rukovanju, načinu programiranja i rada.

PLC je isključivo namijenjen za industrijske procese kao što su kontrola proizvodnih linija, HVAC (engl. *heating, ventilation and air conditioning*) sustava i ostalih strojeva. Robustan je, odnosno, otporan na nepovoljne radne uvjete koji podrazumijevaju visoku temperaturu u postrojenju, vlagu, prašinu i elektromagnetne smetnje. Postoji nekoliko softvera u kojima je moguće razvijati programe za automatizaciju procesa pomoću PLC-a kao što su *Ladder Logic*, *Function Block Diagram* i *Structured Text* koji znatno pojednostavljuju kompleksno programiranje. Vrlo je fleksibilan iz razloga što omogućava jednostavno proširivanje s dodatnim I/O (engl. *Input/Output*) modulima. Također, podržava više industrijskih komunikacijkih protokola kao što su Profibus i Ethernet/IP, dok noviji PLC-ovi podržavaju i Profinet koji je nadogradnja Profibusa i omogućuje brzu i sigurnu komunikaciju između PLC-ova, senzora i aktuatora.

Mikrokontroler je dizajniran za širok spektar primjena koje variraju od osobnih uporaba, gdje se ne zahtjeva brzina, sigurnost i preciznost, sve do sustava u automobilima gdje se od njih očekuje upravo suprotno. Sami mikrokontroler nema visoki stupanj zaštite od nepovoljnih vanjskih utjecaja, ali izgradnjom kućišta specijaliziranih namjena moguće ih je zaštiti. S obzirom na PLC, posjeduje manji broj I/O kanala, no u većini slučajeva moguće je povezati više mikrokontrolera za obavljanje određenog procesa. Moguće ih je programirati u raznim programskim jezicima, neki od njih su C i C++, koji su znatno kompleksniji od onih za programiranje PLC-a. Isto tako podržavaju razna programska okruženja kao što su Arduino IDE, MPLAB X IDE, STM32CubeIDE i MicroPython. Vrlo su fleksibilni za razne sustave, ali za kompleksnije zadaće u većini slučajeva potrebne su im komponente za proširivanje. Najčešći protokoli za komunikaciju koje mikrokontroleri posjeduju su UART, SPI i I2C.

2.4. Osnovni protokoli za primjenu u jednostavnoj automatizaciji

Protokol je skup pravila za formatiranje i obradu podataka te je poput zajedničkog jezika namijenjenog uređajima. Uređaji unutar mreže mogu koristiti različite softvere i hardvere, međutim, korištenje protokola omogućuje im međusobno komuniciranje neovisno o tome.
U današnje vrijeme postoji mnogo protokola koji se mogu koristiti za povezivanje IoT uređaja na mrežu. Tip protokola koji se koristi ovisi o samom uređaju, njegovoj funkciji i korisnicima. Najčešće, udaljenost koju podaci moraju prijeći, bilo kratkog ili dugog dometa, određuje tip protokola.

Mreže male snage i kratkog dometa prikladne su za domaćinstva, urede i druga manja okruženja [5]. Ne zahtijevaju veliku količinu snage i jeftine su za održavanje. U ove mreže se svrstavaju Wi-Fi/802.11, bluetooth, NFC, Z-Wave i Zigbee. Prednost bluetootha je brzi prijenos signala u krugu do 10 metara, NFC (engl. *Near-field communication*) omogućuje razmjenu podataka između dva elektronička uređaja na udaljenosti do 4 cm te nudi niske brzine prijenosa podataka, Z-Wave je mreža koja koristi radiovalove niske energije za komunikaciju među uređajima te Zigbee koji je skup komunikacijskih protokola visoke razine koji se koriste za stvaranje osobnih mreža s malim digitalnim radijusom male snage.

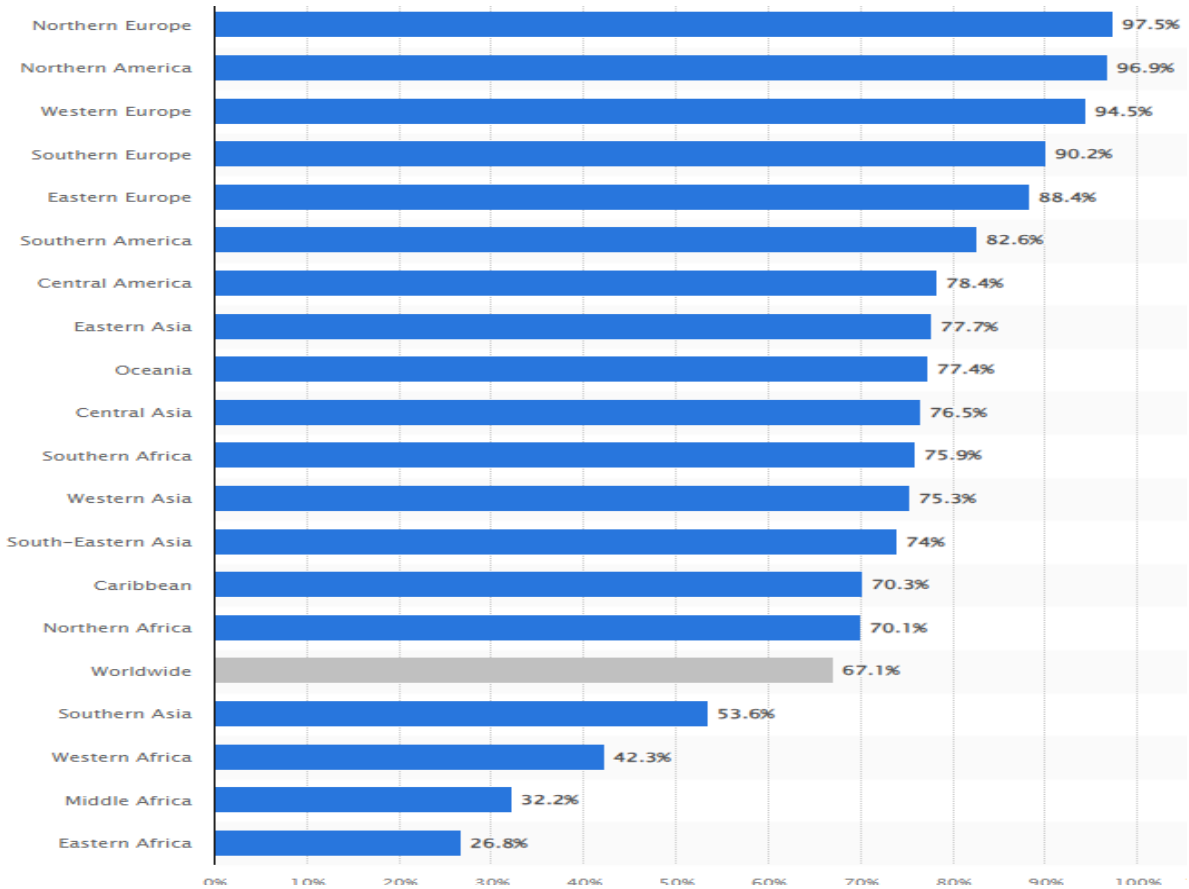
LPWAN-ovi (engl. *Low-power, wide-are networks*) omogućuju komunikaciju na udaljenosti od najmanje 500 metara, također zahtijevaju minimalnu snagu i koriste se za veliki broj IoT uređaja. Pod ove mreže spadaju 4G LTEE IoT, 5G IoT, Cat-0, Cat-1, LoRaWAN, LTE Cat-M1, NB-IoT/Cat-M2 i Sigfox.

2.4.1. Wi-Fi

Wi-Fi je tehnologija bežičnog umrežavanja koja elektroničkim uređajima poput računala, mobilnih uređaja i ostalim uređajima omogućuje međusobno povezivanje i povezivanje s internetom [6]. U današnje vrijeme ova tehnologija je prisutna svugdje u našoj okolini te se gotovo svaka osoba barem jednom povezala sa svojim uređajem na nju. Stoga, Wi-Fi je postao sastavni dio svakodnevnog života.

Wi-Fi koristi elektromagnetne valove, odnosno, radiovalove te funkcionira na nekoliko frekvencijskih opsega, a najčešći su oni od 2.4 GHz i 5 GHz prema IEEE 802.11 standardu. Frekvencijski opseg od 2.4 GHz je najrasprostranjeniji, dok od 5 GHz pruža veće brzine, ali i manji domet.

Na slici 2.1 prikazana je globalna stopa rasprostranjenosti interneta u srpnju 2024 godine, pri čemu je ista iznosila približno 67,1%. Primjećuje se da je rasprostranjenost u Europi visoko iznad prosjeka.



Slika 2.1 Globalna rasprostranjenost interneta u srpnju 2024.godine [7]

2.5. Softversko okruženje

Okruženje se odnosi na skup hardverskih i softverskih alata koje programer sustava koristi za izgradnju softverskih sustava [8].

Softversko okruženje služi za razvoj, testiranje i održavanje koda u mikrokontrolerima. Sastoji se od više raznih podsustava koji to omogućuju. Najčešće su to uređivač koda, compiler ili interpreter i program za ispravljanje pogrešaka (engl. *debugger*), kojima se pristupa putem grafičkog korisničkog sučelja. Programer piše i uređuje izvorni kod u uređivaču koda, a prevoditelj prevodi izvorni kod u jezik čitljiv računalu. Komponenta programa za ispravljanje pogrešaka pruža programeru mogućnost pregledavanja memorije i varijabli, pokretanje programa do sljedeće točke (engl. *breakpoint*), izvršavanje sljedeće linije koda i u nekim slučajevima, promjene vrijednosti varijabli ili čak promjene sadržaja linije koda koja se izvršava. Neka okruženja imaju ugrađene podsustave za simuliranje rada mikrokontrolera na temelju napisanoga koda, to omogućuje programerima testiranje koda i praćenje ponašanja mikrokontrolera bez potrebe povezivanja na isti.

Mikrokontroleri imaju ugrađen softver za učitavanje *firmwarea* putem USB-a ili nekog drugog komunikacijskog modula, koji se naziva *bootloader*. Ključan je u inicijalizaciji hardvera i pokretanju sustava u mikrokontrolerima.

2.5.1. Arduino IDE

Arduino IDE (engl. *Integrated Development Environment*) je napredno softversko okruženje koje programerima omogućuje učinkoviti razvoj softverskog koda namijenjen mikrokontrolerima te povećava produktivnost programera kombinirajući mogućnosti kao što su uređivanje i izgradnja softvera, testiranje i otklanjanje pogrešaka.

Uređivač koda ima mogućnost predviđanja funkcija i njihovo automatsko ispisivanje koje ubrzava proces pisanja samoga koda te omogućuje postojanje više od jedne datoteke u projektu. Kod pisan u Arduino programskom jeziku baziran je na C i C++ programskim jezicima. Komunikacija s mikrokontrolerima omogućena je serijskim putem pomoću USB-a, a kako bi se izbjegli dodatni moduli, komunikacija se odvija i s bootloaderom na mikrokontrolerima koji omogućavaju *upload*, odnosno, učitavanje koda na iste. Serijski monitor omogućuje programerima nadziranje i kontinuirano praćenje podataka koji se razmjenjuju između mikrokontrolera u stvarnom vremenu, isto tako ovim putem moguće je prosljeđivati podatke, također u stvarnom vremenu. Ovo programsko okruženje, osim Arduino pločica, podržava i široki raspon ostalih pločica kao što su ESP32, ESP8266, Teensy modeli i ostale.

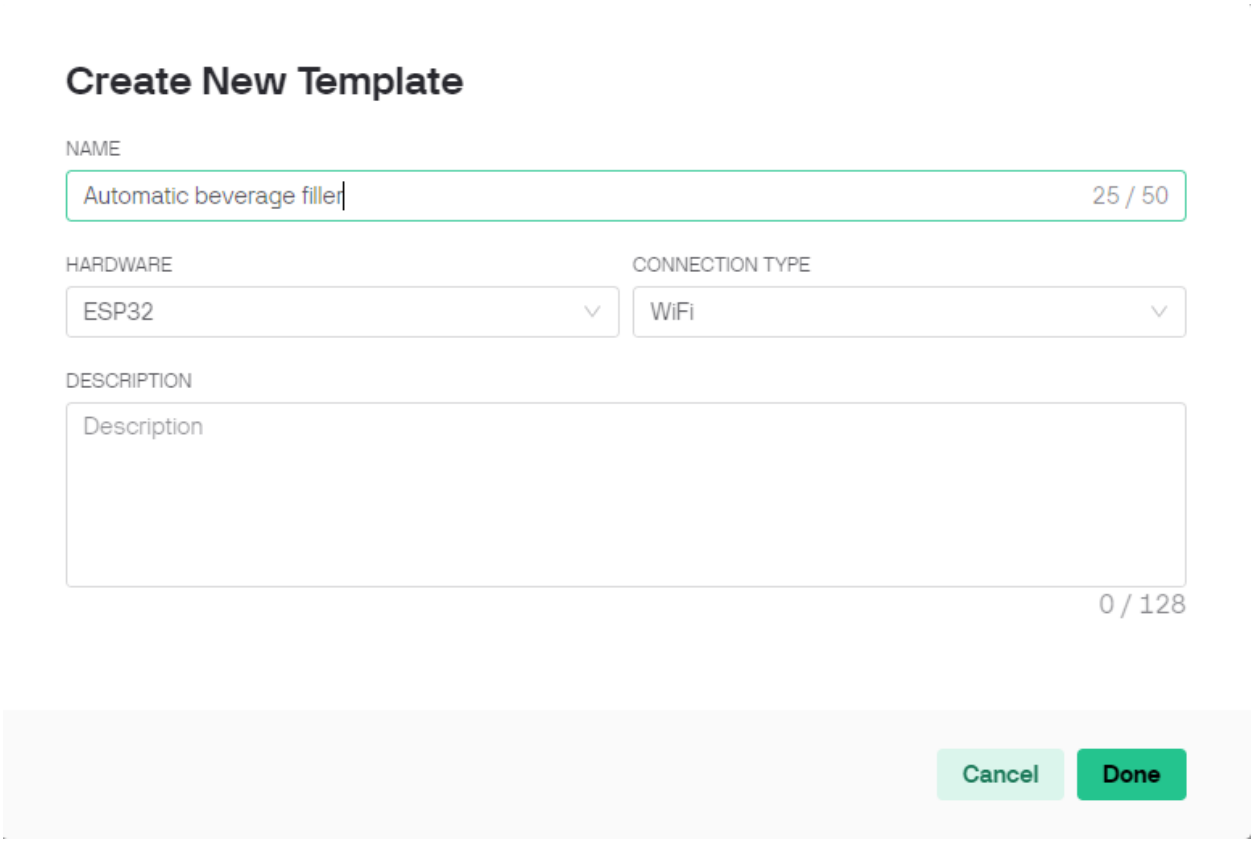
Biblioteka je zbirka unaprijed napisanoga koda koji se koristi za izvođenje određenih zadataka. Namijenjene su najprije kako bi se smanjila količina koda koju programer treba napisati pružanjem funkcija i klasa koje se mogu pozivati prema potrebi programera.

Uz sve spomenute prednosti ovog razvojnog okruženja, vrijedno je spomenuti i biblioteke kao prednost. Arduino IDE ima već ugrađen poveći skup biblioteka za lakšu izgradnju koda, ali omogućuje i proširivanje tog skupa vrlo jednostavnim postupkom instaliranja novih biblioteka, što rezultira još jednostavnijim, prilagodljivijim i bržim razvojem koda. Ove spomenute prednosti omogućuju manje iskusnim programerima razvoj programskog koda u mikrokontrolerima te razvoj kompleksnijih projekata.

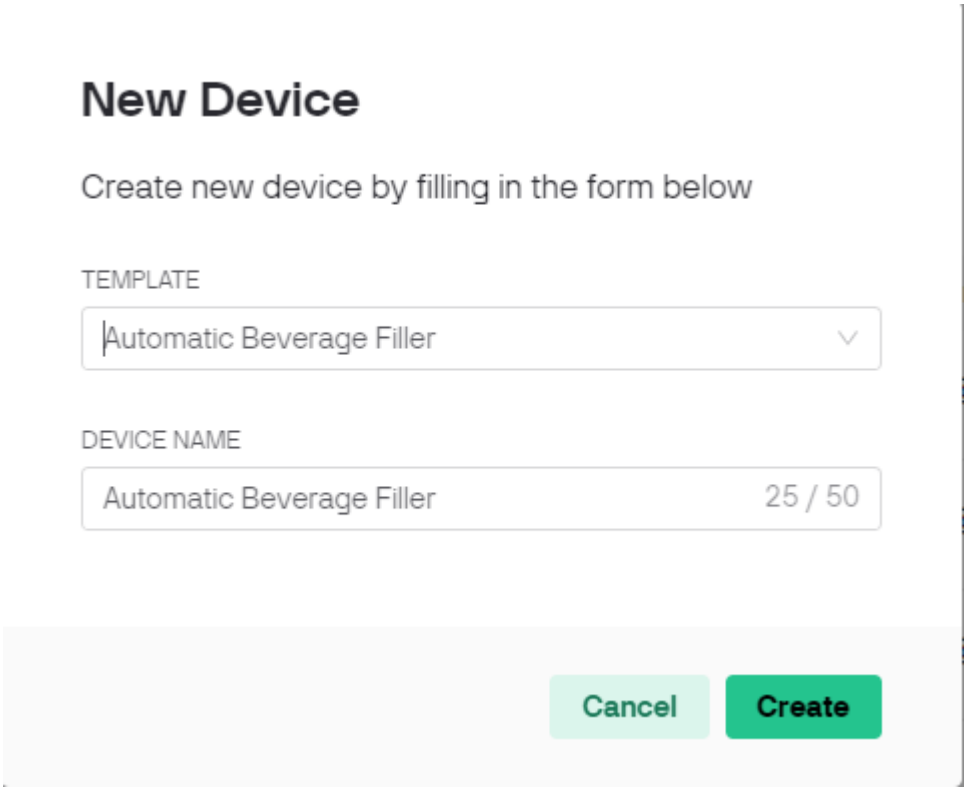
2.5.2. Blynk

Blynk je IoT platfroma za iOS ili Android pametne telefone koja se koristi za upravljanje mikrokontrolerom putem interneta. Omogućuje stvaranje grafičkog sučelja ili HMI-a (engl. *Human machine interface*). Za povezivanje mikrokontrolera s Blynk aplikacijom potrebne su tri stavke kao što su sama Blynk aplikacija, server i Blynk biblioteka. U aplikaciji, korisničko sučelje sadržava razne *widgete*, kao što su virtualne tipke, klizači, *displayi* i ostali, koji se mogu proizvoljno oblikovati i pozicionirati na željeni dio ekrana što omogućava jednostavno korištenje i upravljanje. Blynk pruža gotov *cloud server* koji se ne naplaćuje te na kojem se spremaju svi podaci za ispravno funkcioniranje sustava. Blynk biblioteka funkcionalna je za preko 400 pločica kao što su ESP32, Raspberry Pi, Arduino, Particle i mnoge druge.

Kako bi povezivanje i upravljanje pomoću Blynk aplikacije s mikrokontrolerom bilo uspješno najprije je potrebno instalirati Blynk aplikaciju, zatim stvoriti novi projekt i u njemu stvoriti tzv. „*template*“ koji najčešće predstavlja naziv projekta, kao što se može vidjeti na slici 2.2. Idući korak jest dodavanje novog uređaja s kojim se aplikacija povezuje kao na slici 2.3. Svaki projekt generira svoj tzv. autentifikacijski token koji je nužan za povezivanje mikrokontrolera s aplikacijom, slika 2.4. U korisničkom sučelju moguće je podesiti *widgete* kojima će se slati naredbe mikrokontroleru kao na slici 2.5. U korištenom softverskom okruženju potrebno je instalirati Blynk biblioteku te kroz nekoliko linija koda, uz unošenje autentifikacijskog tokena, povezati se na mrežu. Na kraju u softverskom okruženju piše se programski kod za mikrokontroler te je potrebno testirati funkcionalnost upravljanja mikrokontrolera putem aplikacije.



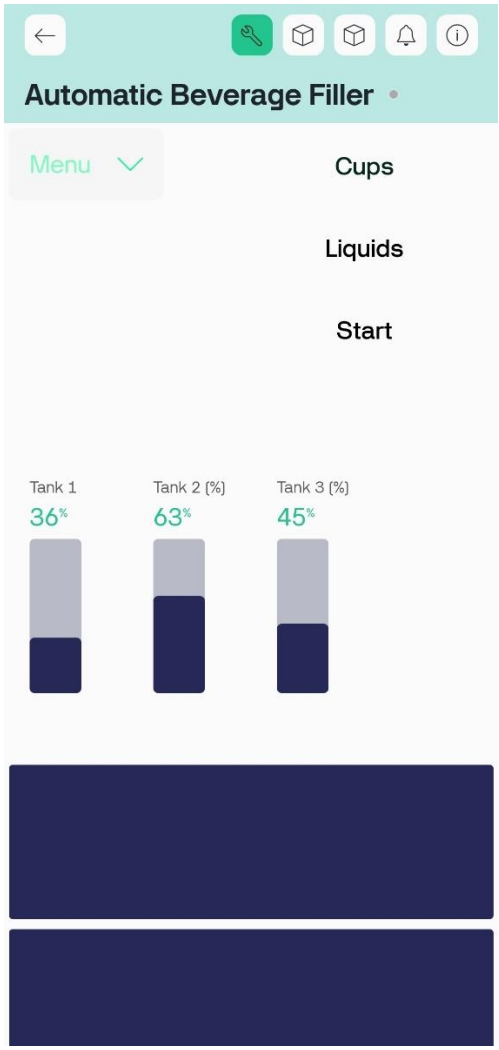
Slika 2.2 *Kreiranje "template-a"*



Slika 2.3 *Dodavanje uređaja*

```
#define BLYNK_TEMPLATE_ID "TMPL[redacted]"
#define BLYNK_TEMPLATE_NAME "Automatic Beverage Filler"
#define BLYNK_AUTH_TOKEN "[redacted] - Q0Rx"
```

Slika 2.4 *Generirani podaci potrebni za povezivanje mikrokontrolera s projektom*



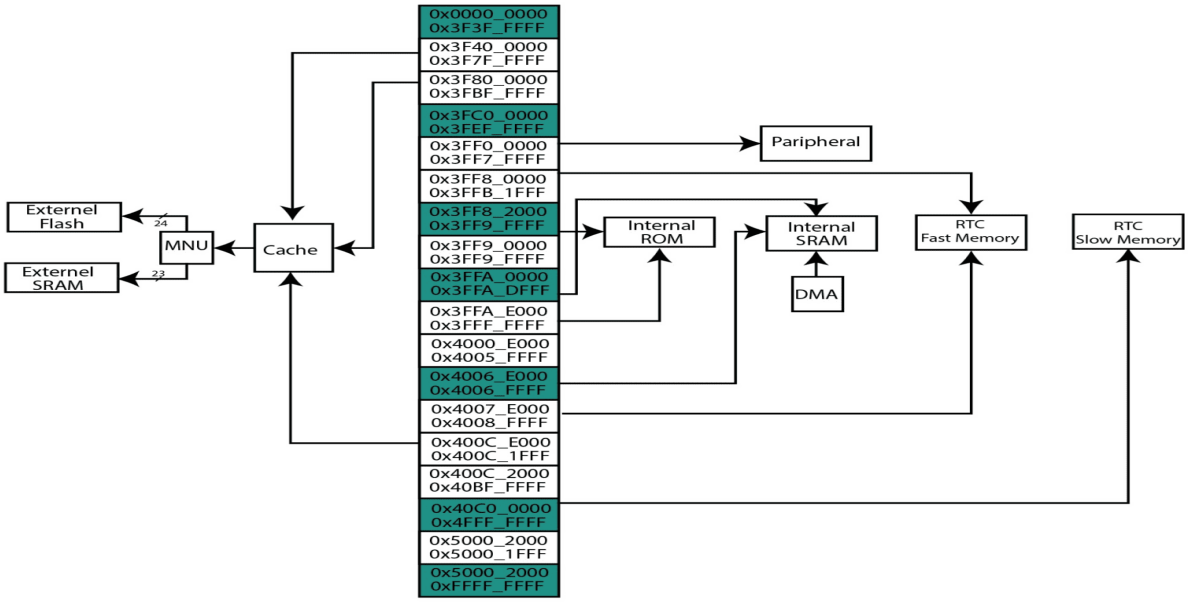
Slika 2.5 *Prilagođeni izgled aplikacije na mobilnom uređaju*

2.6. ESP32

ESP32 je serija jeftinih, a naprednih mikrokontrolera s integriranim Wi-Fi i bluetooth modulima koje je razvila tvrtka Espressif Systems. Nadogradnja je ESP8266 mikrokontrolera te u odnosu na njega sadrži značajnija poboljšanja. Prikladan je za jednostavnije projekte pa tako i za automatsku punilicu.

Za izvrsne performanse ovoga mikrokontrolera zadužen je ugrađeni dvojezgreni procesor Xtensa LX6 čija je arhitektura prikazana na slici 2.6. U pogledu bežičnih komunikacija podržava Wi-Fi za 802.11 standarde te bluetooth 4.2 i BLE (engl. *Bluetooth Low Energy*). Sadrži 36 GPIO pinova, 16 ADC pinova za pretvaranje analognih signala u digitalne, dva DAC pina za pretvaranje digitalnih signala u analogne, 16 pinova koji podržavaju PWM koji omogućavaju precizno upravljanje brzinom motora i razne druge funkcije [9]. Sadrži nekoliko komunikacijskih protokola kao što su SPI, I2C, UART i CAN koji omogućuju učinkovitu komunikaciju s ostalim uređajima.

U izradi automatske punilice, s gledišta iskoristivih GPIO pinova, postižu se limiti ovoga kontrolera. S obzirom da određeni pinovi imaju predodređena stanja, kao što su HIGH ili LOW, za vrijeme procesa *bootanja* mikrokontrolera potrebno ih je postaviti u to stanje. Stoga, ne preporuča se na iste pinove povezivati periferne uređaje kako ne bi došlo do neželjene greške. Ukoliko su na iste pinove povezani uređaji, potrebno ih je ukloniti prije procesa *bootanja*. Ovo za posljedicu ima smanjeni broj pinova na koje je moguće povezati periferne uređaje.



Slika 2.6 *Arhitektura Xtensa LX6 mikroprocesora [10]*

3. RAZVOJ SOFTVERA

ESP32 mikrokontroler programiran je u Arduino IDE okruženju. Najprije je potrebno dodati ESP32 pločicu u Arduino IDE upisivanjem URL veze u za to predviđeni prostor [11]. Nakon toga nužno je instalirati paket od Espressif Systems. Idući korak jest odabrati ESP32 pločicu u ovisnosti o nazivu pločice koju se planira programirati, odabrati „Port" na kojem se nalazi te omogućiti serijsku komunikaciju. Kako bi program funkcionirao ispravno, najprije je potrebno instalirati biblioteke.

Korištene biblioteke pri razvoju softvera su *BlynkSimpleEsp32.h, WiFi.h, WiFiClient.h, A4988.h, Wire.h, LiquidCrystal_I2C.h i ezButton.h*. *BlynkSimpleEsp32.h, WiFi.h, WiFiClient.h* biblioteke služe za povezivanje mikrokontrolera s Blynk cloud serverom i WiFi mrežom te omogućuju mikrokontroleru ulogu klijenta u mreži. Biblioteka *A4988.h* služi za programiranje istoimenog *drivera* kojim se upravlja koračnim motor. Biblioteke *Wire.h* i *LiquidCrystal_I2C.h* omogućuju *I2C* komunikaciju te uporabu *LCD-a* sa samo dva pina (*SCL, SDA*).

Nakon što su sve biblioteke instalirane, potrebno je dodijeliti pinove na koje su spojeni svi periferni sklopovi te stvoriti varijable koje će biti nužne za ispravno funkcioniranje koda. Na slici 3.1 su prikazane varijable *pump1, pump2, pump3* koje predstavljaju tri pumpe u sustavu te pinovi na koje su spojene, kao i varijable *wfs1, wfs2, wfs3* koje predstavljaju tri senzora protoka. Na isti način su ostalim perifernim sklopovima, osim *LCD-a*, dodijeljeni zasebni pinovi.

```
65   const int pump1 = 18;
66   const int pump2 = 5;
67   const int pump3 = 19;
68   const int wfs1 = 39;
69   const int wfs2 = 36;
70   const int wfs3 = 35;
```

Slika 3.1 Varijable pumpi i senzora protoka

Na slici 3.2 prikazane su varijable *duration* i *distance* koje će služiti za izračunavanje stvarne udaljenosti koju mjeri ultrazvučni senzor. Pomoću varijabli *d, h* i *distance* izračunava se varijabla *postotak* koja će predstavljati razinu tekućine u spremniku preračunatu u postotak.

```
128   long duration1, duration2, duration3;
129   float distance1, distance2, distance3;
130   int postotak1, postotak2, postotak3;
131   float h = 11; //visina tekućine kada je spremnik na 100%
132   float d = 9; //udaljenost senzora od tekućine kada je spremnik 100% pun
```

Slika 3.2 Varijable za ultrazvučne senzore

U *Setup()* funkciji pinovi se postavljaju kao ulaz (engl. *input*) ili izlaz (engl. *output*), kao što se može vidjeti na slici 3.3. Također, u istoj funkciji postavljaju se prekidi (engl. *interrupt*) koji zaustavljaju izvođenje glavnog toka programa kako bi se izvršio određeni zadatak, započet specifičnim događajem, najčešće senzorom ili na vremenskoj bazi.

```
213   pinMode(pump1, OUTPUT);
214   pinMode(pump2, OUTPUT);
215   pinMode(pump3, OUTPUT);
216
217   pinMode(wfs1, INPUT_PULLUP);
218   pinMode(wfs2, INPUT_PULLUP);
219   pinMode(wfs3, INPUT_PULLUP);
```

Slika 3.3 Definiranje pinova

```
240   attachInterrupt(digitalPinToInterrupt(irSensorPin), handleInterrupt, CHANGE);
241
242   attachInterrupt(digitalPinToInterrupt(wfs1), pulseCounter1, FALLING);
243   attachInterrupt(digitalPinToInterrupt(wfs2), pulseCounter2, FALLING);
244   attachInterrupt(digitalPinToInterrupt(wfs3), pulseCounter3, FALLING);
```

Slika 3.4 Programski prekidi (engl. Interrupt)

Na slici 3.4 su prikazana četiri programska prekida. U prvom programskom prekidu *digitalPinToInterrupt(irSensorPin)* predstavlja aktivaciju samog prekida infracrvenim senzorom, gdje je *irSensorPin* infracrveni senzor. *handleInterrupt* predstavlja funkciju koja će biti pozvana kada se prekid pozove. Zadnji atribut predstavlja način aktivacije prekida, u ovom slučaju je postavljen na *CHANGE*. U ovom sustavu ovaj prekid omogućuje zaustavljanje koračnog motora u trenutku kada je infracrveni senzor aktivan, odnosno kada je nadolazeća boca detektirana. Iduća tri programska prekida aktiviraju se kada senzori protoka detektiraju tok tekućine te pozivaju funkcije *pulseCounter1, pulseCounter2* i *pulseCounter3*.

Kako bi puštanje sustava u rad bilo u potpunosti kontrolirano, u glavnoj petlji (*loop()*), najprije se sustav mora inicijalizirati, slika 3.5. Inicijalizacija je izvedena na način da se, ako je zastavica *initialization_req* koja se postavlja putem aplikacije istinita, poziva funkcija *Rotate()* koja postavlja zastavicu *motorRunning* kao istinitu. U idućoj liniji provjerava se istinitost zastavice *motorRunning*. Ukoliko je istinita, koračni motor započinje s radom pomoću funkcije *stepper.startMove()* te se varijabli *stepperState* dodjeljuje vrijednost 1 koja predstavlja proces inicijalizacije. Idući korak jest provjera istinitosti zastavice *flag_sensorActive* i vrijednosti *stepperState* varijable te proteklo vrijeme od prve detekcije boce. Odnosno, ukoliko je infracrveni senzor detektirao bocu, motor se rotira te ako je proteklo vrijeme veće ili jednako *stopDelay* (400 milisekundi), koračni motor će se zaustaviti pomoću funkcije *stopRotate* te dodijeliti *stepperState* varijabli vrijednost 2, koja predstavlja spremnost za punjenje boce. Varijabli *stopDelay* dodijeljena je vrijednost od 400 milisekundi kako bi se postigla odgoda trenutačnog zaustavljanja boce te na taj način došla na željenu poziciju za punjenje. Na kraju se poziva funkcija *Initialized()* koja postavlja zastavicu *initialized* kao istinitu te zastavicu *initialization_req* kao neistinitu.

```
308   if(initialization_req){
309     Rotate();
310     if(motorRunning){
311       Serial.println(stepperState);
312       stepper.startMove(1 * MOTOR_STEPS);
313       stepperState = 1;
314       if (flag_sensorActive && stepperState == 1 && millis() - lastActiveTime
          >= stopDelay)
315       {
316         flag_sensorActive = false;
317         stopRotate();
318         stepperState = 2;
319         Serial.println("Stop");
320         Initialized();
321       }
322     }
323     stepper.nextAction();
324   }
```

Slika 3.5 Logika za inicijalizaciju sustava

Nakon što je sustav inicijaliziran, motor ne rotira, pritisnuta je tipka *start* u aplikaciji, vrijednost stanja *stepperState* je 2 ili 3, zatražena je jedna ili više boca za punjenje, zbroj postotaka svake zatražene tekućine iznosi 100% te sve dok je broj napunjenih boca manji od broja zatraženih boca, poziva se funkcija *Filling()*. U funkciji *Filling()* zastavica *filled* se postavlja kao istinita. Ukoliko je ista zastavica istinita, poziva se funkcija *resFillingParam()* koja resetira parametre punjenja kako bi se omogućilo ponovno punjenje svake iduće boce. Također, povećanjem varijable *cups_filled* prati se broj napunjenih boca. Varijabla *stepperState* poprima vrijednost 3, koja označava stanje rotacije motora, zatim se koračni motor rotira za 40°. U idućem koraku provjerava se je li motor zaustavljen, odnosno, je li prešao 40°. Ukoliko jest, varijabla *stepperState* poprima vrijednost 2, označavajući da je nova boca na mjestu za punjenje. Posljednja provjera jest provjera je li broj napunjenih boca jednak broju zatraženih boca. Ukoliko jest, varijable *cups_filled* i *cups_req* postavljaju se na nulu. (Slika 3.6)

```
328   if(initialized && !motorRunning && start && (stepperState == 2 || stepperState == 3)){
329
330     if(cups_req >= 1){
331       if(liq_req_sum == 100){
332         while(!flag_sensorActive && cups_filled <= cups_req && stepperState == 2){
333           flag_sensorActive = false;
334           Serial.print("Sensor active: ");Serial.println(flag_sensorActive);
335           Filling();
336           if(filled){
337             resFillingParam();
338             cups_filled++;
339             //Serial.println(flag_sensorActive);
340             Serial.print("Cups filled: ");Serial.print(cups_filled);Serial.print("/");Serial.println(cups_req);
341             filled = false;
342             stepperState = 3;
343             stepper.enable();
344             stepper.startMove(40);
345           }
346         }
347         if(stepper.getCurrentState() == 0){
348           Serial.println("STOP");
349           stepperState = 2;
350           stepper.disable();
351         }
352         stepper.nextAction();
353
354         if(cups_filled == cups_req){
355           start = false;
356           cups_filled = 0;
357           cups_req = 0;
358           stepper.disable();
359           //resAppParam();
360           Serial.print(start);
361           Serial.print("\n");
362           Serial.print(cups_filled);
363           Serial.print("\n");
364           Serial.print(cups_req);
365           Serial.print("\n");
366         }
367       }else{
368         Serial.print("Max 100%");
369       }
370     }else{
371       Serial.print("Zahtjevanih boca: 0");
372       Serial.print("\n");
373     }
374   }
375 }
```

Slika 3.6 Glavna petlja koda

Unutar funkcije *Filling()* cjelokupna logika odvija se u *if (currentMillis - previousMillis > interval){}* petlji, gdje varijabla *interval* ima vrijednost 1000 (1s). Najprije se izvršava preračunavanje pulseva, koje detektiraju senzori protoka, u protok. Stoga, *totalMilliLitres1, totalMilliLitres2 i totalMilliLitres3* predstavljaju protok u mililitrima.

```
if (currentMillis - previousMillis > interval) {

  pulse1Sec1 = pulseCount1;
  pulse1Sec2 = pulseCount2;
  pulse1Sec3 = pulseCount3;
  pulseCount1 = 0;
  pulseCount2 = 0;
  pulseCount3 = 0;

  flowRate1 = ((1000.0 / (millis() - previousMillis)) * pulse1Sec1) / calibrationFactor;
  flowRate2 = ((1000.0 / (millis() - previousMillis)) * pulse1Sec2) / calibrationFactor;
  flowRate3 = ((1000.0 / (millis() - previousMillis)) * pulse1Sec3) / calibrationFactor;

  previousMillis = millis();

  flowMilliLitres1 = (flowRate1 / 60) * 1000;
  flowMilliLitres2 = (flowRate2 / 60) * 1000;
  flowMilliLitres3 = (flowRate3 / 60) * 1000;

  totalMilliLitres1 += flowMilliLitres1;
  totalMilliLitres2 += flowMilliLitres2;
  totalMilliLitres3 += flowMilliLitres3;            //Preračunavanje pulseva u mililitre
```

Slika 3.7 Preračunavanje protoka unutar funkcije Filling()

Nakon preračunavanja protoka, izvršava se logika za upravljanje pumpama. Ova logika strukturirana je tako da provjerava sve moguće kombinacije vrsta tekućina i njihovih postotaka, koji mogu biti 0%, 100% ili bilo koja vrijednost između. Na slici 3.8 prikazan je jedan od slučajeva, kada je putem aplikacije zatraženo miješanje sve tri tekućine, odnosno, varijable *liq1_req, liq2_req i liq3_req* svake pojedinačno imaju vrijednost veću od 0 te je njihov zbroj jednak 100. U idućem uvjetu provjerava se je li varijabla *totalMilliLitres1* manja od varijable *liq1_ml*, koja predstavlja zatraženi volumen u mililitrima. Sve dok je ova tvrdnja istinita pumpa će biti uključena. Kada tvrdnja postane neistinita, to znači da je postignuta željena količina tekućine te se pumpa isključuje. Na kraju petlje, zastavica *pump1_done* postavlja se kao istinita kako bi se znalo da je prva pumpa isključena. Svaka iduća petlja provjerava varijable vezane za onu tekućinu koja se pumpa te se dodatno provjerava je li prethodna pumpa isključena. Na ovaj način postignut je sekvencijalni rad pumpi koji je bio nužan kako ne bi došlo do izlijevanja tekućine pri istovremenom radu sve tri pumpe. Na kraju vanjske petlje provjerava se jesu li sve pumpe isključene, ako jesu zastavica *filled* se postavlja kao istinita te na taj način omogućuje nastavak izvršavanja glavne petlje koda.

```
455   if(liq1_req > 0 && liq2_req > 0 && liq3_req > 0 && liq_req_sum == 100){
456
457     if (liq1_req > 0 && totalMilliLitres1 < liq1_ml) {
458       digitalWrite(pump1, HIGH);
459     } else {
460       digitalWrite(pump1, LOW);
461       pump1_done = true;
462     }
463
464     if (pump1_done && liq2_req > 0 && totalMilliLitres2 < liq2_ml) {
465       digitalWrite(pump2, HIGH);
466     } else if (pump1_done) {
467       digitalWrite(pump2, LOW);
468       pump2_done = true;
469     }
470
471     if (pump1_done && pump2_done && liq3_req > 0 && totalMilliLitres3 < liq3_ml) {
472       digitalWrite(pump3, HIGH);
473     } else if (pump1_done && pump2_done) {
474       digitalWrite(pump3, LOW);
475       pump3_done = true;
476     }
477
478     // Provjeravamo je li sve napunjeno
479     if (pump1_done && pump2_done && pump3_done) {
480       filled = true;
481     }
482   }
```

Slika 3.8 Logika za rad pumpi pri zahtjevu miješanja tri tekućine unutar funkcije Filling()

Kako bi se omogućilo povezivanje mikrokontrolera s WiFi mrežom najprije se definiraju makroi te se uključuju biblioteke kao na slici 3.9.

```
1   #define BLYNK_TEMPLATE_ID "TMPL[redacted]"
2   #define BLYNK_TEMPLATE_NAME "Automatic Beverage Filler"
3   #define BLYNK_PRINT Serial
4
5   #include <BlynkSimpleEsp32.h>
6   #include <WiFi.h>
7   #include <WiFiClient.h>
```

Slika 3.9 Makroi i biblioteke za povezivanje s WiFi mrežom

Za povezivanje mikrokontrolera s WiFi mrežom potrebno je unijeti autentifikacijski token, naziv WiFi mreže te lozinku istoga kao na slici 3.10.

```
19  char auth[] = "[redacted]0Rx";
20  char ssid[] = "TCHQ[redacted]";
21  char pass[] = "[redacted]";
```

Slika 3.10 Varijable o podacima WiFi mreže

Blnyk.begin() funkcija omogućuje povezivanje mikrokontrolera s Blynk serverom, dok se s *timer.setInterval()* funkcijom poziva *sendLiqLevelData()* funkcija svakih 6 sekundi.

```
255   Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
256   timer.setInterval(6000L, sendLiqLevelData);
```

Slika 3.11 Funkcije za povezivanje s Blynk aplikacijom te slanje podataka u istu

Funkcija *sendLiqLevelData(),* pomoću *Blynk.virtualWrite()* funkcije, u aplikaciju šalje podatke o postotku napunjenosti spremnika, koristeći virtualne pinove *V6, V7 i V8*. U korisničkom sučelju aplikacije ovi pinovi predstavljaju stupčaste dijagrame pod nazivom *Tank 1, Tank 2 i Tank 3*.

```
693   void sendLiqLevelData(){
694     Blynk.virtualWrite(V6, postotak1);
695     Blynk.virtualWrite(V7, postotak2);
696     Blynk.virtualWrite(V8, postotak3);
697   }
```

Slika 3.12 Funkcija sendLiqLevelData()

3.1. Prednosti i nedostaci softvera

Pri razvoju softvera za automatsku punilicu pića ustrajalo se da kod bude što jednostavniji i čitljiviji. Jednostavnost koda omogućava veliku brzinu izvršavanja cjelokupnog koda, što je nužno za preciznost, brže vrijeme punjenja boca te sve ostale atribute koji karakteriziraju kvalitetnu punilicu. Kako bi softver bilo jednostavno za održavati ili kako bi se napravile manje promjene, važno je da isti bude čitljiv. Softverski je omogućeno povezivanje s Wi-Fi mrežom, preko koje je moguće mijenjati vrijednosti određenim varijablama unutar softvera, što za posljedicu ima mogućnost upravljanja uređajem na daljinu.

Određeni problemi javljaju se u pogledu daljinskog upravljanja, ali se ne smatraju kritičnima. Pri upravljanju uređajem na Wi-Fi mreži može doći do prekida veze, što za posljedicu ima nemogućnost upravljanja uređajem sve dok se pametni telefon ili uređaj ponovno ne poveže s Wi-Fi mrežom. Softver je razvijen tako da, ako dođe do prekida s Wi-Fi mrežom, uređaj ne može ući u nepoznato stanje. Stoga, uređaj će ili izvršiti proces punjenja ili uopće neće pokrenuti proces, ovisno kada je veza s Wi-Fi mrežom prekinuta.

3.2. Testiranje potrošnje

Karakteristika ovog sustava/uređaja koja je testirana jest potrošnja električne energije tijekom njegovog radnog ciklusa. U nastavku je prikazana podjela radnog ciklusa sustava kako bi bilo jasnije u kojim trenutcima je mjerenje potrošnje električne energije provedeno.

- Sustav isključen
- Trenutak uključivanja
- Mirovanje
- Rotacija boca (rad motora)
- Rad pumpe/i

Jasno je da sustav ne troši električnu energiju kada je isključen. Pri pokretanju, očekuje se nešto veća potrošnja u odnosu na stanje mirovanja, budući da se mikrokontroler u tom trenutku povezuje s Wi-Fi mrežom, dok u mirovanju obavlja samo pozadinske procese [12]. Nakon što se mikrokontroler povezao na mrežu u aplikaciji se podešavaju svi potrebni parametri kako bi mogao započet proces rotacije. U trenutku rotacije potrošnja ponajviše ovisi o samom koračnom motoru. Kada je prva boca detektirana, motor se zaustavlja te u istom trenutku, ako je boca dobro postavljena, putem aplikacije započinje proces punjenja. Zanemare li se razlike u potrošnji između sve tri pumpe, pri procesu punjenja očekuje se da će ukupna potrošnja biti približno ista, bez obzira na to radi li jedna ili sve tri pumpe. S obzirom na to da boca mora biti napunjena do 100% volumena, jedna pumpa će raditi duže ako je jedina uključena, dok će sve tri pumpe raditi kraće kada rade sekvencijalno. Ipak, zbroj vremena rada sve tri pumpe bit će približno jednako vremenu rada jedne pumpe koja puni cijelu bocu. Postavljeni omjeri tekućina u aplikaciji pri mjerenju potrošnje iznose 33%, 33% i 34% (zbroj = 100%).

3.2.1. Rezultati testiranja

Kako bi se dobili vjerodostojni rezultati potrošnje električne energije, ista je mjerena 10 puta u svakom koraku u ciklusu te se izračunava srednja vrijednost. Obavljena su testiranja kada rade jedna, dvije i sve tri pumpe istovremeno, ali kao osnovni graf potrošnje se uzima onaj s potrošnjom jedne pumpe iz razloga što se iz njega dobivaju svi potrebni podaci o potrošnji, odnosno, za sve ostale slučajeve graf varira ovisno o podacima koji su već izmjereni.

Br. mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Struja	131	130.4	130.9	128.6	130	130.2	130.6	129.8	130	130.1

Tablica 3.1 Potrošnja električne energije pri uključivanju sustava

Srednja vrijednost potrošnje električne energije pri uključivanju sustava:

$$I_{sr} = 130.16\ mA \approx 130\ mA,$$

Br. mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Struja	95	129.1	110.6	129.9	119.8	95	95.1	100.7	105.5	120.4

Tablica 3.2 Potrošnja električne energije u mirovanju sustava

Srednja vrijednost potrošnje električne energije u stanju mirovanja sustava:

$$I_{sr} = 110.24\ mA \approx 110\ mA,$$

Br. mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Struja	429.9	425.4	430.1	419.8	425.8	431	435.5	420.1	429	429.3

Tablica 1.3 Potrošnja električne energije pri rotaciji

Srednja vrijednost potrošnje električne energije pri rotaciji::

$$I_{sr} = 427.59\ mA \approx 430\ mA,$$

Br. mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Struja	320	315.4	329	319.9	320.4	329.6	325.4	316	320.1	319.3

Tablica 3.4 Potrošnja električne energije pri radu jedne pumpe

Srednja vrijednost potrošnje električne energije pri radu jedne pumpe:

$$I_{sr} = 321.51\ mA \approx 320\ mA,$$

Br. mjerenja	1.		2.		3.		4.		5.	
Pumpa	1	2	1	2	1	2	1	2	1	2
Struja	320.5	320.1	314.8	329.3	323.8	322	315.1	317	330	325.4

Br mjerenja	6.		7.		8.		9.		10.	
Pumpa	1	2	1	2	1	2	1	2	1	2
Struja	320	315.2	330	328	326.5	317.8	325.4	320	329.9	320.7

Tablica 3.5 Potrošnja električne energije pri radu dvije pumpe (omjer tekućina je 50/50)

Srednja vrijednost potrošnje električne energije pri radu dvije pumpe:

$$I_{sr} = 322.5\ mA \approx 320\ mA,$$

Br. mjerenja	1.			2.			3.			4.		
Pumpa	1	2	3	1	2	3	1	2	3	1	2	3
Struja	314	322.9	320.1	326.6	330.7	330	319.6	316.3	329	315.4	330	321.3

5.			6.			7.			8.		
1	2	3	1	2	3	1	2	3	1	2	3
320	317.4	318.6	324.1	320	325.4	315	322.5	319.3	329.3	320.9	330

9.			10.		
1	2	3	1	2	3
330.1	326.2	324.4	316.1	320.1	328.7

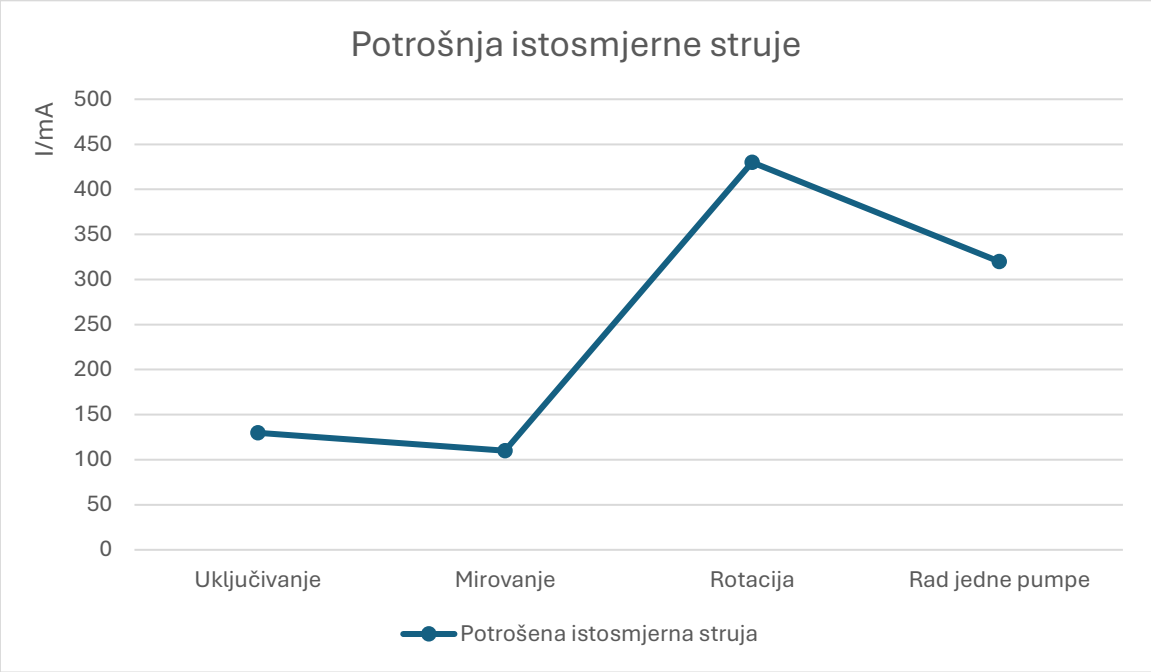
Tablica 3.6 Potrošnja električne energije pri radu tri pumpe (omjer tekućina je 33/33/34)

Srednja vrijednost potrošnje električne energije pri radu tri pumpe:

$$I_{sr} = 322.8\ mA \approx 320\ mA.$$

Iz provedenih testiranja primjećuje se da su najveća odstupanja potrošnje u trenutku mirovanja sustava, što ovisi o pozadinskim procesima. Koračni motor troši najviše struje, oko 430 mA

umanjeno za iznos struje u stanju mirovanja. Isto tako zaključuje se da je pretpostavka o potrošnji pri radu sve tri pumpe ispravna.



Slika 3.13 Grafički prikaz potrošnje istosmjerne struje sustava

Graf potrošnje istosmjerne struje jasno pokazuje u kojim trenutcima sustav „povlači“ najviše struje. Očekivano, u mirovanju sustav troši najmanje struje, dok pri rotaciji troši gotovo 4 puta više struje nego u mirovanju.

Kako bi se potrošnja preračunala u električnu snagu istosmjerne struje koristi se sljedeća formula:

$$P = U * I \ [W],$$

uvrštavanjem izmjerene vrijednosti struje u trenutku rotacije i poznatog napona napajanja u navedenu formulu, dobiva se slijedeći rezultat:

$$P_{rot} = 12\ V * \ 430\ mA = 5.16\ W.$$

Zaključuje se da, pri rotaciji, ovaj uređaj/sustav ima snagu od 5.16 W.

4. ZAKLJUČAK

U današnje vrijeme brzog razvoja tehnologija pa tako i razvojnih softverskih okruženja, vrlo je jednostavno automatizirati, odnosno, razviti softver za jednostavniji sustav. Iako se jednostavnost smatra prednošću, izazovi koje donosi ova jednostavna automatizacija odnose se na pouzdanost i sigurnost.

Glavni zadatak ovog rada bio je razviti softver za automatsku punilicu pića na što jednostavniji i optimalniji način. Iako je softver u potpunosti funkcionalan, postoje određeni nedostaci poput prekida s Wi-Fi mrežom na koje je skoro nemoguće softverski utjecati. Također, detaljno je prikazano kako je softver razvijan te na koji način pojedini dijelovi koda funkcioniraju te za što su zaduženi. Razni su načini razvijanja softvera, no u ovom radu odabran je način kojim bi se postigla maksimalna jednostavnost i čitljivost koda u svrhu jednostavnijeg održavanja i nadograđivanja istog. Na temelju provedenih testova potrošnje električne energije, utvrđeno je da sustav pokazuje nisku razinu energetske potrošnje. Ovaj rezultat postignut je optimizacijom arhitekture sustava, koja obuhvaća minimizaciju broja suvišnih komponenti te integraciju softverskih i hardverskih rješenja. Važno je istaknuti optimizaciju upravljanja koračnim motorom, koji troši električnu energiju i u stanju mirovanja. Implementacijom kombinacije softverskih i hardverskih rješenja u potpunosti je eliminirana ta potrošnja.

Ovakav sustav moguće je unaprijediti uvođenjem raznih dodatnih funkcija u softver. Ove funkcije imaju potencijal za smanjenje količine napisanog koda te podizanja pouzdanosti sustava na višu razinu. Iako softver ima ključnu ulogu u kvaliteti cjelokupnog sustava, još bolje performanse mogle bi se postići ugradnjom skuplje opreme, no to za posljedicu ima veće financijske izdatke.

LITERATURA

[1] Md. Liton Ahmed, Shantonu Kundu, Md. Rafiquzzaman: Automatic Bottle Filling System Using PLC Based Controller, pristup 22.6.2024., https://core.ac.uk/download/211829785.pdf

[2] Yusuf Perwej, Kashiful Haq, Firoj Parwej, Mumdouh M. Mohamed Hassan: The Internet of Things (IoT) and its Application Domains, pristup 23.6.2024., https://www.researchgate.net/profile/Dr-Yusuf-Perwej/publication/332374473_The_Internet_of_Things_IoT_and_its_Application_Domains/links/5cb245afa6fdcc1d49931068/The-Internet-of-Things-IoT-and-its-Application-Domains.pdf

[3] Shree Krishna, Xianbin Wang: Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks, pristup 23.6.2024., https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7882669

[4] Neil Gershenfeld, Raffi Krikorian and Danny Cohen: Th Internet of Things, pristup 23.6.2024., https://cseweb.ucsd.edu/~schulman/class/cse291_f18/docs/cohen_internet.pdf

[5] Saleh Ali Alomari, Putra Sumari, Alireza Taghizadeh: A Comprehensive Study of Wireless Communication Technology for the Future Mobile Devices, pristup 23.6.2024., https://www.researchgate.net/profile/Saleh-Alomari/publication/290005450_A_comprehensive_study_of_wireless_communication_technology_for_the_future_mobile_devices/links/5b1637f5aca272d43b7ea2f4/A-comprehensive-study-of-wireless-communication-technology-for-the-future-mobile-devices.pdf

[6] Abdullah Ahmed Bahashwan, Mohammed Anbar, Nibras Abdullah, Tawfik Al-Hadhrami, and Sabri M. Hanshi: Review on Common IoT Communication Technologies for Both Long-Range Network (LPWAN) and Short-Range Network, pristup 24.6.2024., https://www.researchgate.net/profile/Abdullah-Bahashwan/publication/344933088_Review_on_Common_IoT_Communication_Technologies_for_Both_Long-Range_Network_LPWAN_and_Short-Range_Network/links/5f999766a6fdccfd7b85090e/Review-on-Common-IoT-Communication-Technologies-for-Both-Long-Range-Network-LPWAN-and-Short-Range-Network.pdf

[7] Globalna rasprostranjenost interneta u srpnju 2024.godine, pristup 29.8.2024., https://www.statista.com/statistics/269329/penetration-rate-of-the-internet-by-region/

[8] Susan A. Dart, Robert J. Ellison, Peter H. Feiler, and A. Nico Habermann Edited by Peter Fritzson: Overview of Software Development Environments, pristup 25.6.2024., https://ics.uci.edu/~andre/ics228s2006/dartellisonfeilerhabermann.pdf

[9] ESP32 datasheet, pristup 25.6.2024., https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf

[10] Arhitektura Xtensa LX6 mikroprocesora, pristup 25.6.2024., https://linuxhint.com/what-chip-esp32/

[11] URL veza za instalaciju pločice, pristup 29.8.2024., https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

[12] ESP32 Series Datasheet Version 4.6, Espressif Systems, 18.8.2024. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

POPIS OZNAKA I KRATICA

PLC - Programmable Logic Controller

IoT – Internet of Things

IIoT – Industrial Internet of Things

HVAC - Heating, Ventilation and Air Conditioning

I/O – Input/output

NFC - Near-field communication

LPWAN - Low-power, Wide-area Networks

IEEE - Institute of Electrical and Electronics Engineers

HMI – Human Machine Interface

BLE - Bluetooth Low Energy

GPIO – General Purpose Input/Otuput

ADC - Analog-to-Digital Converter

DAC - Digital-to-Analog Converter

PWM - Pulse Width Modulation

SPI - Serial Peripheral Interface

I2C - Inter-Integrated Circuit

UART - Universal Asynchronous Receiver/Transmitter

CAN - Controller Area Network

LCD - Liquid Crystal Display

SCL - Serial Clock Line

SDA - Serial Data Line

mA – miliAmper

SAŽETAK

Sustavi automatskog punjenja i miješanja pića neizostavni su u današnjim tvornicama koje proizvode različite vrste tekućina. Ubrzanim razvojem tehnologija za izradu softvera, dolazi do problema vezanih uz pouzdanost i sigurnost softvera. Stoga, uz hardver otporan na teške uvjete rada, od iznimne važnosti jest dobro razvijen i održavan softver. Kako bi se postigla pouzdanost i sigurnost softvera u ovom sustavu, najprije su se pokušali predvidjeti zadaci te mogući problemi sustava. Važno je razlučiti što je zadaća sustava te na taj način razviti „kostur“ softvera. Nakon testiranja osnovnih funkcija softvera pokazuju se određeni nedostaci koji su nakon tog otklonjeni. Daljnjim i detaljnijim testiranjem softvera izrađuju se razne funkcije, također na osnovu analiziranih nedostataka i zadataka sustava, što rezultira njegovim razvijanjem u smjeru bolje pouzdanosti i kvalitete.

Ključne riječi: Automatska punilica pića, automatizacija, Arduino IDE, Blynk, razvoj softvera

ABSTRACT

Automatic beverage filling and mixing systems are indispensable in today's factories that produce different types of liquids. With the rapid development of technologies for creating software, there are problems related to the reliability and safeness of software. Therefore, along with hard-wearing hardware, well-developed and maintained software is extremely important. In order to achieve the reliability and safeness of the software in this system, the tasks and possible problems of the system were first predicted. It is important to distinguish what the task of the system is and in this way to develop the "frame" of the software. After testing the basic functions of the software, certain defects were revealed, which were then removed. Through further and more detailed testing of the software, various functions are created, also based on the analyzed shortcomings and tasks of the system, which results in its development in the direction of better reliability and quality.

Keywords: Automatic beverage filler, automation, Arduino IDE, Blynk, software development

ŽIVOTOPIS

Marin Zdunić rođen je 24.12.2001. godine u Osijeku. Od rođenja živi u Osijeku, gdje je pohađao osnovnu školu Dobriša Cesarić. Godine 2016. upisuje Elektrotehničku i Prometnu školu Osijek, smjer Tehničar za mehatroniku. Po završetku srednje škole polaže maturu te godinu dana radi na održavanju automatiziranih strojeva i njihovom puštanju u rad. Godine 2021. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.