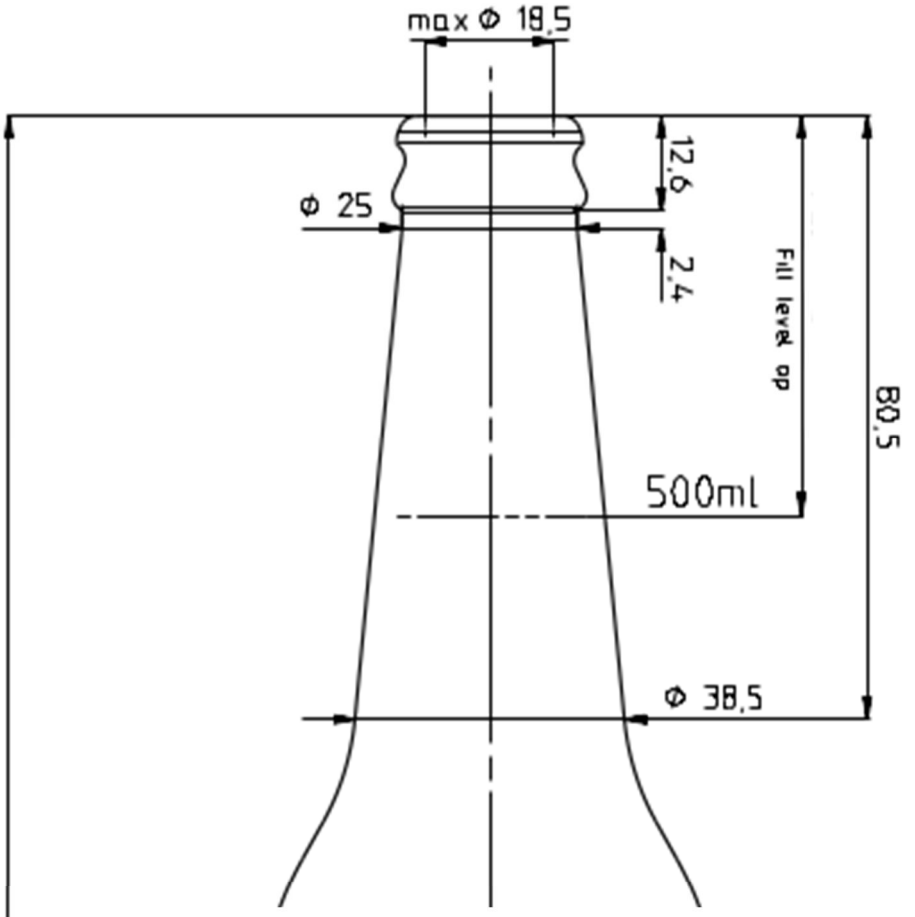
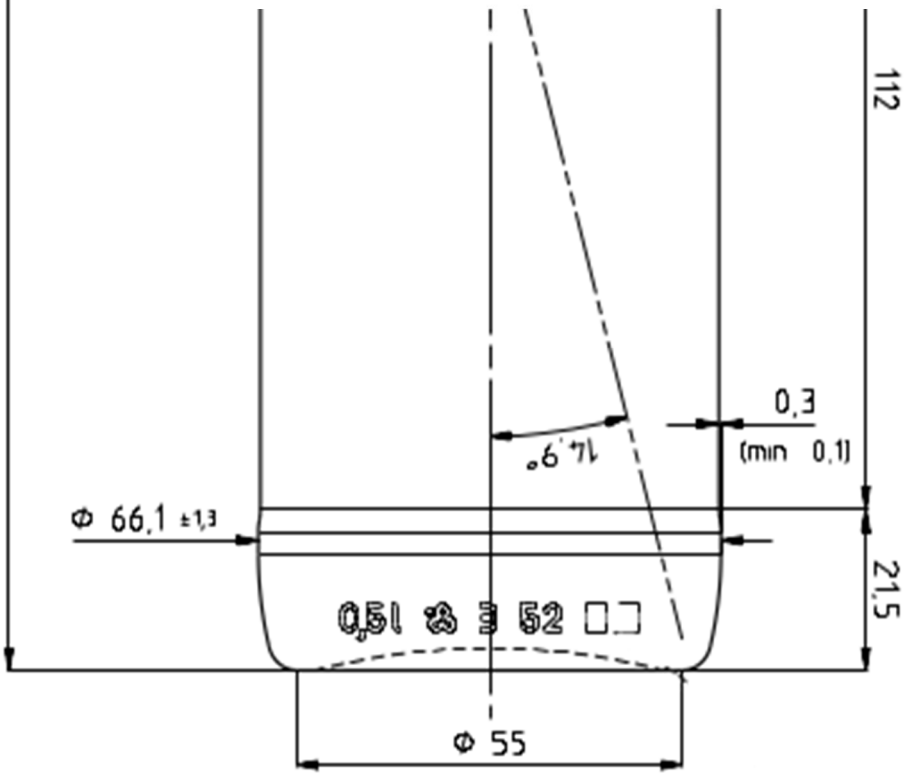
Fakultet elektrotehnike, strojarstva i brodogradnje Sveučilišta u Splitu



Sustav za praćenje, brojanje i pozicioniranje boca pri procesu punjenja boca unutar projekta OPINAU-FILLER-CV

Naručitelj: Codex Apertus d.o.o.
Put Radoševca 14
21 000 Split

temeljem vaučera izdanog u okviru Poziva na dostavu projektnih prijedloga Inovacijski vaučeri za MSP KK.03.2.2.03 (P21_INO:2018_0039)



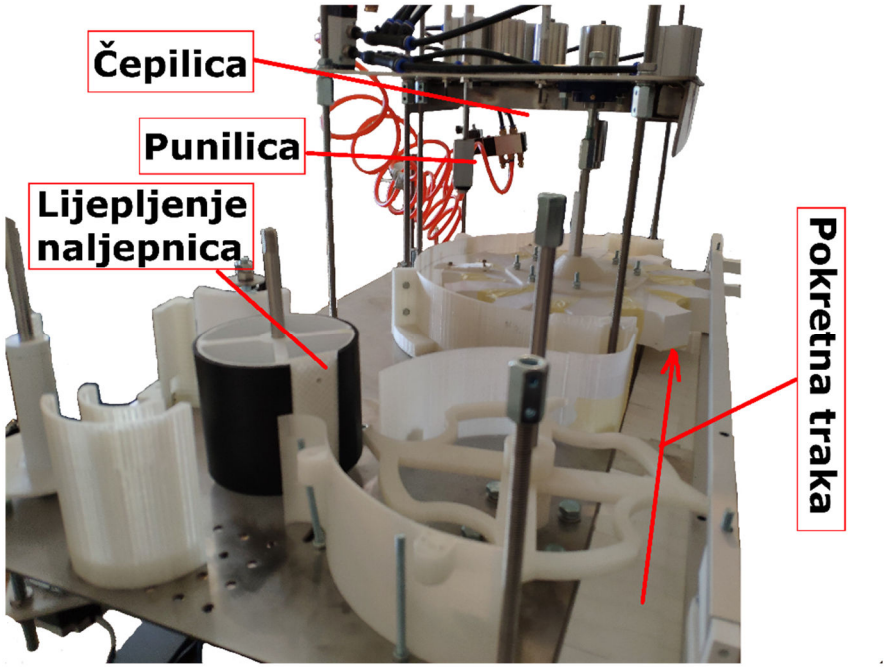
Autori:
doc. dr. sc. Ivo Stančić
izv. prof. dr. sc. Josip Musić

Split, srpanj, 2020.

SADRŽAJ

1. MOTIVACIJA I CILJ

Naručitelj sustava, tvrtka Codex Apertus d.o.o., je u procesu izrade *open-source* (sklopovski i programski) tipa punionice za male pivovare odnosno tzv. *craft* piva. Glavna odlika takvih pivovara odnosno piva je mala serija proizvodnje odnosno izrada od strane pojedinaca (entuzijasta) i manjih poduzeća. Stoga je jedna od glavnih odlika koje razvijeni sustav treba imati cjenovna pristupačnost kao i jednostavna dostupnost svih komponenti od kojih se sustav sastoji, pri čemu sustav ne mora biti u mogućnosti obraditi veliki broj boca kao velike pivovare. Dodatno, potrebno je da je sustav lako za održavati te zamijeniti neispravne komponente. Tragom navedenoga tvrtka Codex Apertus je izradila prototip punionice (Slika 1) koja sa sastoji od pokretne trake, dijela za lijepljenje naljepnica na boce, punilice pivom i čepilice, a dijelovi za koje su dostupni kao STL datoteke za 3D printanje, odnosno ključne elektroničke komponente putem platformi za trgovinu kao što su E-Bay odnosno AliExpress.



Slika 1 – Radni prototip OPINAU-FILLER sa naznačenim osnovnim dijelovima (strelica prikazuje smjer kretanja boce)

Glavni pokretači pokretne trake (ali i većine ostalih komponenti) su stepper/koračajni motori kao što je npr. Nema 23 bipolarni koračajni motor s korakom od 1.8 stupnja[1] sa popratnim sklopovljem za upravljanje (tzv. driverima). Iako navedene motore karakterizira dobra točnost i ponovljivost, budući da je cijeli sustav postavljen u konfiguraciji otvorene regulacijske petlje (tj. niz akcija se izvodi bez povratne informacije o njihovom (ne)uspjehu) potrebno je u cjeloviti sustav integrirati podsustav za kontrolu i praćenje stanja zanimljivih

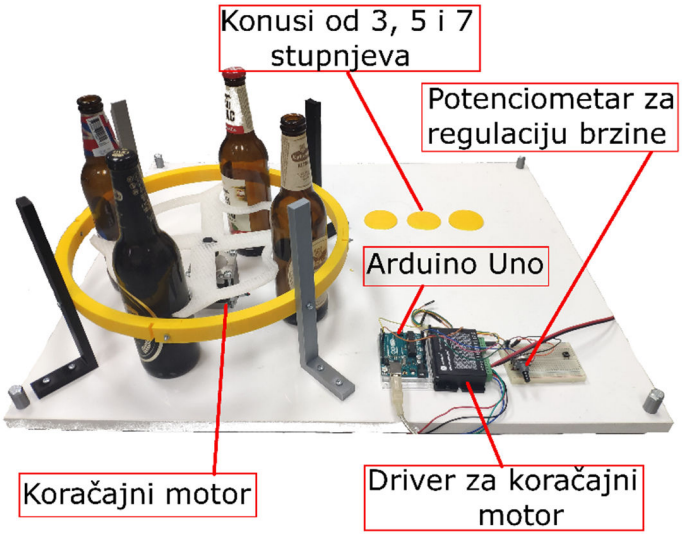
[1] https://www.omc-stepperonline.com/nema-23-bipolar-1-8deg-1-26nm-178-4oz-in-2-8a-2-5v-57x57x56mm-4-wires.html?search=23hs22-2804s

parametra koji utječu na kvalitetu rada sustava (npr. da li su boce dobro začepljene, napunjene do tražene razine, koliko boca je do sada napunjeno i slično). Na taj način bi se u stvarnosti zatvorila regulacijska petlja koja bi pak omogućila praćenje pojedinih parametara punionice kao i sprečavanje eventualnih problema ili oštećenja koja se mogu javiti kao posljedica nefunkcioniranje neke od komponenti punionice, ali i detekcije (anomalija) rada pojedinih dijelova punionice.

Zbog različitih potreba i zahtjeva za različite dijelove sustava, prepoznato je da jedan sustav nije u mogućnosti (poštujući prije spomenute zahtjeve o lakoj dostupnosti i cjenovnoj pristupačnosti) ispuniti sva tražena mjerenje, te se stoga predlaže uvođenje zasebnih modula sa striktno definiranom (ograničenom) funkcionalnošću. Stoga se u ovoj razvojnoj fazi cilj predložiti tri modula s potrebnim funkcionalnostima: 1.) modul za detekciju prisustva i ispravnosti čepa na boci postavljen nakon punilice/čepilice na samom izlazu s pokretne trake, 2.) modul za precizno pozicioniranje boca u kritičkim dijelovima sustavima kao što je punilica/čepilica, te 3.) modul za praćenje kretanja boca u dijelovima sustava od interesa kao npr. dijelu za lijepljenje naljepnica (ili čak u cijelom sustavu). Na ovaj način je ostvarena modularnost sustava uz mogućnost nadogradnje postojećih modula, ali i jednostavnu ugradnju novih. S time na umu, svaki od tri modula je na svome izlazu prilagođen za jednostavnu integraciju unutar ROS (eng. *Robotic Operating System*[2]) okvira kojega upravo karakterizira heterogenost i jednostavno dodavanje funkcionalnosti.

U nastavku studije su predstavljena tri razvijena prototipa modula na sklopovskoj i programskoj razini. Navedeni su osnovni sastavni dijelovi kako bi se mogli jednostavno reproducirati (open-source filozofija), kao i cjeloviti kod koji se planira učiniti javno dostupnim (pod odgovarajućim licencama) putem GitHub platforme[3] (opet, u skladu s open source filozofijom). Potrebno je napomenuti da su svi moduli detaljno testirani u laboratorijskim uvjetima sa ciljanim tipovima boca piva (kojima su i prilagođeni) dok u praktičnoj eksploataciji se mogu javiti neke dodatne neplanirane situacije koje će se trebati naknadno adresirati. Također, ukoliko se u proizvodnoj liniji promjeni tip boce, biti će potrebno promijeniti nekoliko parametara u programskom kodu, ali i prilagoditi nekoliko dijelova sklopova (npr. položaj senzora blizine). Svi kodovi su pisani u Python i Processing jeziku, tako da i korisnici sa manje iskustva mogu konfigurirati ključne postavke rada pojedinog podsustava jednostavnom izmjenom pojedinih parametara u zaglavlju priloženog koda. Naposljetku je potrebno istaknuti i činjenicu da je tijekom razvoja predloženih sustava testirano više različitih kamera, senzora i mikrokontrolera (u vidu brzine i pouzdanosti rada kao i jednostavnosti upotrebe i nadogradnje), te su predstavljene komponente one koju su naposljetku odabrane a koje ispunjavaju tražene kriterije naručitelja. Dio testiranja je izvršen na posebnom *testbench*u prikazanom na slici 2, a koja je trebala simulirati dijelove proizvodnog procesa u kojima imamo rotaciju boca piva (kao složenijeg gibanja), zajedno sa zaklanjanima dijela konstrukcije (3D printani dijelovi). Također su korišteni i posebni umeci konusnog presjeka koji su osiguravali unaprijed poznati nagib boce u odnosu na podlogu.

[2] https://www.ros.org/
[3] https://github.com/



Slika 2 – Primjer jedne od testnih postavki korištenih pri izradi studije

Primjer preliminarnih testiranja i razmatranja drugih komponenti je dan u Tablici 1 kako bi čitatelj dobio bolji osjećaj za performanse drugog sklopovlja, te stoga nisu namijenjeni za direktnu usporedbu s konačnim sustavom. Također, smatramo da bi uključivanje potpunih preliminarnih rezultata u ovu studiju nepotrebno pomaklo fokus sa predloženog sustava i njegovih karakteristika, te stoga oni i nisu uključeni (već samo njihov sažetak kako slijedi).

Prilikom razvoja sustava za praćenje je testirano nekoliko kombinacija najpopularnijih kamera i mikro kontrolera koji se koriste u MV-u (*eng. Machine Vision*). Odluka da se odabere mikrokontroler na ne (ugradbeno) računalo je prvenstveno radi jednostavnosti izgradnje i održavanja sustava, u kojem velika brzina rada nije ključna značajka, već je naglasak stavljen na cijeni korištenja pojedinih komponenti i jednostavnosti održavanja. Zanimljiva je usporedba jednog od danas najpopularnijih „malih ugradbenih računala“ (koji je mnogima prvi izbor u razvoju jednostavnih rješenja strojnog vida) Raspberry Pi (RPi), sa OpenMV H7 mikrokontrolerom koji je prvenstveno optimiziran za strojni vid. Premda bi jednostavna usporedba nazivnih vrijednosti stala na stranu RPi platforme, testovi pokazuju malu razliku između OpenMV H7 kamere koja radi na taktu 480 MHz i Raspberry Pi 3 na taktu 900 MHz (*Coremark Performance* 2040 u odnosu na 2700)[4] [5]. Mikrokontroleri ovu činjenicu mogu zahvaliti iznimno brzoj sabirnici (3.84 GB/s) u usporedbi sa ARM11 / ARM-A53 mikroprocesorima koji se koriste u RPi zero/3 platformama (3.6 GB/s), te broju instrukcija koje obrađuju po taktu (približno 2.5 puta više).

U studiji se isprobalo i rješenje primjenom iznimno brzog mikrokontrolera baziranog na Cortex-M7 arhitekturi , Teensy 4.0, u kombinaciji sa popularnom serijom ArduCam kamera. Međutim, pokazalo se da je usko grlo spomenute kombinacije SPI sabirnica kamere koja radi na najvećoj frekvenciji od 8MHz, premda sam mikrokontroler može podržati znatno veće brzine iste sabirnice (a temeljeno na dosadašnjem iskustvu autora sa spomenutim

[4] https://www.eembc.org/coremark//index.php
[5] https://openmv.io/pages/faq Ovakve performanse ARM's Cortex-M7

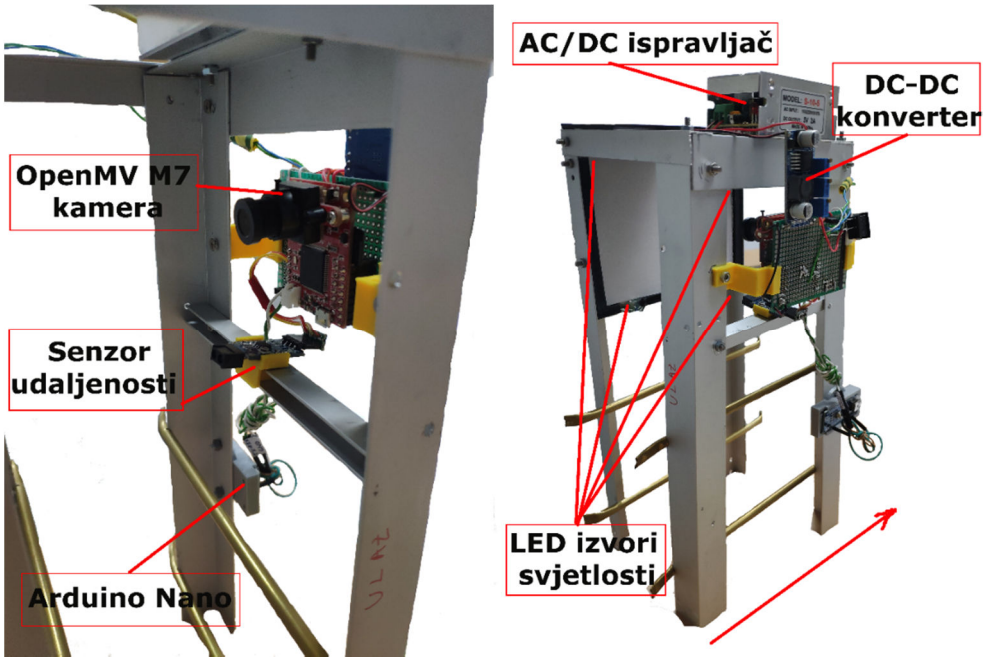
mikrokontrolerima). Kao primjer navodimo da je za očitavanje slike veličine 320 x 240 sa Omnivision2640 kamere u RGB565 formatu boja bilo potrebno ~ 200 ms, što znači da realizirani sustav nikako ne bi mogao raditi brže od 5-6 Hz. Usporedbe radi, ista brzina se postiže i očitavanjem slike iste veličine sa znatno sporijim i ograničenim Atmega328 mikrokontrolerom koji se temelji na AVR arhitekturi. Iz navedenog se može zaključiti da ARM-M7 bazirani mirkokontroleri imaju potencijala, međutim nedostupnost cjenovno prihvatljive odvojene kamere s traženim performansama, čini ovu kombinaciju za sada neizvedivom. Međutim, OpenMV projekt temeljen na ARM-M7 arhitekturi i sa ugrađenom kamerom premošćuje upravo spomenuti problem, i čini ga idealnim kandidatom u opisanom projektu. U nastavku je priložena tablica 1 koja ukratko navodi prednosti i mane, te performanse testiranih kombinacija kamere i mikrokontrolera.

Tablica 1 – Usporedba karakteristika komponenti razmatranih za upotrebu u razvijenim modulima

Mikrokontroler / kamera	**Arhitektura / takt**	**Količina RAM**	**Brzina akvizicija i jednostavna obrada slike rezolucije 320 x 240**	**Prednost**	**Mana**
Arduino UNO + OV2640	AVR / 16 MHz	2KB	5 Hz	Cijena.	Zbog sporosti sabirnice neupotrebljiv za CV. Ograničen RAM.
Teensy 4.0 + OV2640	ARM Cortex-M7 / 600 MHz	1024KB	6.5 Hz	Cijena. Sirova brzina mikrokontrolera.	Ne postoji kompatibilna brza SPI kamera. Programiranje u C / Processingu.
OpenMV H7	ARM Cortex-M7 /480 MHz	1024 KB	60 Hz	Brzina. Prilagođen MV-u. Jednostavno programiranje (Python).	Ograničenje RAM-a (memorija dovoljna za tek nekoliko slika).
Raspberry Pi 3 ArduCam + OV5647	Cortex-A53 900 - 1.4 GHz	1 GB	90 Hz	Cijena. Početno podešavanje i instalacija.	Brz. Dovoljno RAM-a za snimanje dulje sekvence slika u memoriju.

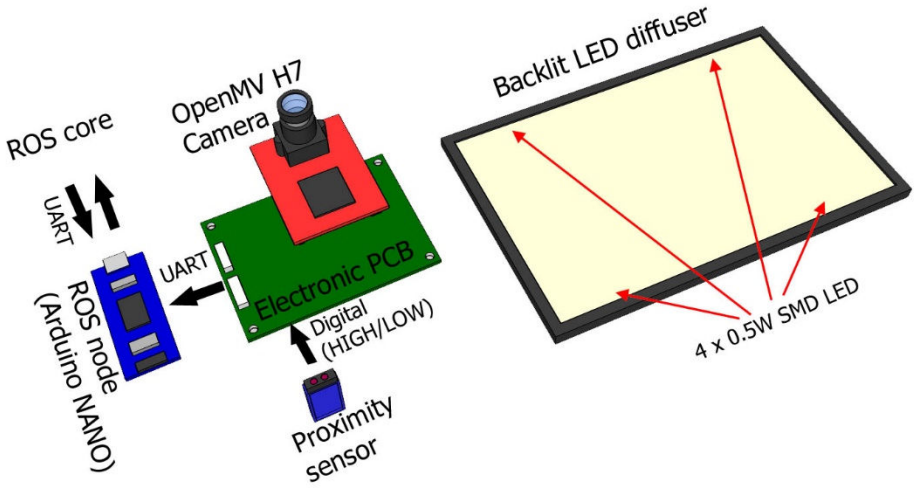
2. MODUL ZA DETEKCIJU ČEPOVA

Prototip dijela za detekciju čepova je prikazan na Slici 3, a njegovo planirano mjesto na proizvodnoj liniji (Slika 1) je nakon punilice i čepilice prilikom izlaska iz OPINAU-FILLER sustava. Potrebno je primijetiti da prototip sadrži i neke konstrukcijske i nosive dijelove (aluminijski profili i 3D printani dijelovi) koji neće nužno biti dio prototipa na proizvodnoj liniji budući da će se sustav postavljati na već postojeće konstrukcije.



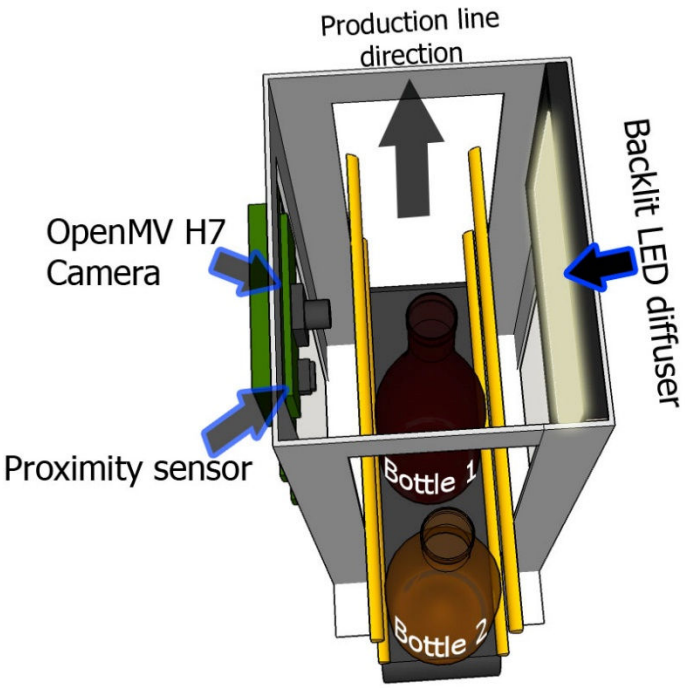
Slika 3 – Prototip modula za detekciju čepova na boci s označenim osnovnim dijelovima i smjerom kretanja boce (strelica na desnoj slici)

Tri glavna dijela razvijenog sustava su OpenMV7 kamera, senzor udaljenosti/blizine i LED izvori svjetlosti s odgovarajućim difuzorom svijetlosti. Osim toga sustav posjeduje dodatni mikrokontroler i elektroničku pločicu koja povezuje sve navedene komponente, što je detaljnije prikazano na slici 4. Difuzor svijetlosti osigurava jednolično osvjetljenje u pozadine boce.



Slika 4 – Funkcionalne komponente modula za detekciju čepova na boci

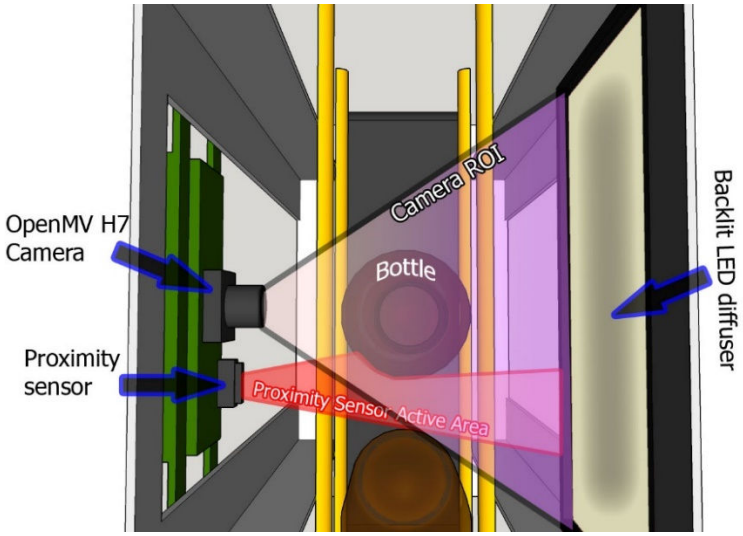
Difuzor se sastoji od 4 SMD LED postavljene na rubovima stakloplastike gdje je jedna strana izbrušena (matirana). Okidanje LED-ice stvara jednoličnо i snažno osvjetljenu pozadinu, što omogućava kameri da stvori upotrebljivu sliku u iznimno kratkom periodu, što neutralizira posljedice kretanja objekta. Okidanje se vrši korištenjem MOSFET-a IRFZ44N kao sklopke, gdje se propušta stabilizirani napon od 3.6V (doveden sa naponskog regulatora) prema napajanju LED-ica. Vizualizacija cjelokupnog sustava u radnim uvjetima je prikazan na slici 5. Primijetite da se kamera i senzor blizine nalaze sa jedne strane snimanog objekta (boca), dok se difuzor nalazi sa suprotne strane objekta. Također je važno da se vodilicama objekt zadrži / ograniči unutar linije kretanja za koje je cijeli sustav optimiziran.



Slika 5 – Vizualizacija prototipa modula za detekciju čepova na boci (u radnim uvjetima) s označenim osnovnim dijelovima

Konceptualno sustav funkcionira na slijedeći način (pogledajte sliku 5) : boca se kreće kroz proizvodnu liniju u točno predviđenom smjeru i prihvatljivom brzinom (u trenutnoj verziji programske podrške ali i mogućnosti raspoloživog sklopovlja, kameri je potrebno približno 300 ms da obradi jednu sliku i na izlazu generira informaciju o statusu čepa). U trenutku nailaska na područje detekcije senzora blizine, isti okida i šalje odgovarajući signal LED izvorima svjetlosti koji se nakratko pale (kao blic kod kamera) a istovremeno kamera snima svoje radno područje. Treba napomenuti da se obrada vrši samo kada senzor blizine detektira postojanje objekta na traci prvi put, dok se provjera postojanja boce / objekta odvija znatno većom brzinom i u mogućnosti je detektirati i jako brze objekte koji dođu u radno područje kamere. U slučaju da se na traci nalazi više boca, jedna iza druge, tada bi trebalo uzeti u obzir da period kada sljedeća boca dođe u radno područje ne bude kraće od 300 ms. Područje rada kamere, kao i područje detekcije senzora blizine je vizualizirano na slici 6. Vrijeme potrebno kameri da izvrši snimanje je oko 0.4 ms (po potrebi se može promijeniti unutar programskog koda), što je dovoljno brzo za neutralizaciju efekta kretanja snimanog objekta. Samo vrijeme paljenja

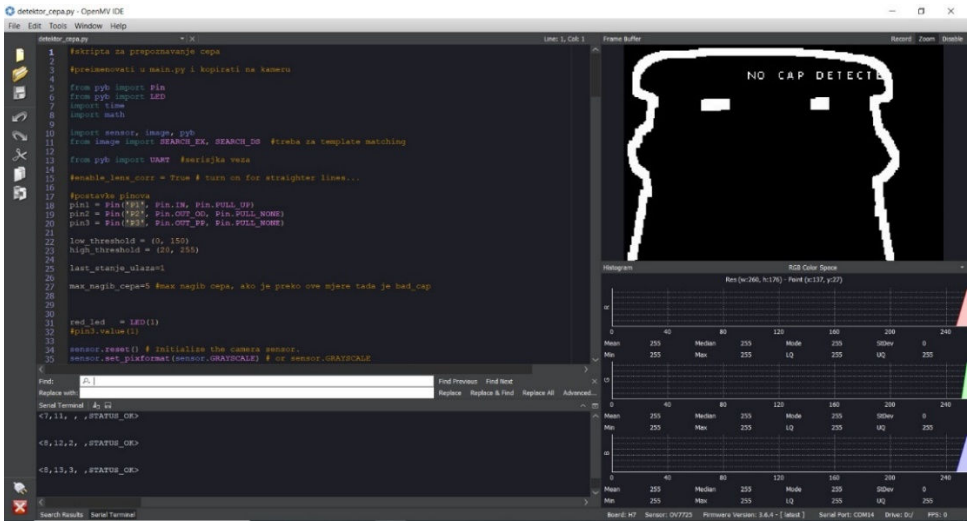
LED-ice je oko 10 ms, tako da slika do memorije računala dolazi za ~20 ms od trenutka kada je senzor blizine detektirao postojanje objekta.



Slika 6 – Označeno područje rada kamere (Camera ROI) i područje okidanja / detekcije senzora udaljenosti

Treba napomenuti i da je od kritične važnosti da kamera i senzor udaljenosti/blizine budu točno pozicionirani jedan u odnosi na drugog odnosno s obzirom na nadolazeću bocu. Naime, kamera je postavljena vrlo blizu bocama (2-3 cm, zbog što bolje rezolucije slike) te je potrebno precizno okidanje kako bi bocu slikala unutar područja pogleda a ne da dio boce bude izvan slike.

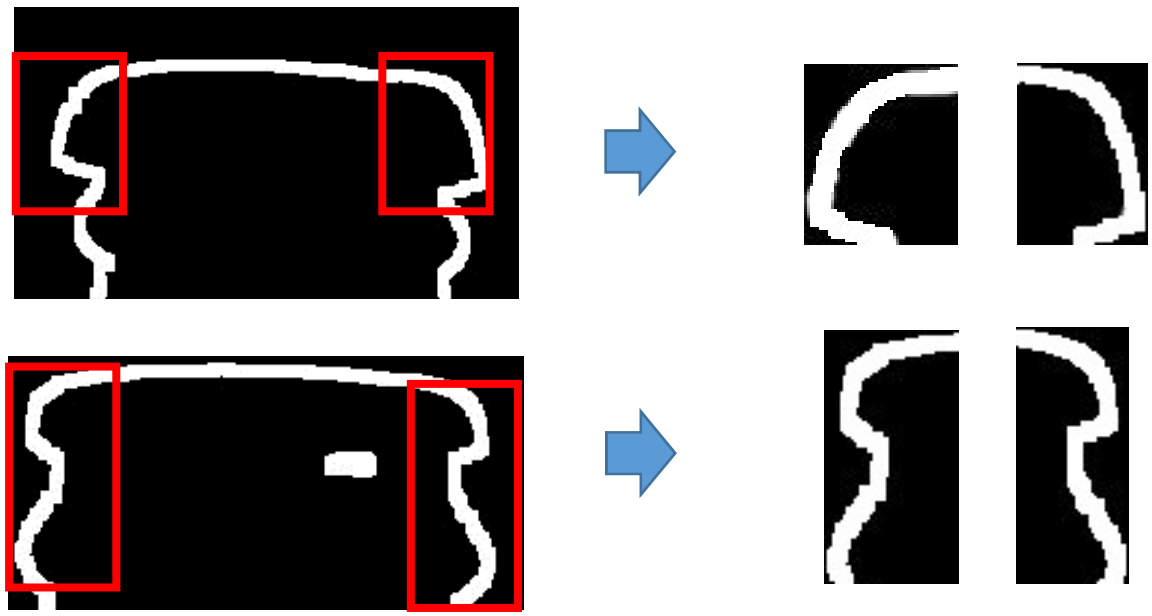
Na ovo je potrebno obratiti posebnu pažnju prilikom ugradnje u proizvodnu liniju. Tipičan izgled jedne snimke je vidljiv na slici 7 zajedno s razvojnim okruženjem (unutar Python programskog jezika) koji se isporučuje sa odabranom kamerom. Zanimljivo je napomenuti da je kamera posebno namijenjena strojnom vidu pa dolazi s nizom korisnih, već definiranih funkcija, a što pak posljedično olakšava eventualne nadogradnje u budućnosti (bilo od proizvođača ili samog krajnjeg korisnika).



Slika 7 – Izgled razvojnog okruženja OpenMV M7 kamere

Jednom kada je slika dohvaćena od strane kamere ona se na njoj i obrađuje (dakle kamera ima svoj mikrokontroler). Budući da sam mikrokontroler ima ograničene resurse, ali i zbog želje za jednostavnošću i pouzdanosti pristupa, detekcija čepa se temelji na inačici podudaranja s predloškom (*eng. template matching*). Detalji implementacije su dostupni u programskom kodu koji se naručitelju isporučuje u elektronskom obliku uz ovo Izvješće. Primjeri uzoraka sa i bez čepa (i to desna i lijeva strana) su prikazani na Slici 8. Bitno je primijetiti da sustav detektira samo pojedine dijelove boce koji su ključni za razlikovanje ima li boca čep ili nema, a ne cjelovitu bocu.

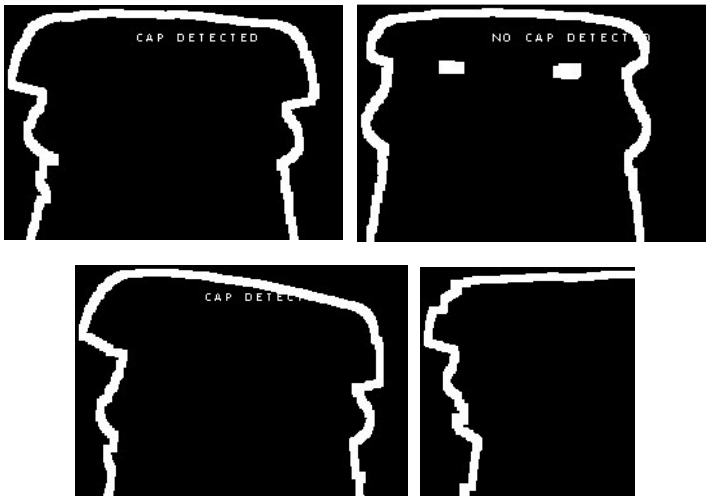
Iz slike 9 kao i same činjenice da se koristi podudaranje s uzorcima, može se zaključiti da pristup (tj. uzorci) ovise o vrsti/tipu boce koja se koristi. Stoga, ukoliko se tijekom eksploatacije promjeni boca koja se upotrebljava (osim već spomenute korekcije u pozicioniranja senzora udaljenosti/blizine i kamere) potrebno je ažurirati i bazu uzoraka po kojoj sustav vrši prepoznavanje. Primjeri nekoliko različitih vrsta detekcije su prikazani na slici 6. Sustav je laboratorijski testiran na nekoliko standardnih boca i čepova, i u mogućnosti je sve detektirati i razlikovati.



Slika 8 – Slika detektiranog ruba boce (lijevo) i izgled predložaka za ciljanu bocu sa i bez čepa (lijevi i desni brid boce) (desno)

Nakon što obradi snimljenu sliku, mikrokontroler kamere (ovisno o rezultatima) generira poruku/string koji se šalje na Arduino Nano (putem UART-a) na koju se nalazi ROS čvor i koji je u principu zadužen za daljinu komunikaciju s OPINAU-FILLER sustavom i njegovim komponentama. Ovakva arhitektura je odabrana jer se time rasterećuje mikronkontroler kamere (koji ne treba pokretati ROS čvor) a omogućeno je da se putem USB veze/kabela koja je dostupna na svim Arduino razvojni pločicama može jednostavno povezati s ROS Core dijelom sustava. Ovdje je potrebno napomenuti da iako sustav u laboratorijskim uvjetima je dobro funkcionirao sa Arduino Nano pločicom, zbog činjenice da s trenutnim kodom je

iskorišteno 86% njegove radne memorije (od ukupno 2kB), postoji mogućnost potencijalno nestabilnog rada tijekom eksploatacije. Stoga se predlaže zamijeniti navedeni mikrokontroler sa Mega 2560 PRO inačicom Arduino mikrokontrolerske pločice koju karakterizira značajno veća radna memorija (8 kB). Dodatna pogodnost upotrebe ovakvog mikrokontrolera je postojanje sklopovske serijske veze (UART) koja se može koristiti za komunikaciju s kamerom (trenutno se koristi programska inačica serijske veze putem *SoftSerial* biblioteke, budući da Arduino Nano ima samo jedan sklopovski serijski port odnosno Tx/Rx pinove, a koji su zauzeti za komunikaciju s drugim dijelovima sustava).



Slika 9 – Nekoliko primjera detekcije sa i bez čepa, kao i slučaj kada je boca izašla iz područja interesa

Format poruke koju mikrokontroler kamere šalje Arduino mikrokontroleru (putem UART veze) je *<B, D, S, K, P>* gdje je:

- < i > su znakovi početka i kraja poruke koja se šalje, a koja pomaže kod parsiranja primljene poruke na strani Arduina,
- *B* je broj detekcije boce od trenutka uključenja mikrokontrolera. Ova varijabla je cjelobrojna vrijednost tipa *Long* čime se osigurava maksimalna vrijednost od 2.147.483.647, a što smatramo da je više nego dovoljno za male, craft pivovare i interval paljenja/gašenja proizvodne linije.
- *D* je ukupni broj detekcija sustava od trenutka uključenja mikrokontrolera. Naime, sustav je sposoban zabilježiti i detekciju koja ne odgovara uzorku boce (sa ili bez čepa) te o tome obavijestiti korisnika. Vrijednost varijable *D* je uvijek veća ili jednaka vrijednosti varijable *B*.
- *S* je status trenutno detektiranog čepa/boce. Stanja si kodirana s pet i*nteger* vrijednosti i to kako slijedi: 0 – nije detektirana boca, 1 – detektirana je boca s čepom, 2 – nije detektirana boca s čepom, 3 – detektirana je boca s čepom ali čep nije ispravno postavljen, i 4 – boca nije dobro snimljena odnosno je izvan regije od interesa (ROI).
- *K* je kut nagiba vrha čepa u odnosu na horizontalu. Naime, ukoliko je detektirana boca s čepom (status 1) onda se detektiraju dvije krajnje gornje točke na čepu s lijeve i desne

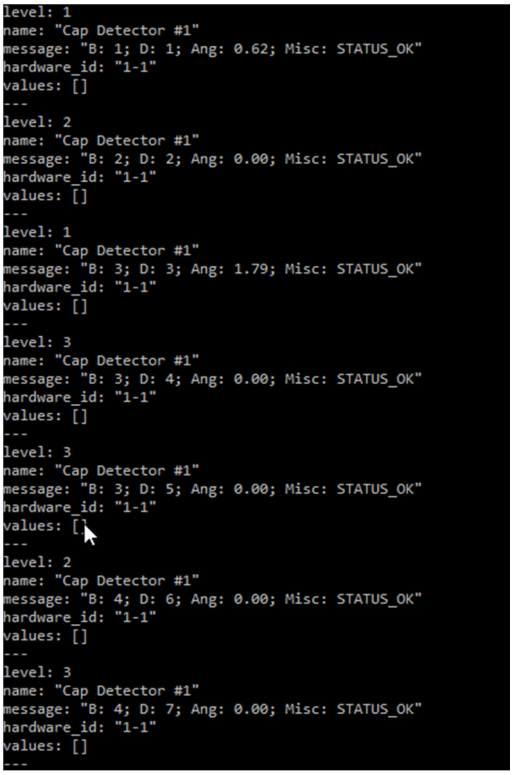
strane, te se povlači pravac kroz njih, nakon čega se računa nagib toga pravca u odnosu na horizontalu (jednostavna trigonometrijska operacija). Ukoliko je kut veći od nekog unaprijed definiranog praga (u programskom kodu, npr. 7 stupnjeva) detektira se slabo ili krivo postavljen čep i postavlja se status 3. Potrebno je napomenuti da se ovaj kut ne računa u slučajevima kada su detektirani drugi statusi čepa osim 1. U tome slučaju za ovo polje se šalje prazan podatak koji je definiran u programskom kodu.

- *P* je varijabla tipa *string* (tekstualna varijabla). U trenutnoj arhitekturi nema konkretnu funkciju (šalje se poruka „STATUS_OK“) već je ostavljena za buduće namjene koje korisnik može imati.

Jednom kada poruka sa mikrokontrolera kamere stigne do Arduino Nano tamo se parsira na način da se prvo traže znakovi za početak i kraj poruke, nakon čega se pojedine vrijednosti (koje su odvojene zarezom, tako da je to korišteni delimiter) spremaju u odgovarajuće varijable. Ovdje valjda napomenuti da se zbog već prije spomenutog ograničenja radne memorije Nano mikrokontrolera za varijable *B* i *D* nije upotrebljavao *long* već *integer* tip varijable (vrijednosti do 32.767). Međutim, ovo je nešto što se može brzo i jednostavno promijeniti jednom kada se uključi drugi, preporučeni mikrokontroler.

Nakon parsiranja, podaci se pakiraju u odgovarajuću ROS poruku. Za prototip je odabrana poruka tipa *DiagnosticStatus* iz *diagnostic_msgs* standardnog ROS paketa. Definicija svih polja poruke je dostupna na http://docs.ros.org/api/diagnostic_msgs/html/msg/DiagnosticStatus.html. Ova poruka je odabrana jer ima sva polja koja su iskoristiva s obzirom na podatke koji su dostupne te je dio svake standardne ROS distribucije. Međutim, naručitelje je u kasnijim inačicama sustava i programske podrške slobodan definirati i svoju poruku koja pobliže prati strukturu i tipove podataka koji se koriste kod detektora čepa. Upravo odvajanje ROS funkcionalnosti na drugi mikrokontroler omogućava da korisnik mijenja tip poruke bez da se utječe na funkcionalist drugih podsustava.

Nakon toga se na mikrokontroleru pokreće izdavača za temu */CapStatus (*primjenom *rosserial* paketa*)* na koju se objavljuju (putem USB veze) statusi detekcije čepa. Statusi se ne objavljuju neprestano (kako bi se uštedjeli resursi sustava) već samo u trenutcima kada sustav izvrši novu detekciju. Primjeri nekoliko poruka za različite detekcije je moguće vidjeti na slici 10. Ovdje se može primijetiti da polje *DiagnosticStatus* poruke *level* sadrži varijablu *S* dok su ostale varijable sadržane u polju *message* (i u obliku stringa su). Optimalniji način distribucije podataka u polju *message* bi bio korištenjem *values* polja koje prima varijablu tipa niz od varijabli *KeyValue* (ROS poruka). Međutim zbog problema sa serializatorom na strani Arduina i kratkog vremena trajanja projekta problem prijenosa informacija je riješen spremanjem svih dodatnih informacija (osim statusa) u *message* polje. Proširenje na *values* polje je svakako jedno od područja za buduća unaprijeđenja sustava.



Slika 10 – Primjer ispisa poruka o detekciji na strani računala s ROS-om

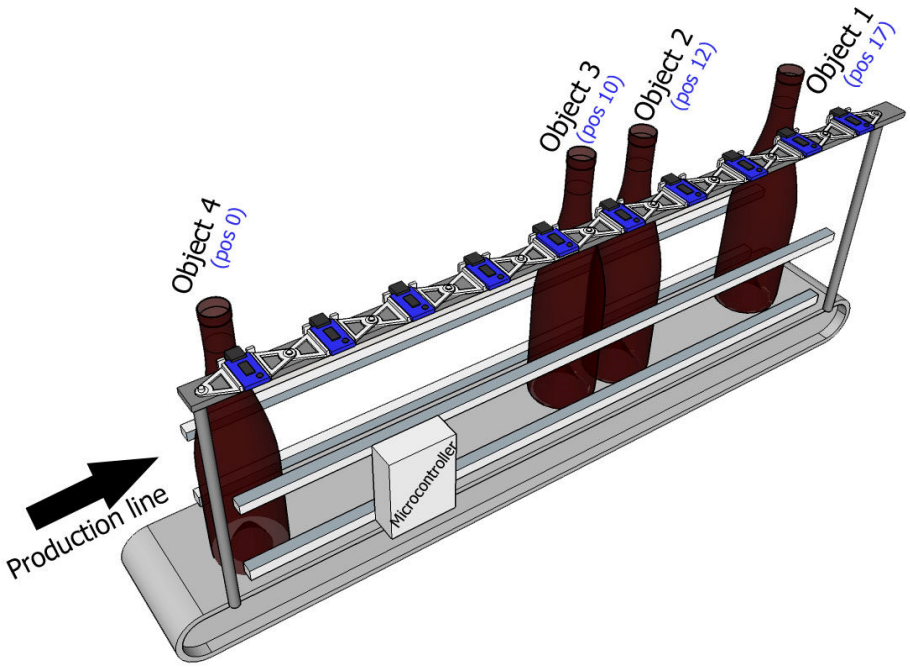
Naposljetku, navodimo tablicu (Tablica 2) materijala korištenih pri izradi laboratorijskog prototipa, pri čemu ne navodimo količinu za konstrukcijske/potporne materijale budući da oni neće biti potrebni (ne barem u tome obliku i obujmu na stvarnoj proizvodnoj liniji).

Tablica 2 – Najvažnije komponente korištene pri izradi Modula 1

Komponenta	Količina (jedinica)
OpenMV Cam H7 Image Sensor	1 (kom)
Distance sensor, reflective 3.3V/5V – 30 mm – Waveshare 9523	1 (kom)
AC 110-240V to DC 5V switching power supply converter SA10-05	1 (kom)
LM2596S DC-DC Constant Current Module Step-down Adjustable CC/CV Power Supply Module	1 (kom)
Ultra Bright 3528 LED SMD White Chip Surface Mount 20mA 7-8LM	4 (kom)
Difuzna pozadina (stakloplastika 4mm, izbrušena brusnim papirom)	1 (kom)
IRFZ44N N-kanalni MOSFET	1 (kom)
Arduino Nano	1 (kom)
Potrošni materija (tiskana pločica, aluminijski profili, 3D printani dijelovi)	/

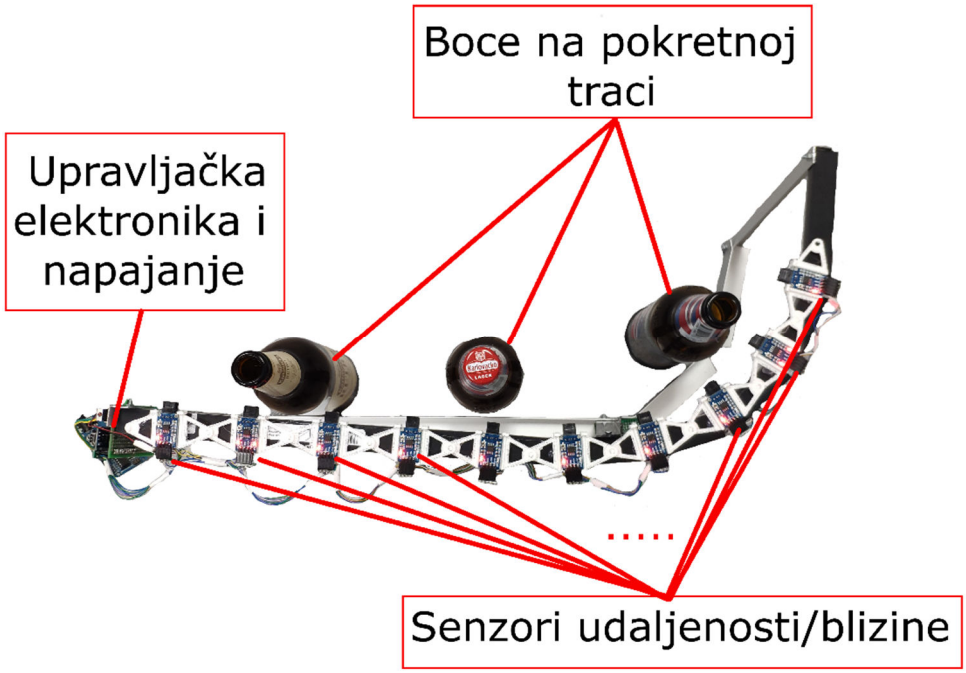
3. MODUL ZA PRAĆENJE KRETANJA BOCA

Vizualizacija sustava za praćenje položaja boca na proizvodnoj linije je prikazana na slici 11, dok je laboratorijski prototip istog modula prikazan na slici 12. Njegova svrha je omogućiti krajnjem korisniku kontinuirano praćenje stanja objekata na proizvodnoj liniji, a njegov položaj unutar OPINAU-FILLER sustava (Slika 1) nije strogo određen, odnosno može se koristiti na svim njezinim dijelovima (s eventualno različitom gustoćom senzora, a ovisno o željenoj rezoluciji određivanja položaja pojedine boce).



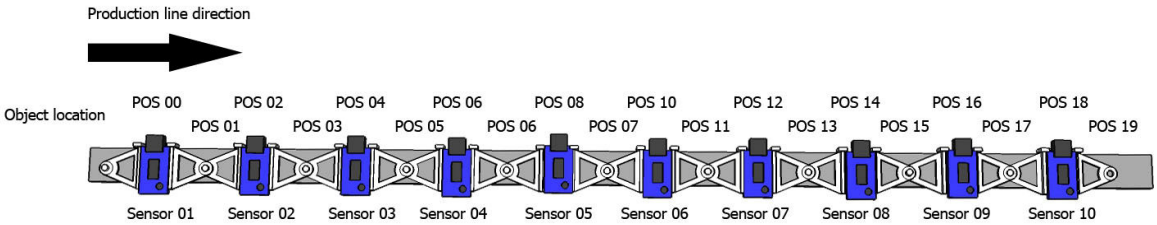
Slika 11 – Vizualizacija sustava za praćenje položaja boca na proizvodnoj liniji

Korisnik može odabrati gušću raspored (pri čemu je nekakav minimum razmaka među dva senzora udaljenosti/blizine oko 6 cm, koliko iznosi promjer jedne boce; kod pozicioniranja senzora, a budući da se radi o aktivnim senzorima, potrebno je obratiti pažnju na potencijalnu međusobnu interferenciju) za dijelove sustava gdje mu je potrebna točnija (i češća) informacija o položaju boca (kao npr. kod punilice i čepilice) dok za dijelove sustava gdje se boce samo transportiraju pokretnom trakom (kao npr. pri prijelazu iz dijela za lijepljenje naljepnica u punilicu) taj raspored može biti nešto rjeđi. Potrebno je primijetiti da prototip sadrži i neke konstrukcijske i nosive dijelove (aluminijski profili i 3D printani dijelovi) koji neće nužno biti dio prototipa na proizvodnoj liniji budući da će se sustav postavljati na već postojeće konstrukcije. Dva glavna dijela razvijenog sustava su: senzor udaljenosti/blizine (Waveshare 9523 u našem slučaju, ali moguće je upotrijebiti i druge, slične tipove senzora) i mikrokontroler koji obrađuje i interpretira podatke, te ih pakira i šalje u obliku pogodnom za ROS sustave.



Slika 12 – Pogled odozgo na modul za praćenje položaja boca na proizvodnoj liniji

Konceptualno sustav sklopovski funkcionira vrlo jednostavno: senzori udaljenosti/blizine su postavljeni duž trake/linije kojom se kreću boce piva (pune ili prazne) na odgovarajućoj visini (zbog potencijalnog prisustva naljepnice) detektiraju blizinu boce putem detekcije reflektirane IR zrake koji sami stvaraju (osjetljivost/udaljenost se može prilagoditi trimerom na samoj senzorskoj pločici). Raspored senzora u sustavu za praćenje objekata, kao i mogući položaji objekta unutar sustava su vizualizirani na slici 13. U ovom primjeru je prikazana najveća gustoća postavljanja senzora (6 cm), gdje je za njihovo točno postavljanje izrađen / 3D printan nosač koji omogućava povezivanje više nosača sa senzorima u seriju (tako da „nos“ jednog ulazi u „rupu“ drugog nosača). 3D model nosača je dostupan u elektroničkom formatu (STL datoteka) u priloženoj dokumentaciji sustava. Ovako izrađeni i spojeni nosači omogućavaju postavljanje senzora pod željenim kutom, tako da se mogu pratiti objekti i u zakrivljenim putanjama unutar sustava, kao što je prikazanu u izrađenom prototipu (slika 12).



Slika 13 – Raspored senzora i mogući položaji objekta na modulu za praćenje položaja boca na proizvodnoj liniji

Detekcijom boce/objekta u blizini, senzor na svoje izlazu generira digitalni izlazni signal kojim signalizira svoje stanje detekcije. Izlazi svih senzora (u našem slučaju 10 senzora što

omogućuje ukupnu rezoluciju od 20 položaja, računajući i položaje među senzorima) su spojeni na Arduino Nano (ili bilo koji drugi) mikrokontroler. Zadatak ovog mikrokontrolera je da prikupi podatke sa svih njemu dostupnih senzora (zbog ograničenog broja pinova Arduino Nano mikrokontrolera, kod većeg broja senzora udaljenosti/blizine trebalo bi koristiti više mikrokontrolera, ili mikrokontrolere sa više dostupnih ulaznih linija) i proslijedi ga idućem mikrokontroleru putem serijske veze. Odabran je Arduino Nano mikrokontroler kako se radi o najdostupnijem mikrokontroleru koji omogućava kontinuiranu provjeru stanja na deset digitalnih ulaza, i slanje statusa preko hardverskog UART sučelja prema drugom računalu. Nikakva druga funkcionalnost (kao primjerice *SoftSerial* ili ROS node) nije nadodana na ovaj mikrokontroler kako bi se omogućio fluentni rad sustava za praćenje.

Ovakva arhitektura se koristi kako bi se osiguralo rasterećenje prvog mikrokontrolera tako da može raditi na što većoj frekvenciji (trenutno može pratiti boce s teoretskom frekvencijom od 100 Hz, što bi od razmak između senzora od 6 cm značilo da je teoretski maksimalno dopuštena brzina boce/trake 6 m/s – naravno stvarne brzine su nešto manje zbog vremena potrebnog za obradu, prijenos podataka i slično). Mikrokontroler također obavlja elementarnu detekciju ključnih događaja promatranjem sekvenci senzora, a temelji se na pretpostavki da boca koja se kreće proizvodnom linijom ne može nestati niti se pojaviti na bilo kojem dijelu linije osim na početku odnosno kraju iste. Temeljem toga sustav može detektirati da li je neka boca zapela/nestala, da li se neka boca kreće unatrag, da li je neka boca pala i slično. Ovdje valja istaknuti da se složenije zaključivanje (npr. uzimajući u obzir i vremensku komponentu generiranih sekvenci) nije radilo na mikrokontroleru, zbog već prije spomenute uštede vremena i resursa, ali su osigurani svi podaci (npr. timestamp) da se isto može obaviti na strani računala odnosno ROS-a.

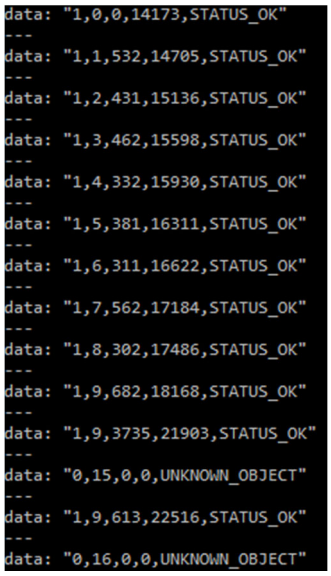
Oblik poruke koji prvi mikrokontoler šalje drugom Arduino Nano mikrokontroleru putem serijske veze je *<ID, Pos, TC, TS, P>* gdje je

- < i > su znakovi početka i kraja poruke koja se šalje, a koja pomaže kod parsiranja primljene poruke na strani drugog Arduina.
- *ID* je jedinstveni broj boce/događaja koji se dodjeljuje kod aktivacije prvog senzora. Ova varijabla je cjelobrojna vrijednost tipa *long* čime se osigurava maksimalna vrijednost od 2.147.483.647, a što smatramo da je dovoljno za male pivovare.
- *Pos* varijabla je varijabla tipa interes i predstavlja jedinstveni kod položaja boce na traci odnosno unutar niza senzora udaljenosti. Zbog postizanja bolje rezolucije mjesta između dva senzora također imaju ovaj jedinstveni kut. U trenutnoj konfiguraciji od 10 senzora, ova varijabla ima 20 razina (odnosno poprima vrijednost iz intervala <0,19>). Potrebno je napomenuti da je za većinu proizvodnih linija moguće ovu varijablu deklarirati da bute tipa *byte* (vrijednosti u intervalu od 0 do 255) a zbog optimizacije upotrebe radne memorije.
- *TC* je vremenska varijabla tipa *long*, a koja predstavlja razliku dva timestampa za posljednja dva događaja (sadašnjeg i prethodno zapisanog) boce iste vrijednosti varijable *ID*. Iako bi se ova varijabla mogla izračunati iz poznavanja *TS* i *ID* boce, njezinim uključivanjem u poruku cilj je bio olakšati krajnjem korisniku da jednostavno i transparentno može procijeniti brzinu kretanja boce/objekta.

- *TS* je varijabla (tipa *long*) koji predstavlja vremenski pečat (timestamp) trenutnog događaja, a izražena je u broju milisekundi od trenutka uključenja mikrokontrolera.
- *P* je varijabla tipa *string* (tekstualna varijabla) a sadrži kratku poruku o statusu sustava koji se dobije jednostavnom logičkom analizom generiranih sekvenci senzora. Temeljem nje korisnik može brzo donijeti odluku o potrebnim akcijama u proizvodnoj liniji, a može je i zanemariti te temeljem dostupnih preostalih podataka provesti svoju analizu sekvenci. U trenutnoj inačici modula, sama poruka može biti jedna od četiri predefinirane: STATUS_OK, STATUS_ERROR_OVERFLOW, UNKNOWN_OBJECT i OBJECT_RETURNED?. Njihovo trenutno značenje je kako slijedi:
    - STATUS_OK – označava da su sve sekvence u skladu s očekivanjima.
    - STATUS_ERROR_OVERFLOW – označava da je boca koja ima viši *ID* „prešla preko“ boce koja ima niži *ID* (odnosno ušla je u praćenje prije). Ovakva situacija se može javiti ako se boca s nižim *ID*-em ukloni s linije odnosno ako neki od senzora ne funkcioniraju kako treba.
    - UNKNOWN_OBJECT – označava da se neki objekt/boca pojavila a da nije prošla cijelu sekvencu iz početka. Ovakva situacije se može npr. javiti ako netko stavi bocu naknadno na proizvodnu liniju ili ako neki od prvih senzora ne rade kako treba.
    - OBJECT_RETURNED? – ova poruka je vezana samo za posljednji senzor udaljenosti/blizine u nizu i to u situacijama kada se njegovo stanje više puta upali/ugasi. Ovakva situacija se može javiti ako boca se lagano zaljulja ali i u ozbiljnijim slučajevima ako je došlo do nekakvog grupiranja i gomilanja boca na proizvodnoj liniji. Slične situacije se na prethodnim senzorskim mjestima posebno ne označavaju budući da ih zbog okolnih senzora možemo detaljnije u vremenu popratiti i donijeti zaključak. Međutim na posljednjem mjestu u nizu nema više senzora i ne možemo biti sigurni što se događa pa zato podižemo ovo poruku (stoga bi se ova poruka mogla više smatrati kao upozorenje a ne pogreška).

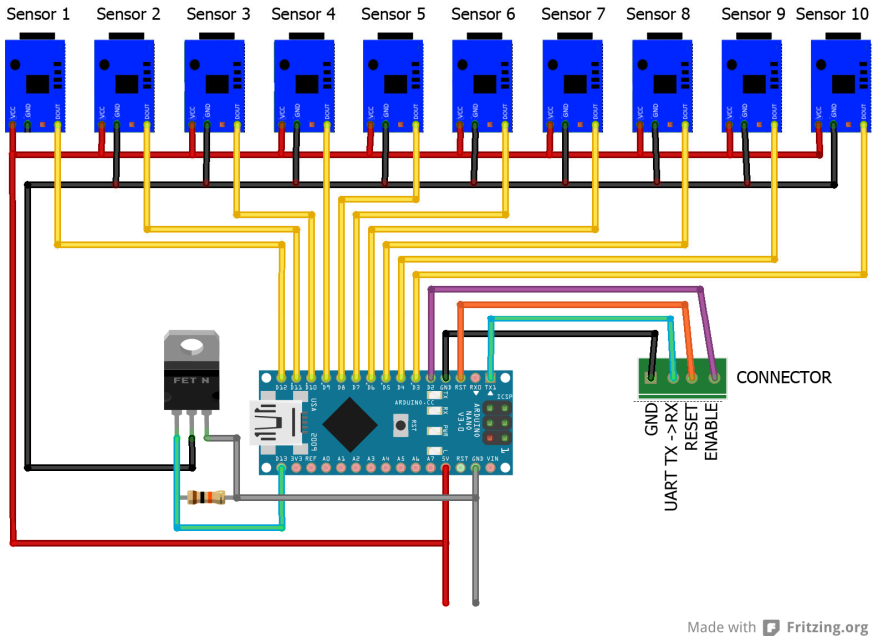
Opisanu poruke se zatim primaju na drugom Arduino Nano mikrokontroleru koji je dio upravljačkog dijela modula. Zadatak drugog modula je osigurati ROS komatibilnu komunikaciju s ostatkom sustava. Zbog osiguranja/zadržavanja visoke brzine rada cjelokupnog sustava (barem u verziji prototipa) ovaj mikrokontroler ne vrši nikakvo parsiranje poruka i dodjeljivanje složenijim tipovima ROS poruka (posebno uzimajući u obzir činjenicu da je Naručitelj izrazio namjeru da sam kreira ROS poruke za OPINAU-FILLER sustav), već se samo uklanjaju znakovi početka i kraja poruke (tj. < i >), te se poruka kao *string* varijabla objavljuje na ROS temu naziva */bottlePosition (*primjenom *rosserial* paketa*).* Krajnji korisnik zatim može na računalu (brže) parsirati poruku i eventualno upotrijebiti složenije algoritme analize uključujući i postupke strojnog učenja. Statusi se ne objavljuju neprestano (kako bi se uštedjeli resursi sustava) već samo u trenutcima kada sustav izvrši novu detekciju. Potrebno je napomenuti da se na ovom Arduino Nano mikrokontroleru pokreće i jedan pretplatnik i to na temu */bottlePositionReset* (također primjenom *rosserial* paketa*).* Tema očekuje prazne poruke (*std_msgs/Empty)*, te u trenutku kada je programski kod na Arduinu detektira odnosno primi,

pokreće funkciju koja podiže stanje jednog (predefiniranoga) pina u stanje HIGH te nakon 0.5s u stanje LOW (tj. generira digitalnog impulsa) a što se detektira s prvim Arduino mikrokontrolerom te se vrijednosti svih varijabli resetiraju na 0. Sličnu stvar je moguće realizirati i na način da se prvi Arduino u potpunosti resetira isključivanjem napona (upotreba *Reset* pina se pokazala kao nepouzdano rješenje zbog parazitnih napona prisutnih na liniji za komunikaciju dvaju Arduina). Primjer ROS poruka na strani računala je prikazan na slici 14. U primjeru je jedna boca krenula iz početka linije prototipa i došla je do mjesta (*Pos*) 9 u kojemu je pomaknuta sa linije i stavljena na položaj (*Pos*) 15, gdje se nastavila kretati prema kraju linije. Iz slike je vidljivo da je taj novi događaj na položaju 15 sustav detektirao kao nepoznati objekt (jer se pojavio van predviđene sekvence), dok je položaj boce 9 i dalje zapamtio (odnosno sustav i dalje smatra da se tamo potencijalno nalazi neka boca). Ovo mjesto će sustav zapamtiti dok ga druga boca (sa većim *ID*-om) ne „prebriše“ (u kojem slučaju će sa dodatno javiti i STATUS_ERROR_OVERFLOW poruka). Ovakvo i slično ponašanje prototipa je testirano na velikom broju ponavljanja u laboratorijskim uvjetima s visokom pouzdanošću i ponovljivošću. Ukoliko naručitelj želi učitati i dodatnu funkcionalnost ili logiku tumačenja sekvenci na modul, to je moguće jednostavno i napraviti, a što je u skladu s *open-source* filozofijom cijelog projekta.



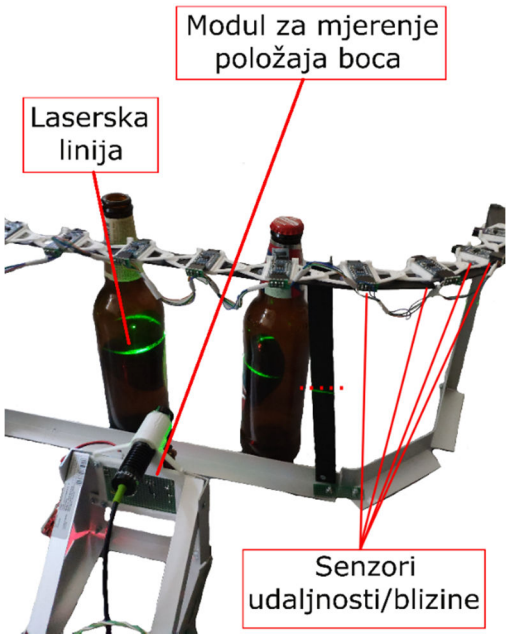
Slika 14 – Primjer ROS poruka (tipa *string*) primljenih na strani računala od drugog modula

Također je potrebno napomenuti da sustav ima na razini prvog Arduino mikrokontrolera previđenu funkcionalnost za odgovaranje isključivo na upit, odnosno sustav ne emitira stanja kada se dogodi promjena (neprestano), već samo u trenutcima kada to krajnji korisnik želi (i kada je naravno promjena prisutna). Na slici 15 se nalazi shema spajanja elektroničkih komponenti sustava za praćenje. Primijetite da svi senzori dijele istu liniju za napajanje, koja se kontrolira putem MOSFET-a odnosno digitalnog pina samog mikrokontrolera. Na elektroničkoj pločici se nalazi dodatni konektor koji omogućava povezivanje sa drugim računalom putem UART veze (Serial), te dodatne linije za reset mikrokontrolera i ENABLE linije koja sinkronizira rad više senzorskih modula (ako se takvi nalaze na liniji).



Slika 15 – Shema spajanja elektroničkih komponenti sustava za praćenje položaja

Ukoliko se ovakav modul koristi na osjetljivijim dijelovima sustava gdje je poznavanje točnog položaja boce (reda veličine milimetra) bitno (kao što je npr. punilica i čepilica), može se koristiti u kombinaciji s modulom 3 (opisanim u idućem poglavlju). Pokazni primjer upotrebe oba sustava istovremeno je prokazan na slici 16.



Slika 16 – Primjer istovremen upotrebe Modula2 i 3 za precizno praćenje položaja boce na osjetljivim dijelovima proizvodne linije

Naposljetku, kao i u prethodnom poglavlju, navodimo tablicu (Tablica 3) materijala korištenih pri izradi laboratorijskog prototipa, pri čemu ne navodimo količinu za konstrukcijske/potporne materijale budući da oni neće biti potrebni (ne barem u tome

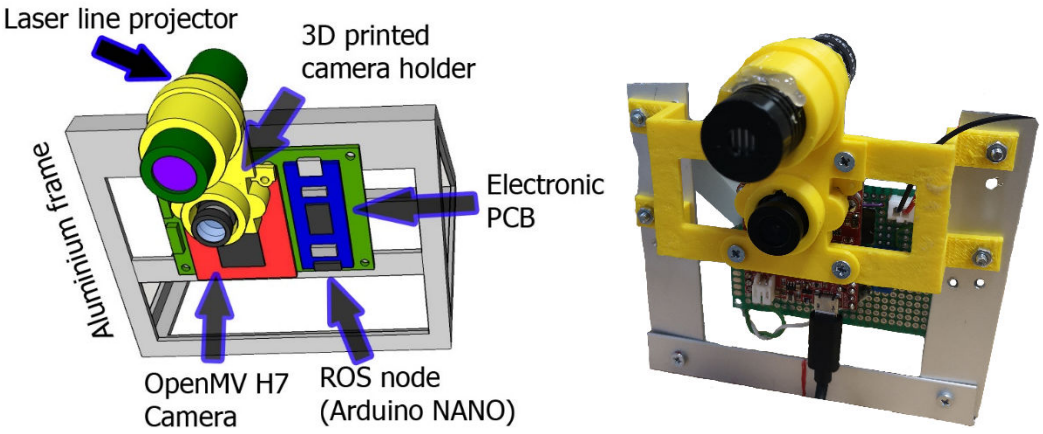
obliku i obujmu na stvarnoj proizvodnoj liniji). Napominjemo da je USB napajanje korišteno samo pri izradi i testiranju prototipa, dok se kod implementacije na proizvodnoj liniji očekuje dostupnost stabiliziranog izvora od 5 V (preporučeno ~500 mA).

Tablica 3 – Najvažnije komponente korištene pri izradi Modula 2

Komponenta	**Količina (jedinica)**
Distance sensor, reflective 3.3V/5V – 30 mm – Waveshare 9523	20 (kom)
DC-DC konverter s izlazom od 5V	1 (kom)
USB kabel za napajanje	1 (kom)
IRFZ44N N-kanalni MOSFET	1 (kom)
Arduino Nano	(kom)
Potrošni materija (tiskana pločica, aluminijski profili, 3D printani dijelovi)	/

4. MODUL ZA PRECIZNO PRAĆENJE POLOŽAJA BOCE

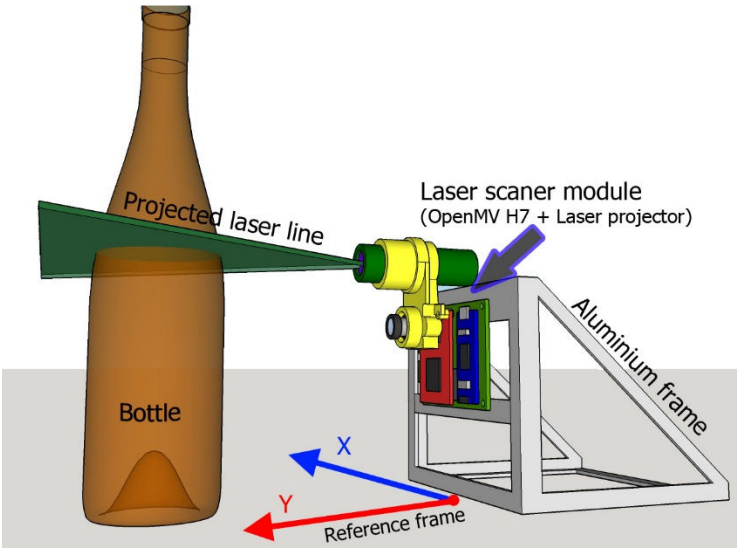
Namjena posljednjeg, trećeg podsustava je precizno praćenje položaja objekta / boce u ograničenom radnom području. On nadopunjava prethodno opisani sustav (Modul 2) koji se temelji na senzorima udaljenosti / blizine. Naime, matrica senzora blizine koji su međusobno postavljeni na međusobnim udaljenostima od ~6 cm omogućava u predloženoj verziji prostornu rezoluciju od ~3 cm (polovinu udaljenosti između objekata), što je prihvatljivo za praćenje objekata promjera ~6 cm. Međutim, u nekim kritičnim lokacijama linije za punjenje potrebno je poznavati položaj boce sa milimetarskom točnošću (primjerice kod podsustava punilice ili čepilice – Slika 1). Na tim mjestima se preporuča koristiti predloženi sustav koja omogućava lokalizaciju boce sa milimetarskom točnošću. Konceptualni i stvarni izgled sustava su prikazani na slici 17.



Slika 17 – Osnovne komponente sustava za precizno praćenje boca

Sustav se bazira na principu rada jednostavnog laserskog 3D skenera, odnosno projiciranjem laserske linije koja se reflektira od objekta, i upada natrag na leću kamere. Zbog vertikalnog pomaka kamere i lasera, na snimljenoj slici postoji disparitet laserske linije koji je vezan uz udaljenost snimanog objekta od kamere. Bliži objekti iskazuju manji disparitet, dok udaljeniji iskazuju veći disparitet. Sami sustav se sastoji od OpenMV H7 kamere i zelenog laserskog projektora snage 30 mW pri čemu su oboje postavljeni u posebno pripremljeno kućište (3D model kućišta je priložen kao *stl* datoteka u dokumentaciji). Upotreba kućišta osigurava fiksaciju njihovog međusobnog položaja, a što je od kritičkog značaja za točno funkcioniranje sustava. Dodatno, kamera je spojena na elektroničku pločicu koja sadrži konektore (napajanje, konektor prema laseru), MOSFET i dodatni Arduino Nano mikrokontroler koji sadrži ROS čvor za komunikaciju sa glavnim računalom. Sve nabrojane komponente su prikazane na slici 17. Važno je da tijekom korištenja ne dolazi do pomaka lasera u odnosu na kameru, jer je tada potrebno izvršiti rekalibraciju cijelog sustava. Kako se glavni dio sustava „izvršava“ na poprilično ograničenom mikrokontroleru (32bit ARM M7), neka ograničenja su trebala biti uzeta u obzir, odnosno izvršiti određenu optimizaciju. Primjerice, nastojalo se ograničiti broj matematičkih (trigonometrijskih) operacija koje on izvršava, tako da se potencirala izrada LUT (*eng. lookup table*), tablica koje već imaju izračunate vrijednosti za određene ulaze. Kako mikrokontroler jako brzo pristupa sadržaju u

memoriji (brzina memorijske sabirnice je preko 3.8 GB /s), a količina unaprijed izračunatih vrijednosti nije jako velika (mogu s spremiti unutar 1MB SRAM-a odnosno 2MB Flash memorije), ova optimizacija znatno doprinosi brzini ali i jednostavnosti samog sustava (a može su u budućnosti nadograditi odnosno korigirati).

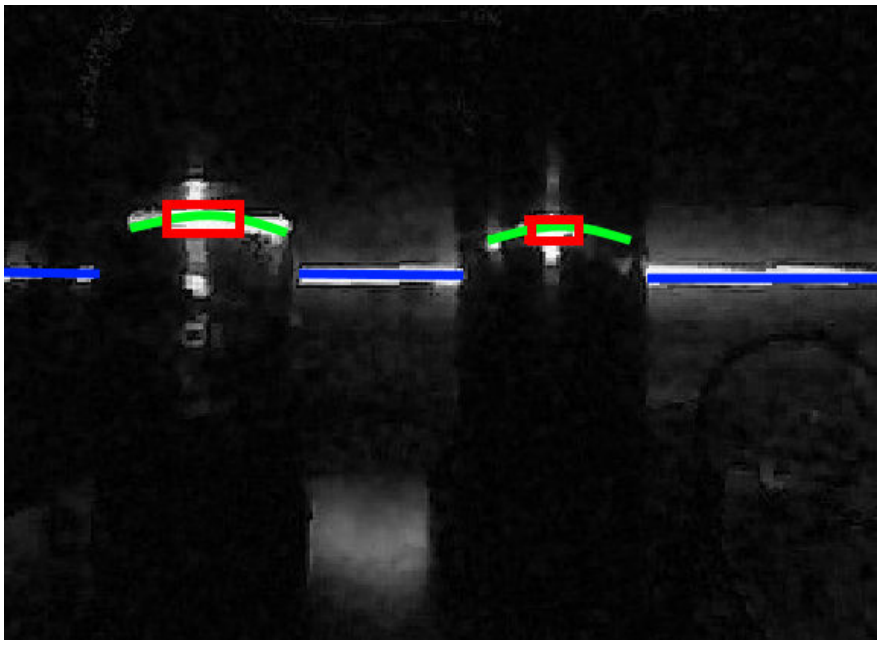


Slika 18 – Ilustracija principa rada laserskog sustava za točnu detekciju položaja

Princip rada sustava je ilustriran na Slici 18. Laserski projektor u jednom trenutku projicira lasersku liniju (u prostoru je to ravnina) gdje se linija odbija od prve prepreke (napomena, prolazi i kroz prozirne objekte kao što je pivska boca) , te se reflektira natrag u kameru. Neobrađena slika se sprema u *framebuffer*, nakon čega se gasi laserska zraka i izvrši se snimanje iste scene bez laserske linije. Ovo je napravljeno kako bi se metodom frame-differencinga (razlikovanjem okvira slika) mogla izdvojiti reflektirana laserska linija. Kako je primjena ovog sustava snimanje prozirnih i polu-prozirnih materijala (staklo) osim reflektirane linije (gdje je radi jako malog koeficijenta refleksije stakla zraka jedva vidljiva), može se promotriti i svjetlost koja prođe kroz bocu (transmisija) kao i raspršenje koje je posljedica prolaska zrake kroz rubne dijelove boce. Ovo predstavlja znatni izazov za obradu dobivenih podataka, i pronalaska središta boce u koordinatnom sustavu. Koordinatni sustav je definiran s obzirom na kameru / laser, odnosno ishodište se postavlja u središte aluminijske konstrukcije (radi lakšeg naknadnog umjeravanja), kao što je i prikazano na slici 18.

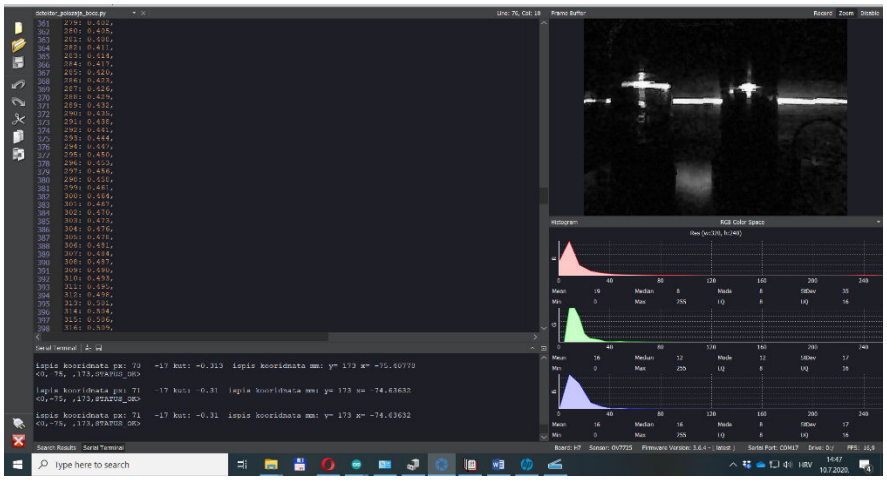
Primjer što kamera vidi nakon „frame differecing-a“ je prikazano na slici 19. Objekt (pozadina) reflektira liniju koja je umjetno obojana plavom linijom. Zelenom bojom su izdvojene krivulje koje pripadaju objektu koji se snima. Zakrivljenije linije nastaje kao posljedica različite udaljenosti površine od kamere. Bliži objekti imaju veći disparitet (udaljeniji su od plave linije), dok su dalji bliži plavoj liniji. Ako se kamera dobro kalibrira, i uklone distorzije, tada se točan broj piksela dispariteta može povezati sa udaljenosti od sustava (izvora linije). Crvenim pravokutnikom je označeno najsvjetlije područje detektirane krivulje odnosno pretpostavljena točka središta refleksije sa površine boce.

Dobro kalibrirana kamera dozvoljava da se svaki horizontalni piksel poveže sa kutom od centralne osi kamere (geometrijska kalibracija kamere sa centralnom projekcijom). Točna informacija o kutu se izvlači iz LUT tablice koja je spremljena u memoriji mikrokontrolera. Ako je poznata udaljenost snimanog objekta i kut pod kojim se nalazi od kamere, jednostavnom trigonometrijskom relacijom se može izračunati točan položaj gdje se u ravnini reflektirala zraka od površine boce. Ova lokacija ne odgovara izravno središtu boce, koje se naknadno izračunava iz poznatog promjera pivske boce (u našem slučaju 6 cm), pri čemu je potrebno uvrstiti i korekciju kako reflektirana zraka ne pada točno na središte boce već je pomaknuto prema kameri (trigonometrijska relacija). Primjer izgleda sučelja sa OpenMV je prikazan na slici 20, gdje se sa desne strane možete primijetiti slika koja se trenutno nalazi u frame bufferu.



Slika 19 – Primjer obrađene slike nakon frame-differencig metode

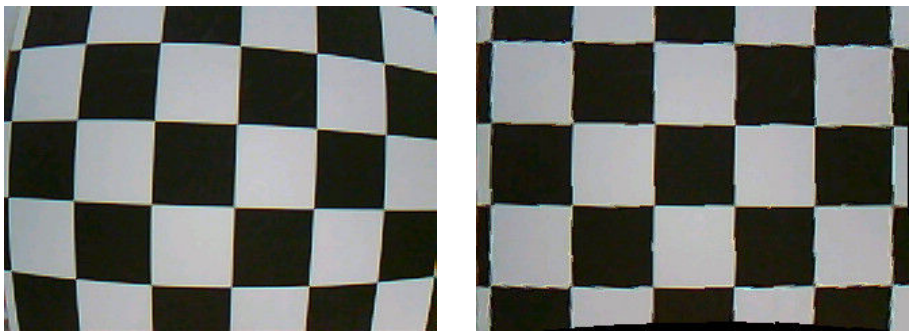
Napominjemo, da ukoliko je potrebno da sustava može mjeriti i nagib boce u više smjerova potrebno je koristiti još jedan laserski projektor koji je dodatno vertikalno pomaknut te analizirati dvije rekonstruirane točke sa površine boce. Osim toga radi razlikovanja tri slike (laser1, laser2 i bez lasera), rezultirajući sustav bi bio nešto sporiji (i do 50%, odnosno stabilno bi radio na 20 fps)



Slika 20 – Sučelje za programiranje i debugiranje OpenMV kamere

U sadašnjoj fazi razvoja sustav je u mogućnosti detektirati jednu bocu (traži najbližu, dok ostale zanemaruje), ali se može jednostavno nadograditi da prati točan položaj više boca unutar vidnog polja kamere. Ovaj korak nije napravljen jer se pretpostavlja da se u malom i ograničenom području (20 x 20 cm) neće nalaziti više od jedne boce, dok bi pokušaj traženja druge boce koja bi mogla biti djelomično zaklonjena, mogao dati krive rezultate za tu bocu. Realizirani sustav je u mogućnosti pratiti položaj boce frekvencijom od 30-ak uzoraka po sekundi (tj. slika u sekundi - fps), pri čemu se kontinuirano prijavljuje položaj putem UART-a mikrokontroleru koji sadrži ROS čvor. Zbog velike količine podataka koje se šalju putem serijske veze, preporuča se uspostava nešto brže veze (57600 baudrate ili 112000 baudrae) sa ROS čvorom, iako prototip radi u laboratorijskim uvjetima stabilno na brzini od 9600 bauda.

Kalibracija kamere se vrši kako bi se uklonila radijalna distorzija uzrokovana lećom kamere. Originalna slika sa kamere je prikazana na slici 21 lijevo, dok je ista slika nakon uklonjene distorzije prikazana na istoj slici desno. Zbog jako male fokalne duljine leće (2.8 mm leća sa M12 navojem) radijalna distorzija je znatna, te u potpunosti onemogućava da se kamera koristi za točno mjerenje ukoliko distorzija nije uklonjena.



Slika 21 – Distorzirana slika sa kamere (lijevo), te slika nakon uklanjanja distorzije (desno)

U slučaju promjene leće kamere, korisnik se može poslužiti priloženim skriptom namjestanje_parametra_lece.py gdje se mogu izračunati parametri nove leće ili kamere.

Važno je napomenuti da dvije iste kamere i leće nemaju iste parametre, i svaka promjena komponente sustava zahtjeva ponovno namještanje parametara leće (odnosno intrinzičnu kalibraciju). Računaju se parametri radijalne distorzije: središte distorzije (koje ne mora i niti je u sredini slike) i u konačnici ispravlja se nagib tj. rotacija kamere. Ista skripta se može koristiti i za (ekstrinzičnu) kalibraciju kamere, tj. povezivane piksela kamere (točke na slici kamere) i kuta od centralne osi kamere, pri čemu se izrađuje novi LUT.

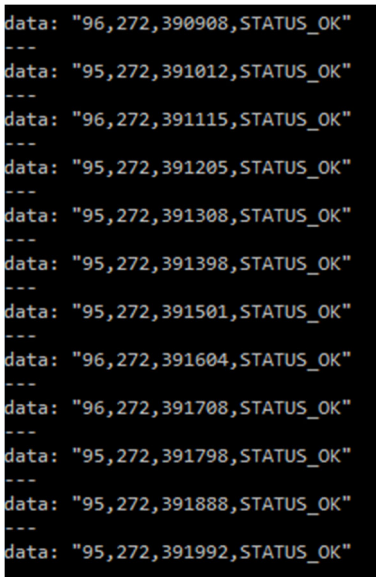
Spomenuti LUT-ovi se izrađuju pomoću skripte koja je kompatibilna sa MATLAB-om (ili Octave-om). U skriptama korisnik prati upute i unosi očitane koordinate za nekoliko točaka na temelju čega se interpoliraju vrijednosti za sve moguće vrijednosti na slici, i pripremaju u format zapisa P*ython dictionary*-a. Takav kod (tekstualni) se kopira u zaglavlje Python skripte na za to predviđeno mjesto. Ovakav pristup omogućava krajnjem korisniku da po potrebi jednostavno izmjeni postavke rada sustava. Kao i kod primjera sustava za detekciju čepa boce, priloženi Python kod omogućava korisniku da jednostavno utječe na parametre rada sustava, bez potrebe za poznavanjem funkcionalnosti cijelog sustava.

Poruke koje OpenMV kamera generira imaju oblik <*X, Y, TS, P*> gdje je:

- < i > su znakovi početka i kraja poruke koja se šalje, a koja pomaže kod parsiranja primljene poruke na strani Arduino Nana.
- *X* je varijabla tipa *integer* a predstavlja x koordinatu središta pivske boce (u milimetrima) u odnosu na ishodište koordinatnog sustava definiranog kao na slici 18. Potrebno je primijetiti da je za izračun koordinate pretpostavljena boca promjera 6 cm. Ukoliko se upotrebljava boca drugačije promjera, korisnik navedeno može korigirati u izbornom kodu. Slična napomena vrijedi i za izračunatu y koordinatu središta boce.
- *Y* je varijabla tipa *integer* a predstavlja y koordinatu središta pivske boce (u milimetrima) u odnosu na ishodište koordinatnog sustava definiranog kao na slici 18. Potrebno je primijetiti da je za izračun koordinate pretpostavljena boca promjera 6 cm.
- *TS* je varijabla (tipa *integer*) koji predstavlja vremenski pečat (timestamp) trenutnog događaja, a izražena je u broju milisekundi od trenutka uključenja mikrokontrolera.
- *P* je varijabla tipa *string* (tekstualna varijabla). U trenutnoj arhitekturi nema konkretnu funkciju (šalje se poruka „STATUS_OK“) već je ostavljena za buduće namjene koje korisnik može imati i naknadno definirati.

Navedene poruke se zatim šalju do Arduino Nano mikrokontrolera na kojemu se pokreče ROS čvor koji osigurava kompatibilnost sa ROS okvirom u vidu prikladnog načina komunikacije i izmjene podataka/informacija. Zbog osiguranja/zadržavanja visoke brzine (fps) rada cjelokupnog sustava (barem u verziji prototipa) ovaj mikrokontroler ne vrši nikakvo parsiranje poruka i dodjeljivanje složenijim tipovima ROS poruka (posebno uzimajući u obzir činjenicu da je Naručitelj izrazio namjeru da sam kreira ROS poruke za OPINAU-FILLER sustav), već se samo uklanjaju znakovi početka i kraja poruke (tj. < i >), te se poruka kao *string* varijabla objavljuje na ROS temu naziva */bottlePrecisePosition (*primjenom *rosserial* paketa*).* Krajnji korisnik zatim može na vanjskom računalu (brže) parsirati poruku i eventualno upotrijebiti dobivenu informaciju kao dio složenijeg sustava

upravljanja i kontrole proizvodnog procesa. Statusi se objavljuju kontinuirano na odgovarajuću ROS temu i to brzinom kojom ih kamera šalje (max. 30 fps). Ukoliko to nije nužno za daljnje procese upravljanja i kontrole proizvodnog proces, a uzimajući u obzir tipičnu brzinu kretanja boce u dijelu procesa koji se nadzire, Naručitelj može ovu brzinu i smanjiti a kako bi se uštedjeli računski resursi i povećala pouzdanost sustava. Primjer ispisa sa odgovarajuće ROS teme na strani računala je prikazan na slici 22.



Slika 22 – Primjer ROS poruka (tipa *string*) primljenih na strani računala od trećeg modula

Naposljetku, kao i u prethodnom poglavlju, navodimo tablicu (Tablica 4) materijala korištenih pri izradi laboratorijskog prototipa, pri čemu ne navodimo količinu za konstrukcijske/potporne materijale budući da oni neće biti potrebni (ne barem u tome obliku i obujmu na stvarnoj proizvodnoj liniji).

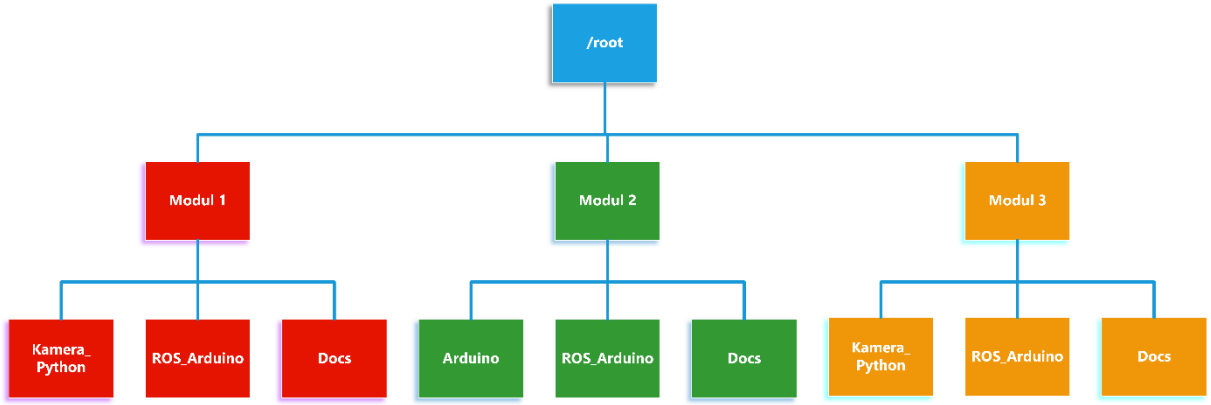
Tablica 4 – Najvažnije komponente korištene pri izradi Modula 3

Komponenta	**Količina (jedinica)**
OpenMV Cam H7 Image Sensor	1 (kom)
Compact line laser module GREEN 30mW 532nm	1 (kom)
Difuzna pozadina (stakloplastika 4mm, izbrušena brusnim papirom)	1 (kom)
IRFZ44N N-kanalni MOSFET	1 (kom)
Arduino Nano	1 (kom)
Potrošni materija (tiskana pločica, aluminijski profili, 3D printani dijelovi)	/

5. STRUKTURA POPRATNOG CD-a SA PROGRAMSKIM DATOTEKAMA

Svi kodovi korišteni pri izradi sva tri modula su priloženi na CD-u koji je sastavni dio ove studije. Komentari koji se nalaze u odgovarajućim kodovima, zajedno s tekstom ove studije bi trebali biti dovoljni da bi svaki krajnji korisnik mogao shvatiti osnove funkcioniranja i karakteristike svakog modula, te promijeniti odgovarajuće parametre sustava u slučaju promjene neke od sklopovskih (ili drugih) komponenti sustava, a kako je to već na nekoliko mjesta istaknuto u studiji.

Radi lakšeg snalaženja na CD-u ovdje navodimo strukturu direktorija i to grafički slikom 23. Iz same strukture i naziva (pod) direktorija bi trebalo biti jasno što on sadrži.



Slika 23 – Struktura direktorija na priloženom CD-u