Izrada autonomnog drona za pomoć pri parkiranju

Safundžić, Tomislav

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:495607

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-06**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

IZRADA AUTONOMNOG DRONA ZA POMOĆ PRI PARKIRANJU

Završni rad

Tomislav Safundžić

Osijek, 2016.

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 06.07.2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Tomislav Safundžić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3521, 02.08.2013.
OIB studenta:	39029575028
Mentor:	Doc.dr.sc. Ivan Aleksi
Sumentor:	
Naslov završnog rada:	Izrada autonomnog drona za pomoć pri parkiranju
Znanstvena grana rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 3 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	06.07.2016.
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

Sveučilište Josipa Jurja Strossmayera u Osijeku

ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 07.09.2016.

Ime i prezime studenta:	Tomislav Safundžić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3521, 02.08.2013.
Ephorus podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada autonomnog drona za pomoć pri parkiranju**

izrađen pod vodstvom mentora Doc.dr.sc. Ivan Aleksi

i sumentora

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 studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA

Ja, Tomislav Safundžić, OIB: 39029575028, student/ica na studiju: Preddiplomski sveučilišni studij Računarstvo, dajem suglasnost Elektrotehničkom fakultetu Osijek da pohrani i javno objavi moj **završni rad**:

Izrada autonomnog drona za pomoć pri parkiranju

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 07.09.2016. 2016.

_____

potpis

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. PRIMJENJENE TEHNOLOGIJE I ALATI 2
2.1. Alati i tehnologije potrebni za izradu drona 2
2.2. Alati i tehnologije potrebni za obradu slike 3
2.3. Pravilnik o bespilotnim letjelicama 5
3. REALIZACIJA SUSTAVA 6
3.1. Sklopovska izrada makete drona 6
3.2. Implementacija programskog dijela makete drona 9
3.3. Implementacija programskog dijela za obradu slike 10
3.4. Objedinjenje makete drona i rezultata obrade slike 13
4. ZAKLJUČAK 16
LITERATURA 17
SAŽETAK 18
ABSTRACT 19
ŽIVOTOPIS 20
PRILOZI 21
PRILOG A. Fotografije izrađene makete drona 21
PRILOG B. Programski dio makete drona 22
PRILOG C. Program za obradu slike 23
PRILOG D. Ostvarena komunikacija unutar programa za obradu slike 26
PRILOG E. Ostvarena komunikacija za Arduino Uno 27

1. UVOD

Cilj ovog završnog rada je izraditi maketu letećeg drona koji bi osigurao pomoć pri parkiranju automobila. Nakon uzleta, dron prati položaj automobila. Dron lebdi na prikladnoj visini za snimanje automobila. Praćenje je ostvareno pomoću obrade slike i izdvajanja automobila iz okoline. U radu će se prvo razmotriti primijenjeni alati i tehnologije, a potom i sama realizacija sustava.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je napraviti maketu letećeg drona s kamerom za prijenos videa uživo. Nakon uzleta iz ruku vozača koji sjedi u automobilu, prateći položaje žmigavaca, dron bi lebdio na prikladnoj visini za snimanje automobila. Praćenjem položaja automobila vozač se lakše parkira.

2. PRIMJENJENE TEHNOLOGIJE I ALATI

2.1. Alati i tehnologije potrebni za izradu drona

Pri izradi drona, kao glavni dio i kontroler svih parametara drona korišten je Arduino Uno. Arduino Uno, za razliku od Arduino Pro Mini, omogućuje dostatno napajanje ostalih komponenti, te također ostavlja dovoljno prostora i za eventualne buduće preinake i poboljšanja sustava.
Slijedeći važan dio je giroskop. On prepoznaje trenutne promjene položaja i u realnom vremenu ih odašilje upravljaču drona, u ovom slučaju Arduinu Uno. Odabran je L3G4200D giroskop koji koristi tri osne ravnine. Navedeni giroskop ne uspoređuje trenutnu vrijednost položaja s nekom referentnom veličinom, već ju uspoređuje s prethodnom vrijednošću. Pa tako za slučaj stacionarnog stanja, vrijednost giroskopa na izlazu će biti 0.

Sl. 2.1. L3G4200D giroskop

Okvir drona je karbonski, što mu daje veliku čvrstoću i veliki modul elastičnosti. No ono najvažnije je – masa. Karbonski materijal veoma lagan u usporedbi s drugim materijalima sličnih karakteristika. Upravo je masa i glavni razlog odabira karbonskog okvira jer je masa pri letenju veoma važno svojstvo letjelice. Masa samog okvira drona je 558 grama, a udaljenost dvaju suprotnih krajeva iznosi 580mm. Dodatna prednost okvira je u već raspoređenim izvodima za povezivanje elemenata na napajanje.
Motori su Turnigy Multistar Elite 2216 čije su specifikacije vidljive u [1]. Za navedene motore potrebni su propeleri promjera 254 mm (10 inča). Odabrani propeleri su, kao i okvir drona, karbonski. Kontroleri brzine (ESC) su zaduženi za prijenos kontrola između Arduina Uno i motora drona.
Naposljetku, sklopovski dio makete drona, potrebno je objediniti u smislenu cjelinu implementacijom programskog dijela. Sav dio programskog dijela je napravljen unutar Arduino programa

verzije 1.6.7. Riječ je o programu otvorenog koda koji omogućuje pisanje programa za Arduino razvojno okruženje te prijenos gotovog programa na povezani Arduino Uno.

Sl. 2.2. Arduino razvojno okruženje

2.2. Alati i tehnologije potrebni za obradu slike

Za obradu slike, odnosno videa – izdvajanje određenih objekata uz zanemarivanje ostalih potreban je nekih od alata za računalni vid. U ovom projektu se za te potrebe koristi OpenCV. Riječ je o besplatnom alatu kompatibilnom s velikom većinom operacijskih sustava i programskim jezicima C, C++, Phyton, Java, Matlab. OpenCV omogućuje obradu slike učitavanjem već gotovih fotografija ili video zapisa, ali i obradu u realnom vremenu putem web i IP kamera. No, da bi se OpenCV mogao koristiti unutar programskog okruženja, potrebno je instalirati OpenCV biblioteku [2]. Nakon uspješnog preuzimanja verzije OpenCV-a sa službenih stranica, potrebno je raspakirati datoteku. Potom, potrebno je definirati varijable okruženja ovisno o tome gdje se nalazi raspakirana datoteka.

OPENCV_BUILD=C:\opencv\build\x64\vc12

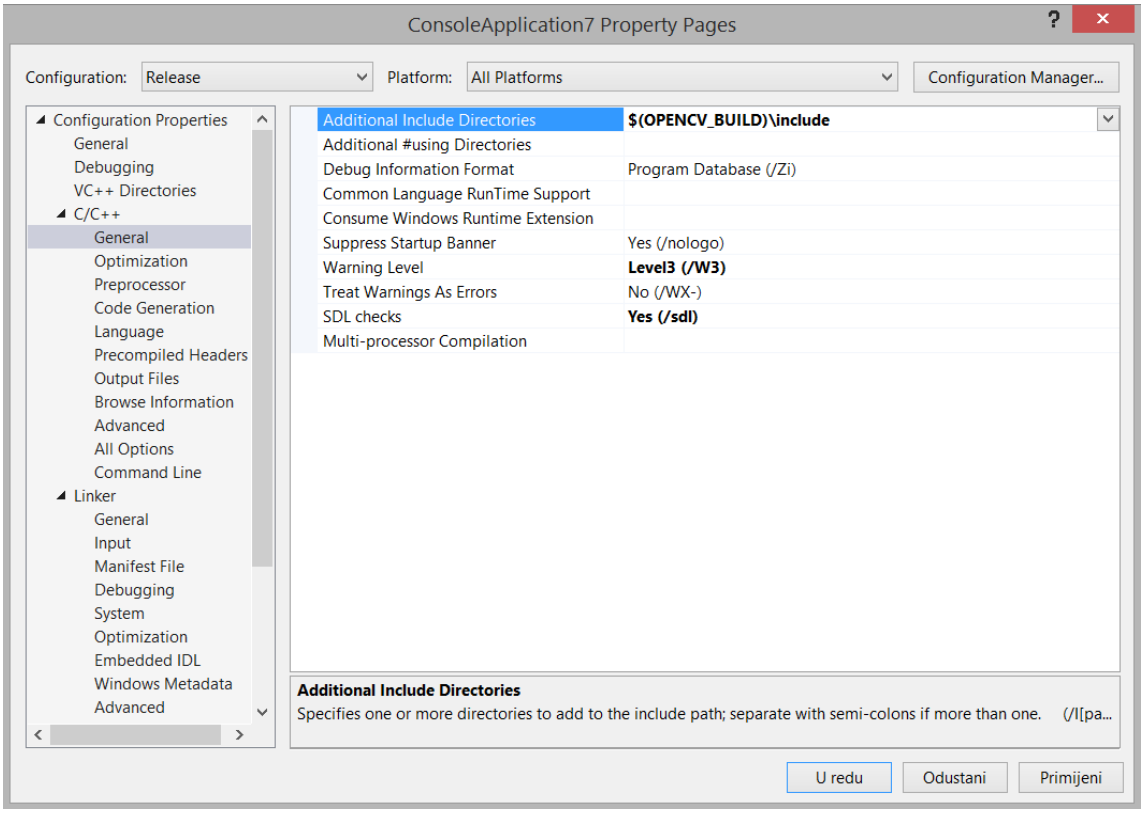
Path=C:\opencv\build\x64\vc12\bin;

Nakon toga, unutar programskog okruženja, u ovom slučaju unutar Visual Studia 2015, potrebno je podesiti nekoliko postavki projekta kako bi se ostvarila mogućnost korištenja OpenCV biblioteka (Postavke za *Release* konfiguraciju i *x64* platformu):

a) C/C++ – General – Additional Include Directories – $(OPENCV_BUILD)\include

b) Linker – General – Aditional Library Directories – $(OPENCV_BUILD)\ x64\vc12\lib

c) Linker – Input – Additional Dependencies – opencv_core2412.lib
opencv_features2d2412.lib
opencv_flann2412.lib
opencv_highgui2412.lib
opencv_imgproc2412.lib
opencv_ml2412.lib
opencv_objdetect2412.lib
opencv_photo2412.lib
opencv_video2412.lib
opencv_videostab2412.lib



Sl. 2.3. Instalacija OpenCV biblioteka unutar Visual Studia 2015

2.3. Pravilnik o bespilotnim letjelicama

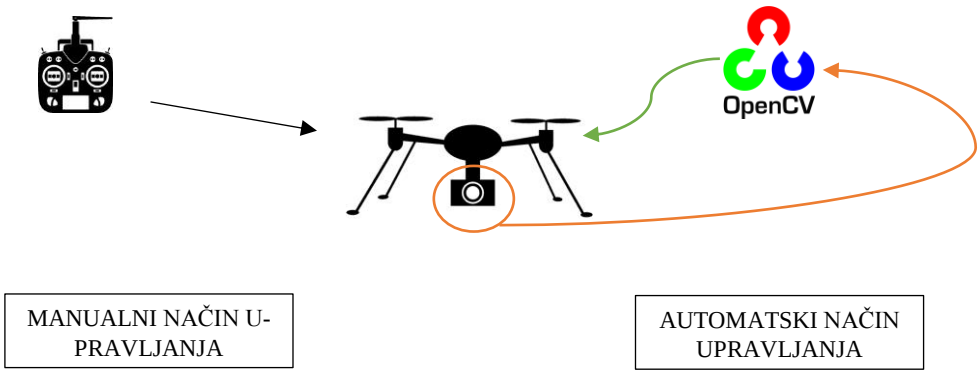
Uslijed ekspanzije bespilotnih letjelica u zrakoplovstvu, CCAA – Hrvatska agencija za civilno zrakoplovstvo po uzoru na regulative Europske Unije u travnju 2015. godine donosi pravilnik o letačkim operacijama sustavima bespilotnih letjelica. Bespilotne letjelice su podijeljene u tri kategorije prema operativnoj masi (letjelice masom manje od 5 kilograma, letjelice između 5 kilograma i 25 kilograma, te letjelice masom veće od 25 kilograma), te u četiri kategorije prema klasi područja na kojem se izvodi letenje u ovisnosti o vrsti površine i naseljenosti područja nad kojim se izvode letačke aktivnosti. [3]

Pa se tako na nenaseljenim područjima i područjima s gospodarskim objektima bez prisutnih osoba uz dozvolu omogućuje letenje svim vrstama bespilotnih letjelica, dok za područja poslovnih i stambenih objekata uz potrebnu dozvolu omogućuje letenje isključivo letjelicama s barem šest motora, izuzev letjelica s četiri motora koje posjeduju padobran. Ograničenja vrijede i za letenje u blizini ljudi.

3. REALIZACIJA SUSTAVA

Cijeli sustav sastoji se od tri glavne cjeline. Fizički izrađena maketa drona s pripadajućim programskim rješenjem, programsko rješenje za prepoznavanje i izdvajanje objekata u videozapisu koji dobivamo preko IP kamere s drona, te programsko rješenje koje objedinjuje prve dvije cjeline u jednu funkcionalnu koja omogućuje automatsko pozicioniranje drona u ovisnosti o obrađenim podacima dobivenih s kamere na dronu.

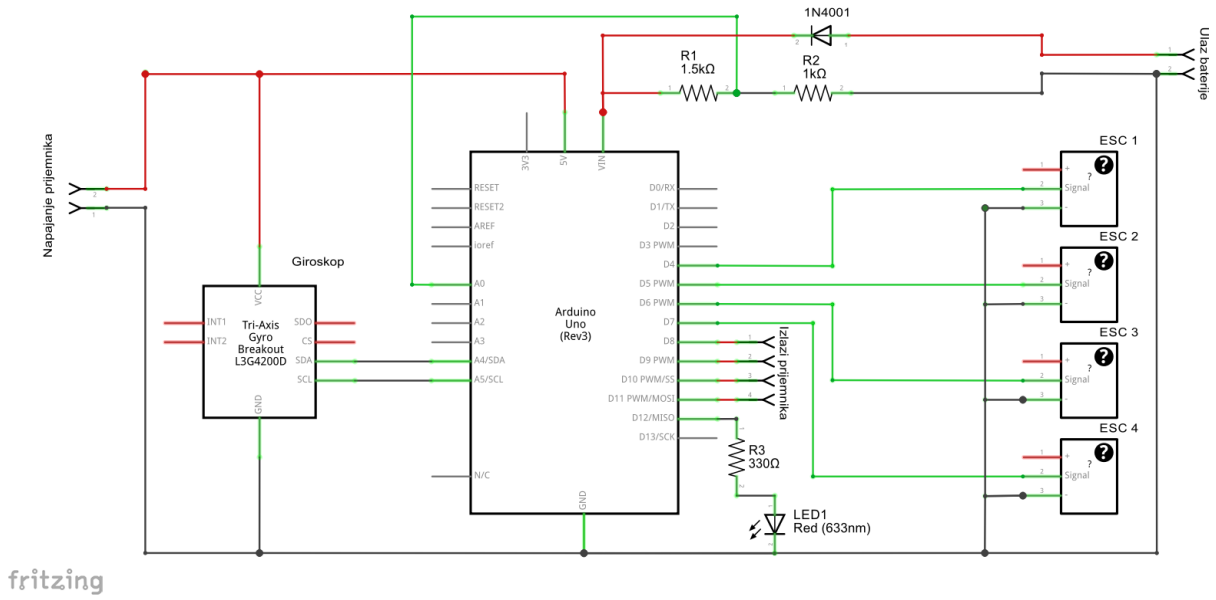
Dron ima dva načina upravljanja (Sl. 3.1.) – manualni pomoću RC kontrolera, te automatski na osnovi slike i rezultata programa za obradu slike. Kod automatskog načina upravljanja dronom, dron već lebdi te video s IP kamere odašilje, odnosno emitira, putem WiFi-a preko Real Time Streaming Protokola (rtsp) u realnom vremenu na korisnikovom laptopu. Na laptopu se nad dobivenim videozapisom vrši obrada, također u realnom vremenu, te se na temelju algoritama izračunava potrebna putanja drona. Tako izračunata putanja se šalje dronu putem Bluetooth veze te se dron na osnovu tog proračuna i dobivenih podataka pozicionira u prostoru.



Sl. 3.1. Načini upravljanja dronom

3.1. Sklopovska izrada makete drona

Kako bi se moglo pristupiti spajanju komponenata, prvo je potrebno složiti karbonski okvir drona. Nakon toga, važno je pravilno rasporediti komponente kako bi se masa ravnomjerno rasporedila. Spajanje komponenti prema shemi (Sl. 3.2.) obavljeno je dijelom pomoću vodiča (tzv. *jumperi*), a dijelom lemljenjem. Pri spajanju lemljenjem, preporučljivo je prethodno nanijeti lema na površine koje se spajaju (krajevi vodiča, pinovi i ostala lemna mjesta komponenti) zbog lakšeg i kvalitetnijeg lemljenja.



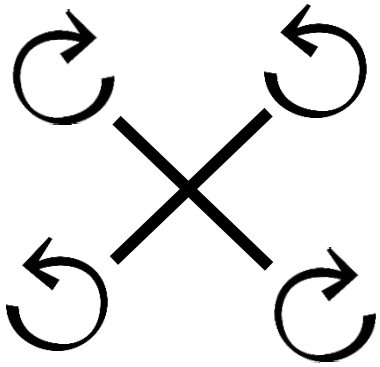
Sl. 3.2. Shema spoja komponenti

Giroskop se napaja preko Arduino Uno razvojne pločice pa je izvode napajanja potrebno spojiti na pinove 5V i GND Arduina. Također, potrebno je spojiti i SDA pin, odnosno pin za serijski prijenos podataka, na A4 pin Arduina, te SCL pin, odnosno pin za serijski sat pomoću kojeg se uzorkuju vrijednosti sa SDA pina, na A5 pin Arduina.

Pri spajanju kontrolera brzine, signalni pin svakog kontrolera brzine potrebno je spojiti s Arduinom i to redom na pinove D4, D5, D6 i D7. Također, potrebno je spojiti negativne polove na GND Arduina kako bi se izbjegle moguće eventualne smetnje pri radu.

Vrlo je važno obaviti grupnu kalibraciju svih četiriju kontrolera brzine kako bi se ostvarila sinkronizacija motora pri radu. Sinkronizacija se obavlja pokretanjem svih četiriju motora preko njihovih konrtolera na punoj snazi. Na taj način kontroleri brzine prepoznaju maksimalnu moguću snagu motora te svojim zvukom sugeriraju da su sinkronizirani. Važno je taj postupak provoditi bez propelera na motorima kako bi se spriječile moguće nezgode i ozljede!

Također, prilikom povezivanja motora i kontrolera brzine važno je pravilno ih spojiti kako ne bi došlo do vrtnje u pogrešnom smjeru. Naime, za dron s četiri motora važan je smjer vrtnje pojedinog motora – prednji lijevi motor i zadnji desni motor se vrte u smjeru kazaljke na satu, dok ostala dva motora, prednji desni motor i zadnji lijevi motor, se vrte u smjeru suprotnom od smjera kazaljke na satu. Pošto motori imaju tri izvoda, u slučaju pogrešnog smjera vrtnje motora, dovoljno je prespojiti dva izvoda i motor će promijeniti smjer vrtnje.



Sl. 3.3. Raspored vrtnje motora

Potrebno je dodati diodu kako bi se spriječilo da napajanje pri spajanju Arduino Uno razvojne pločice s računalom putem USB-a dođe do kontrolera brzine i motora.

Zbog pražnjenja baterije prilikom leta motori proizvode manje snage, odnosno, motori se ponašaju drugačije obzirom na raspoloživi kapacitet baterije. Da bi se to kompenziralo, kontroler mora znati trenutni napon baterije. To je moguće ostvariti djeliteljem napona. Zbog napona na izlazu baterije od 12.6 V, te pada napona na diodi od 0.6 V i potrebe za maksimalno 5 V na analognom ulazu, za bateriju s tri ćelije potrebni su otpornici od 1 kΩ i 1.5 kΩ. Za ostale tipove baterija potrebno je proračunati iznose otpora otpornika.

$$U_{out} = \frac{U_{in}}{R_1+R_2} * R_1 \qquad (3-1)$$

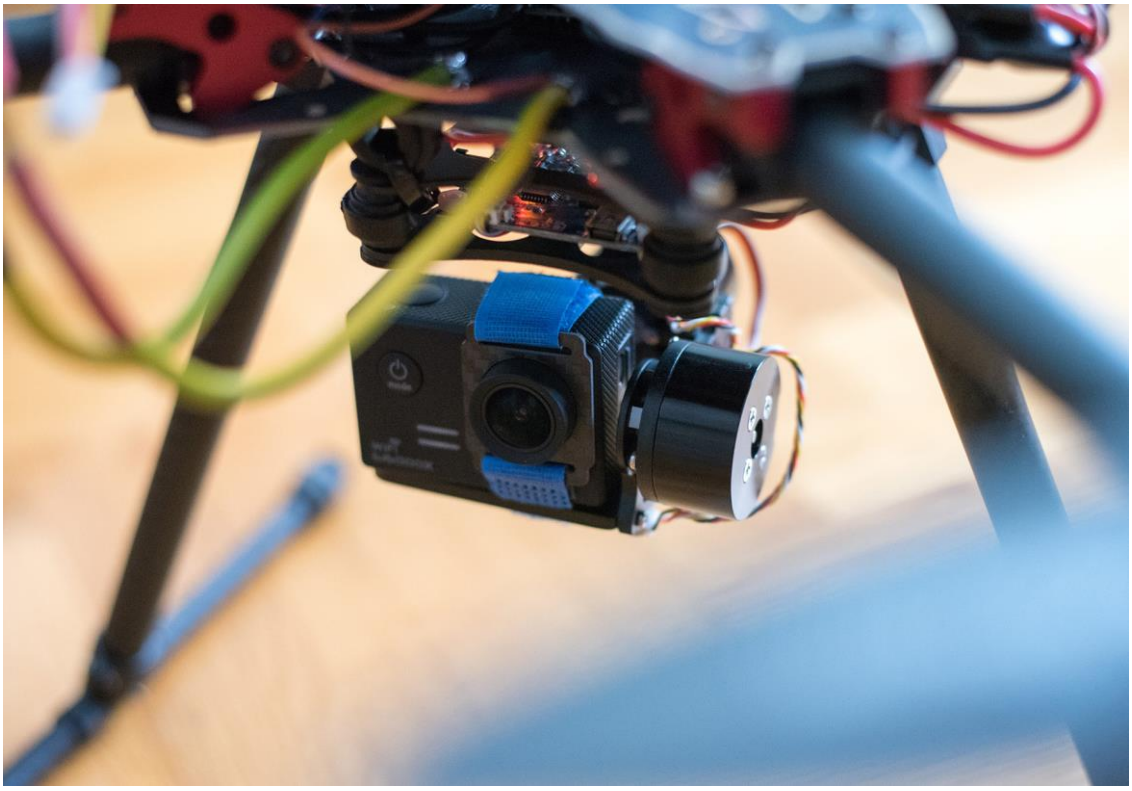
gdje je:

U_{out} – napon analognog ulaza (maksimalna vrijednost napona iznosi 5 V)

U_{in} – napon baterije (ovisno o broju ćelija i tipu baterije)

R_1, R_2– otpornici u djelitelju napona

Kamera drona je sportska kamera SJCAM SJ5000X. Kamera je pričvršćena na dron pomoću nosača kamera koji ujedno i stabilizira kameru u dva pravca (engl. *gimbal*) uslijed pokreta. Na taj način se dobiva stabilnija slika videozapisa koju se može lakše i kvalitetnije obraditi kako bi se izdvojio objekt od interesa.



Sl. 3.4. Stabilizator kamere

3.2. Implementacija programskog dijela makete drona

Programski dio makete drona je zadužen za očitovanja vrijednosti koje daje giroskop, kontrolu i upravljanje motorima preko kontrolera brzine te održavanje drona stabilnog u letu na osnovu uzorkovanih parametara. Uz definiranje globalnih varijabli, kao i kod svih ostalih Arduino programa, postoje *setup* i *loop* funkcija. *Setup* funkcija se pokreće samo jednom, odmah pri početku rada Arduino razvojne pločice. *Setup* funkcijom se izvršava inicijalizacija programa i sklopova vezanih za Arduino. Za razliku od *setup* funkcije, *loop* funkcija se izvršava u beskonačnoj petlji.
U *setup* funkciji se definira registar za podatke giroskopa. Potom se pokreće inicijalizacija giroskopa. Prikuplja se 2000 uzoraka u svrhu početne kalibracije giroskopa. Nakon završetka mjerenja, uzorkovani podaci se obrađuju i dobiva se srednja vrijednost pojedinih vrijednosti giroskopa. Na osnovu toga se neutraliziraju eventualne smetnje i nepravilna očitovanja giroskopa i omogućuje kvalitetnija očitovanja u daljnjem radu.
U *loop* funkciji se za potrebe manualnog načina rada drona, konvertiraju pojedine vrijednosti ulaznih signala. Nad tim dobivenim vrijednostima se zajedno s vrijednostima dobivenim od strane giroskopa vrše korekcije zadane PID regulacijom, te se pomoću njih odvija pravilan rad sustava. Pojedine vrijednosti ulaznih signala – gas (engl. *throttle*), okret (engl. *yaw*), nagib (engl. *pitch*) te

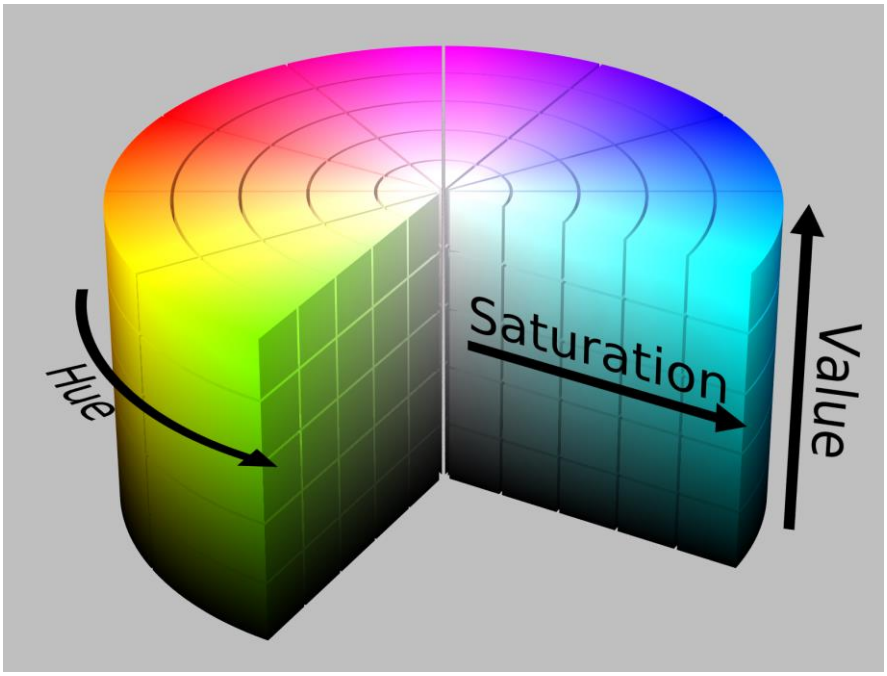
krilca (engl. *roll*) – se konvertiraju u raspon vrijednosti od 1000 do 2000, pri čemu vrijednost 1500 predstavlja nulti položaj. Tako dobivene vrijednosti se šalju kontrolerima brzine. Uz glavne funkcije Arduino programa, veoma važan dio programa su i vrijednosti PID regulatora koji su definirani kao globalne varijable. PID regulacija se obavlja dovođenjem sustava na rub stabilnosti pri određivanju svakog pojedinog parametra regulacije (Ziegler – Nichols metoda) [4]. Prvo je potrebno odrediti P vrijednost, uz ostale dvije vrijednosti postavljene u nulu. P vrijednost se povećava sve dok je sustav stabilan. Kod postizanja vrijednosti za koju je sustav na granici stabilnosti (kod drona se ona očituje trešnjom u pravilnim vremenskim periodima) vrijednost je potrebno smanjiti na prvu nižu vrijednost kod koje je dron stabilan. Nakon toga se na isti način određuje I komponenta regulatora kojom se regulira koliko dobro dron prepoznaje vanjske utjecaje i koliko spremno reagira na njih (npr. vjetar). D komponenta regulatora određuje kako će se dron ponašati pri naglom pokretu, odnosno kako će kompenzirati iznenadno zadanu vrijednost. [5]

3.3. Implementacija programskog dijela za obradu slike

Programski dio za obradu slike dobivene s IP kamere na osnovu OpenCV biblioteka izdvaja objekt interesa – automobil i na osnovu rezultata obrade ukazuje gdje je objekt u ovisnosti na središnji dio slike. Cijeli programski kod vidljiv je u prilogu Prilogu C.

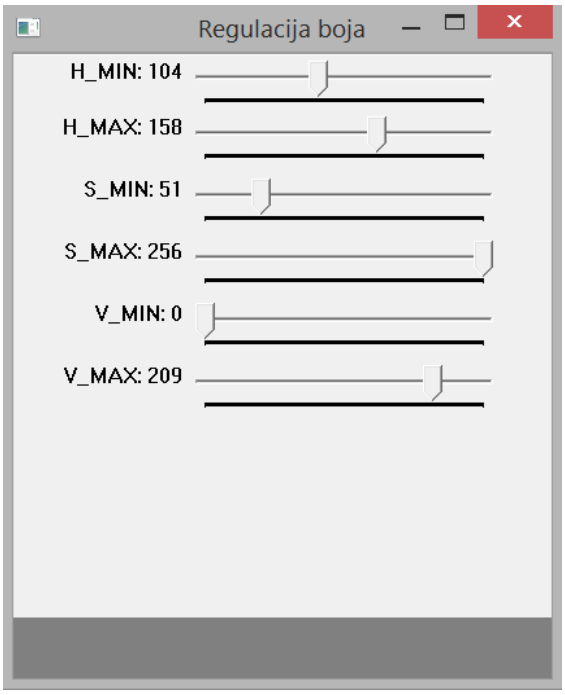
Prvenstveno, potrebno je dodati datoteke potrebne za rad programa. Od OpenCV datoteka, potrebne su *highgui.cpp* i *cv.h*. *highgui.cpp* datoteka omogućuje kreiranje prozora za prikaz videa i slike, omogućuje stvaranje i upravljanje klizačima za regulaciju određenih postavki i obrade nad primljenom slikom s izvora, čitanje i zapisivanje videozapisa u i iz memorije te čitanje videozapisa s kamere. A datoteka *cv.h* omogućuje osnove računalnog vida – sadrži ključne elemente koji su potrebni za rad s OpenCV-om.

Nakon zaglavlja, potrebno je inicijalizirati varijable kojima se regulira filter boja. Umjesto standardnog RGB modela boja, u programu se koristi HSV model boja. Riječ je o cilindričnom modelu boja pomoću kojeg se, za razliku od RGB modela boja koji koristi tri vrijednosti za određivanje kolorita – crvena, zelena i plava, kolorit određuje jednom vrijednošću – nijansom (engl. *hue*). Ostale dvije vrijednosti su zaslužni za zasićenost boje (engl. *saturation*) te za vrijednost osvjetljenja (engl. *brightness*). [6]



Sl. 3.5. Raspon HSV modela boja

Također, odmah na početku, definira se i rezolucija videozapisa kojeg se dohvaća. Riječ je o dimenzijama 640 piksela po širini i 480 pikslea po visini. Deklariraju se i prozori preko kojih se ostvaruje kontrola, pregled i upravljanje obradom slike. Upravljanje i regulacija boja je ostvarena pomoću šest klizača. Pojedina dva klizača zaduženi su za postavljanje minimalne i maksimalne vrijednosti jednog od tri parametra HSV modela boja (nijansa, zasićenost i osvjetljenje).



Sl. 3.6. Regulacija boja

Na osnovu postavljenih klizača, u *treshold* prozoru se dobiva određena binarna slika – riječ je o videozapisu koji se dobiva u realnom vremenu, no konvertiran je u binarnu sliku. Pa tako, ono što je objekt od interesa koji je definiran klizačima za regulaciju boja, predstavlja binarnu jedinicu, a sve ostalo je binarna nula. Pomoću funkcije *findContours* [7] određuju se krajnji dijelovi objekta od interesa. Također, definiraju se i minimalne i maksimalne moguće vrijednosti područja binarne jedinice u *treshold* prozoru. Minimalne vrijednosti je potrebno definirati zbog zanemarivanja mogućeg šuma u slici te objekata sličnih karakteristika objekta od interesa. Maksimalne vrijednosti je potrebno definirati zbog mogućeg loše definiranog klizača za regulaciju boja i sprječavanja pogrešnog definiranja objekta od interesa. Pomoću funkcije *morfo_tranformacija* pikseli binarnih jedinica se grupiraju u veće cjeline te omogućuju jasnije i kvalitetnije izdvajanje objekta od interesa od ostatka slike. Riječ je o OpenCV operaciji koja se sastoji od dvaju operatora – erozije (engl. *erode*) i širenja (engl. *dilate*). Ta dva morfološka operatora omogućuju obrađivanje slike na temelju pojedinih oblika objekata na slici. Omogućuju spajanje individualnih objekata u veće skupove, odnosno grupiranje objekata sličnih svojstava te uklanjanje nepotrebnog šuma. Na taj način, omogućuje se lakše izdvajanje objekta od interesa od ostatka slike zanemarujući eventualne šumove nastale analizi slike. [8]

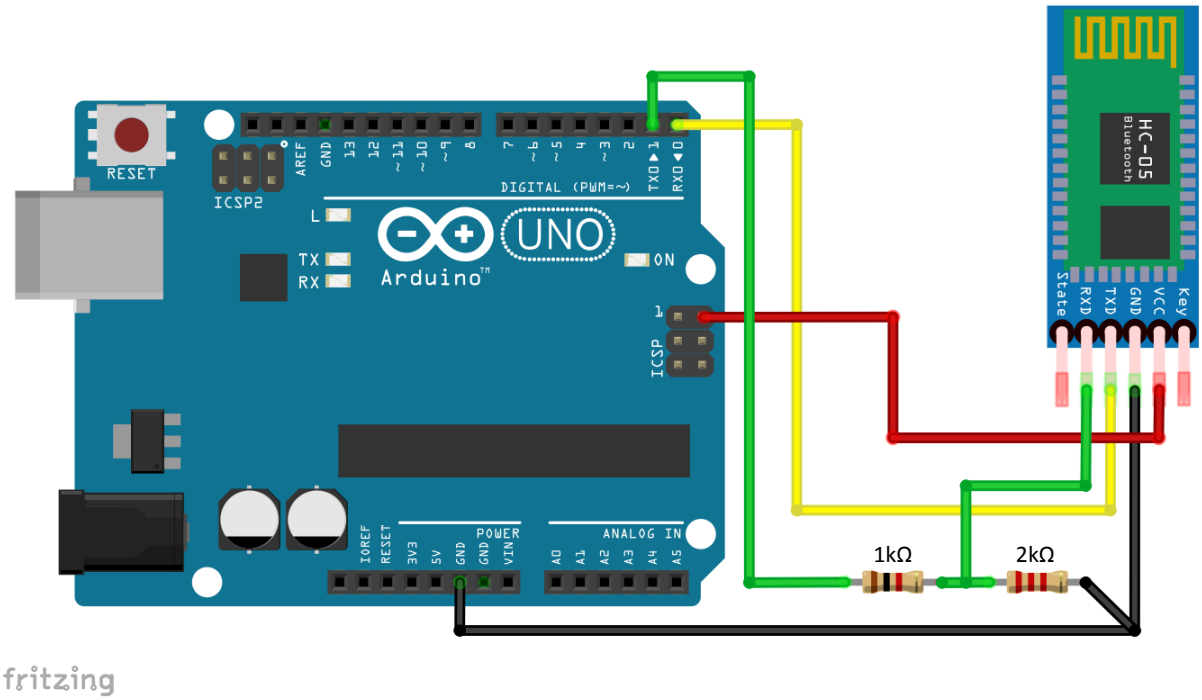
Sl. 3.7. Binarna slika (*treshold*)

Kad je objekt od interesa definiran preko klizača i kad su vidljive njegove konture na *treshold* prozoru, pomoću funkcije *oznaka_objekta* se crta kursor unutar sredine objekta od interesa. Kursor se sastoji od kružnice te dvaju promjera, te lokacije u odnosu na veličinu prozora – 640 piksela po širini i 480 piksela po visini. Pomoću tako dobivene lokacije se u odnosu na sredinu prozora (320,240) definira pomak kako bi objekt od interesa bio u sredini slike.

U *main* funkciji je potrebno omogućiti dohvat videozapisa u realnom vremenu s IP kamere putem rtsp protokola. OpenCV klasa *VideoCapture* omogućuje otvaranje videozapisa preko adrese 192.168.1.254. Nakon uspješnog otvaranja videozapisa, pokreće se beskonačna petlja u kojoj se dobivena slika konvertira u radni model boja – HSV. Zatim se slika konvertira u binarnu sliku te se pozivaju prethodno definirane funkcije za izdvajanje objekta od interesa iz ostatka slike. Te se naposljetku definiraju prozori koji se otvaraju korisniku pri pokretanju programa pomoću OpenCV funkcije *imshow*.

3.4. Objedinjenje makete drona i rezultata obrade slike

Prethodno napravljenu maketu drona s pripadajućom programskom podrškom i programski dio za obradu slike potrebno je povezati kako bi se ostvarila cjelokupna funkcionalnost. Međusobna komunikacija drona i programskog dijela za obradu slike je ostvarena pomoću Bluetooth veze. Prema slici 3.8. vidljivo je da se RX i TX pinovi Bluetooth modula i Arduina Uno međusobno spajaju. Razlog tomu je to što je pin za slanje (TX) na Bluetooth modulu ujedno i pin za primanje (RX) na Arduinu, vrijedi i obratno. Kad bi se spojili međusobno isti pinovi, ne bi došlo do komunikacije. Potrebno je ostvariti i djelilo napona jer podatkovnim pinovima (RX i TX) je potrebno 3.3 V, a napajanje je ostvareno preko Arduinovih 5 V. Djelilo napona je ostvareno pomoću dva otpornika otpora 1 kΩ i 2 kΩ. Pa se tako osigurava da trećina napona napajanja dolazi do RX pina Bluetooth modula što u konačnici iznosi potrebnih 3.33 V.



Sl. 3.8. Spajanje Bluetooth modula i Arduina Uno

U programskom dijelu objašnjenom u poglavlju 3.3. potrebno je dodati programski kod za serijsku komunikaciju vidljiv u Prilogu D. Komunikacija se ostvaruje pomoću već dostupnih biblioteka [9] koje se dodaju u zaglavlje programa. Nakon toga, u funkciji *pracenje_objekta*, odmah nakon uvjeta je li objekt pronađen, potrebno je dodati kod pomoću kojega se određuje vrijednost koju se šalje na osnovu pozicije pokazivača kojeg se dobilo funkcijom *oznaka_objekta*. No prvo je potrebno definirati te ostvariti komunikaciju. *Tserial* klasa pri definiranju nove veze zahtjeva tri parametra – COM priključak Bluetooth modula, brzinu prijenosa podataka izraženih u bitovima po sekundi (engl. *baud rate*) te paritet bita.

Arduino programski kod koji je potrebno dodati u programski dio makete drona objašnjenog u poglavlju 3.2 je vidljiv u Prilogu E. U *setup* funkciji se definira serijska komunikacija. U *loop* funkciji se ovisno o dohvaćenom podatku poduzimaju daljnji koraci upravljanja drona. Upravljanje se ostvaruje na način da na osnovu dohvaćenog podatka spremljenog u varijablu *c*, određuje pojedini iznos varijabli preko kojih se određuju nagib i krilca drona.

Na Sl. 3.9. je prikazana slika koju korisnik dobiva u realnom vremenu na laptop s kamere postavljene na dronu. Obradom dobivena binarna slika na osnovu koje je i prepoznat automobil i pomoću koje je ostvareno praćenje je vidljiva na Sl. 3.7. Svakim pomakom drona, mijenja se i položaj

automobila u odnosu na položaj na prethodnoj sličici u videozapisu (engl. *frame*). Taj pomak prepoznaje se pomoću binarne slike, jer se izdvaja sva nepotrebna okolina od automobila. Na osnovu rezultata tako dobivene binarne slike, u glavnom prozoru s videom u realnom vremenu se iscrtava kursor koji označava centar definiranog objekta – automobila.

Sl. 3.9. Videozapis u realnom vremenu s prepoznatim središtem automobila

Unatoč nezgodnom osvjetljenju zbog jakih sjena i svjetlosti, te boje automobila, program je za vrijeme eksperimentiranja uspješno odradio prepoznavanje automobila. Razlog letu podalje od automobila leži prvenstveno u zaštiti ljudi i imovine te propisanim regulativama o bespilotnim letjelicama.

4. ZAKLJUČAK

U ovom završnom radu izrađena je maketa autonomnog drona za pomoć pri parkiranju. Sustav sastoji se od triju glavnih cjelina. Fizički izrađena maketa drona, programsko rješenje za prepoznavanje i izdvajanje objekata u videozapisu, te programsko rješenje koje objedinjuje prve dvije cjeline u jednu funkcionalnu koja omogućuje automatsko pozicioniranje drona u ovisnosti o obrađenim podacima dobivenih s kamere. Glavna prednost ovog rješenja je u tome što je programsko rješenje primjenjivo na sve letjelice s četiri motora, neovisno o veličini i masi. Potrebno je samo obaviti PID regulaciju za svaku letjelicu posebno jer su ti parametri jedinstveni za svaku letjelicu. Mana ovog rješenja pomoći za parkiranje su vremenski uvjeti (padaline i vjetar) te zakon o bespilotnim letjelicama prema kojemu bi se u ovakav tip drona trebao ugraditi padobran kako bi bio u skladu s trenutnim regulativama. Postoje i moguća poboljšanja sustava u vidu izrade aplikacije za pametne telefone pomoću koje bi nadzor kretanja vozila i osiguravanja parkiranja bio jednostavniji i bliži korisniku.

LITERATURA

[1] Specifikacije Turnigy Multistar Elite 2216 motora,
http://www.turnigy.com/brand/motors/, lipanj 2016.
[2] OpenCV dokumentacija,
http://opencv.org/, lipanj 2016.
[3] Pravilnik o sustavima bespilotnih zrakoplova,
http://narodne-novine.nn.hr/clanci/sluzbeni/2015_05_49_974.html, lipanj 2016.
[4] Ziegler – Nichols metoda PID regulacije,
Perić, N., Automatsko upravljanje - predavanja, Zavodska skripta, FER, Zagreb, 1998.
[5] PID regulacija,
https://oscarliang.com/quadcopter-pid-explained-tuning/, lipanj 2016.
[6] HSV model boja,
http://codeitdown.com/hsl-hsb-hsv-color/, lipanj 2016.
[7] Strukturna analiza,
http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html, lipanj 2016.
[8] Morfološki operatori za obradu slike,
http://goo.gl/K1QL3v, lipanj 2016.
[9] Biblioteke za serijsku komunikaciju,
https://github.com/abdurrahmanbmf/Serial-Communication, lipanj 2016.

SAŽETAK

Naslov: Izrada autonomnog drona za pomoć pri parkiranju

U ovom radu izrađena je maketa autonomnog drona za pomoć pri parkiranju. Sustav se sastoji od tri glavne cjeline: makete drona, programa za prepoznavanje i izdvajanje objekata u videozapisu, te programa koji objedinjuje prve dvije cjeline i omogućuje automatsko praćenje objekta. Kroz rad su razmotreni alati i tehnologije kojima je moguće riješiti zadani problem. Nakon opisa odabranih alata i tehnologija, objašnjena je realizacija sklopa: maketa drona i implementacija programskih rješenja.

Ključne riječi: dron, Arduino, C++, OpenCV, elektronika

ABSTRACT

Title: Autonomous drone for helping with parking

In this final paper a model of autonomous parking assisted dron was made. The system consists of three main parts: the model of drone, program that identifies and extracts object from the video and program which combines first two parts and provides automatic tracking. Firstly, tools and technologies used to develop the system are described. Then, the system implementation is explained – model of the drone and implementation of software solution.

Keywords: drone, Arduino, C++, OpenCV, electronics

ŽIVOTOPIS

Tomislav Safundžić rođen je 3. svibnja 1994. godine u Požegi. Od rođenja živi u Pleternici. Osnovnu školu završio je u Pleternici. Školovanje nastavlja upisom u Tehničku školu u Požegi, gdje 2013. godine stječe kvalifikaciju tehničara za računalstvo. Iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U slobodno vrijeme aktivno se bavi fotografijom i videom, te radi kao *freelancer* fotograf i snimatelj.

PRILOZI

PRILOG A. Fotografije izrađene makete drona

PRILOG B. Programski dio makete drona

Cjelokupni programski kod se nalazi na DVD-u priloženom ovom radu. U nastavku slijedi dio koda važan za praćenje ovog rada.

```
//PID regulator
float pid_p_gain_roll = 0.8;
float pid_i_gain_roll = 0.0001;
float pid_d_gain_roll = 18;
int pid_max_roll = 400;

float pid_p_gain_yaw = 4.0;
float pid_i_gain_yaw = 0.02;
float pid_d_gain_yaw = 0.0;
int pid_max_yaw = 400;


//loop funkcija
receiver_input_channel_1 = convert_receiver_channel(1);
receiver_input_channel_2 = convert_receiver_channel(2);
receiver_input_channel_3 = convert_receiver_channel(3);
receiver_input_channel_4 = convert_receiver_channel(4);

//start
  if(receiver_input_channel_3 < 1050 && receiver_input_channel_4 < 1050)start =
1;
  if(start == 1 && receiver_input_channel_3 < 1050 && receiver_input_channel_4
> 1450){
    start = 2;
    pid_i_mem_roll = 0;
    pid_last_roll_d_error = 0;
    pid_i_mem_pitch = 0;
    pid_last_pitch_d_error = 0;
    pid_i_mem_yaw = 0;
    pid_last_yaw_d_error = 0;
  }

  //stop
  if(start == 2 && receiver_input_channel_3 < 1050 && receiver_input_channel_4
> 1950)start = 0;
```

PRILOG C. Program za obradu slike

```
#define _CRT_SECURE_NO_WARNINGS
#include <windows.h>
#include <iostream>
#include <opencv/highgui.h>
#include <opencv/cv.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

using namespace std;
using namespace cv;

int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
const int FRAME_SIRINA = 640;
const int FRAME_VISINA = 480;
const int MAX_BROJ_OBJEKATA = 50;
const int MIN_PODRUCJE_OBJEKTA = 20 * 20;
const int MAX_PODRUCJE_OBJEKTA = FRAME_VISINA*FRAME_SIRINA / 0.5;
const string prozor = "Original";
const string prozor1 = "HSV";
const string prozor2 = "Thresholded";
const string prozor3 = "Nakon spajanja piksela";
const string prozor_klizac = "Regulacija boja";
string intToString(int number) {
	std::stringstream ss;
	ss << number;
	return ss.str();
}
void klizaci() {
	namedWindow(prozor_klizac, 0);
	char naziv_klizaca[50];
	sprintf(naziv_klizaca, "H_MIN", H_MIN);
	sprintf(naziv_klizaca, "H_MAX", H_MAX);
	sprintf(naziv_klizaca, "S_MIN", S_MIN);
	sprintf(naziv_klizaca, "S_MAX", S_MAX);
	sprintf(naziv_klizaca, "V_MIN", V_MIN);
	sprintf(naziv_klizaca, "V_MAX", V_MAX);
	createTrackbar("H_MIN", prozor_klizac, &H_MIN, H_MAX, promjena_kli-
zaca);
	createTrackbar("H_MAX", prozor_klizac, &H_MAX, H_MAX, promjena_kli-
zaca);
	createTrackbar("S_MIN", prozor_klizac, &S_MIN, S_MAX, promjena_kli-
zaca);
	createTrackbar("S_MAX", prozor_klizac, &S_MAX, S_MAX, promjena_kli-
zaca);
	createTrackbar("V_MIN", prozor_klizac, &V_MIN, V_MAX, promjena_kli-
zaca);
	createTrackbar("V_MAX", prozor_klizac, &V_MAX, V_MAX, promjena_kli-
zaca);
}
void oznaka_objekta(int x, int y, Mat &frame) {
	circle(frame, cv::Point(x, y), 24, cv::Scalar(0, 0, 255));
```

```
58          line(frame, cv::Point(x, y), cv::Point(x, y - 25), cv::Scalar(0, 255,
59 0), 1);
60          line(frame, cv::Point(x, y), cv::Point(x, y + 25), cv::Scalar(0, 255,
61 0), 1);
62          line(frame, cv::Point(x, y), cv::Point(x - 25, y), cv::Scalar(0, 255,
63 0), 1);
64          line(frame, cv::Point(x, y), cv::Point(x + 25, y), cv::Scalar(0, 255,
65 0), 1);
66          putText(frame, intToString(x) + " , " + intToString(y), cv::Point(x, y
67 + 20), 1, 1, Scalar(0, 255, 0));
68 }
69 void morfo_tranformacija(Mat &sum) {
70          Mat erozijski_element = getStructuringElement(MORPH_RECT, Size(7, 7));
71          Mat prosireni_element = getStructuringElement(MORPH_RECT, Size(5, 5));
72
73          erode(sum, sum, erozijski_element);
74          erode(sum, sum, erozijski_element);
75
76          dilate(sum, sum, prosireni_element);
77          dilate(sum, sum, prosireni_element);
78 }
79
80 void pracenje_objekta(Mat threshold, Mat &frame, vector<vector<Point>> &conto-
81 urs, vector<Vec4i> &hierarchy) {
82          int x, y;
83          Mat temp;
84          threshold.copyTo(temp);
85          findContours(temp, contours, hierarchy, CV_RETR_CCOMP, CV_CHAIN_AP-
86 PROX_SIMPLE);
87
88          double refArea = 0;
89          bool objekt_pronaden = false;
90          if (hierarchy.size() > 0) {
91                int numObjects = hierarchy.size();
92
93                if (numObjects<MAX_BROJ_OBJEKATA) {
94                      for (int index = 0; index >= 0; index = hierarchy[in-
95 dex][0]) {
96
97                            Moments moment = moments((cv::Mat)contours[index]);
98                            double area = moment.m00;
99
100                           if (area>MIN_PODRUCJE_OBJEKTA && area<MAX_POD-
101 RUCJE_OBJEKTA && area>refArea) {
102                                 x = moment.m10 / area;
103                                 y = moment.m01 / area;
104                                 objekt_pronaden = true;
105                                 refArea = area;
106                           }
107                           else objekt_pronaden = false;
108
109
110                     }
111
112                     if (objekt_pronaden == true) {
113                           oznaka_objekta(x, y, frame);
114                     }
115               }
116
```

```
117             else putText(frame, "Previse suma. Reguliraj usporedujuci thres-
118 hold!", Point(30, 70), 1, 1.1, Scalar(255, 255, 255));
119       }
120 }
121 int main(int argc, char* argv[])
122 {
123       vector< vector<Point> > contours;
124       vector<Vec4i> hierarchy;
125
126       Mat frame;
127       Mat HSV;
128       Mat threshold;
129       int x = 0, y = 0;
130       klizaci();
131       VideoCapture capture;
132
133       //capture.open(0);
134       const std::string videoStreamAddress =
135 "rtsp://192.168.1.254/sjcam.mov";
136       if (!capture.open(videoStreamAddress)) {
137       std::cout << "Error opening video stream or file" << std::endl;
138       return -1;
139       }
140
141       capture.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_SIRINA);
142       capture.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_VISINA);
143
144       while (1) {
145             capture.read(frame);
146             cvtColor(frame, HSV, COLOR_BGR2HSV);
147             inRange(HSV, Scalar(H_MIN, S_MIN, V_MIN), Scalar(H_MAX, S_MAX,
148 V_MAX), threshold);
149
150             morfo_tranformacija(threshold);
151             pracenje_objekta(threshold, frame, contours, hierarchy);
152
153             imshow(prozor2, threshold);
154             imshow(prozor, frame);
155             //imshow(prozor1, HSV);
156
157             waitKey(30);
158       }
159
160       return 0;
161 }
```

PRILOG D. Ostvarena komunikacija unutar programa za obradu slike

```
#include "tserial.h"
#include "bot_control.h"

//dodano iza linije 113 programa u Prilogu C.
com = new Tserial();
  if (com != 0){
      com->connect("COM5", 9600, spNONE);
       oznaka_objekta(x, y, frame);
       if (x > 320 && y < 240) {
             putText(frame, "naprijed desno", Point(30, 50), 1, 1.1, Sca-
lar(255, 255, 255));
              com->sendChar('9');
              com->disconnect();
       }
       if (x > 320 && y > 240) {
              putText(frame, "nazad desno", Point(30, 50), 1, 1.1, Scalar(255,
255, 255));
              com->sendChar('3');
              com->disconnect();
       }
       if (x < 320 && y < 240) {
              putText(frame, "naprijed lijevo", Point(30, 50), 1, 1.1, Sca-
lar(255, 255, 255));
              com->sendChar('7');
              com->disconnect();
       }
       if (x < 320 && y > 240) {
              putText(frame, "nazad lijevo", Point(30, 50), 1, 1.1, Scalar(255,
255, 255));
              com->sendChar('1');
              com->disconnect();
       }
       if (y == 240 && x == 320) {
              putText(frame, "centriratno je", Point(30, 70), 1, 1.1, Sca-
lar(255, 255, 255));
              com->sendChar('0');
              com->disconnect();
       }
  }
```

PRILOG E. Ostvarena komunikacija za Arduino Uno

```
void setup(){
  Serial.begin(9600);
}

void loop(){
while (Serial.available()) {
  delay(3);
  c = Serial.read();
  switch (c) {
        case '9': // naprijed desno
             receiver_input_channel_4 = 1200;
             receiver_input_channel_2 = 1700;
             Serial.println("Naprijed desno");
             break;
        case '7': // narpijed lijevo
             receiver_input_channel_4 = 1200;
             receiver_input_channel_2 = 1200;
             Serial.println("Naprijed lijevo");
             break;
        case '3': // nazad desno
             receiver_input_channel_4 = 1700;
             receiver_input_channel_2 = 1700;
             Serial.println("Nazad desno");
             break;
        case '1': // nazad lijevo
             receiver_input_channel_4 = 1700;
             receiver_input_channel_2 = 1200;
             Serial.println("Nazad lijevo");
             break;
  }
  delay(50);
  }
}
```