Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Rezelj

Avtonomno pristajanje kvadrokopterja na mobilni platformi za samodejno polnjenje akumulatorja

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00043/2012
Datum: 13.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANŽE REZELJ**

Naslov: **AVTONOMNO PRISTAJANJE KVADROKOPTERJA NA MOBILNI PLATFORMI ZA SAMODEJNO POLNJENJE AKUMULATORJA**

AUTONOMOUS CHARGING OF A QUADCOPTER BY LANDING AT A MOBILE PLATFORM

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V mobilni robotiki ločimo dve veliki skupini inteligentnih robotov: avtonomna terenska vozila in avtonomna zračna plovila. Terenska vozila lahko nosijo težke akumulatorje, ki jim običajno omogočajo dolgo avtonomijo delovanja. Avtonomna zračna plovila so sicer veliko bolj fleksibilna v gibanju, saj lahko letijo po zraku, zato pa imajo zaradi zahteve po lahkih akumulatorjih precej omejen čas delovanja. V diplomski nalogi izdelajte strojno in programsko opremo za nizkocenovni kvadrokopter, ki mu bo omogočala avtonomno pristajanje na mobilni platformi. Pristane naj na način, ki bo omogočal avtomatski priklop na akumulator mobilne platforme in samodejno polnjenje akumulatorja kvadrokopterja. S kombinacijo obeh vozil se bo tako hkrati z večjo fleksibilnostjo gibanja zagotovil tudi daljši čas avtonomnega delovanja.

Mentor:

doc. dr. Danijel Skočaj

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anže Rezelj, z vpisno številko **63090172**, sem avtor diplomskega dela z naslovom:

Avtonomno pristajanje kvadrokopterja na mobilni platformi za samodejno polnjenje akumulatorja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. januarja 2013 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Danijelu Skočaju za strokovno pomoč in napotke pri izdelavi diplomske naloge. Zahvaljujem se tudi as. mag. Luki Čehovinu in doc. dr. Mateju Kristanu za podane predloge in napotke.

Posebna zahvala gre mojim staršem, ki so me tekom študija podpirali tako moralno kot finančno. Zahvaljujem se tudi bratoma in prijateljem za strpnost in moralno pomoč.

Kazalo

Povzetek

Avtonomna terenska vozila so opremljena z zmogljivimi baterijami, ki pa so zelo težka. Zaradi varčevanja s težo, so avtonomna zračna plovila opremljena z lahkimi in manj zmogljivimi baterijami.

Ta problem zasledimo tudi pri kvadrokopterju AR.Drone podjetja Parrot. Kvadrokopter je za razliko od helikopterja opremljen s štirimi rotorji, kar mu omogoča večjo stabilnost v zraku. Zaradi opremljenosti z dvema kamerama, vgrajene brezžične povezave in enostavnega upravljanja uživa veliko priljubljenost v akademskih krogih.

V diplomski nalogi se lotimo problematike avtonomije baterije. Problem poskušamo rešiti na način, da bi lahko avtonomna terenska vozila in kvadrokopter delovala v medsebojni simbiozi. V nalogi smo izdelali sistem, ki je sposoben samodejnega pristajanja in polnjenje baterije. Med pristajanjem si pomagamo z računalniškim vidom, s pomočjo katerega krmilimo kvadrokopter, da lahko pristane na mestu za polnjenje, kjer si baterijo nato tudi napolni. Za polnjenje smo izdelali tudi namensko krmilno vezje, ki krmili in nadzoruje polnjenje. Izdelali smo tudi pristajalno ploščad z mehanizmom, ki omogoča lažje pristajanje. Na koncu predlagamo še možnosti za kasnejše izboljšave.

Ključne besede: avtonomno pristajanje, samodejno polnjenje baterije, Parrot AR.Drone, kvadrokopter, mobilna robotika, računalniški vid

Abstract

Autonomous off-road vehicles are equipped with powerful but very heavy batteries. To save weight, the autonomous aircraft are equipped with less powerful batteries with less weight.

This problem is also found in quadrocopter AR.Drone from company Parrot. Quadrocopter is unlike as helicopter equipped with four rotors, which allows him greater stability in the air. Due to equipment with two cameras, built-in wireless and simple management enjoys great popularity in academic circles.

In this thesis we touch issue with battery life. The problem we try to solve is to enable autonomous off-road vehicles and quadrocopter to work in mutual symbiosis. We developed a system that is capable of landing and automatic battery charging. During landing, we use computer vision, through which quadrocopter is controlled that can land at the site for battery recharge. For recharging we created a dedicated control circuit that controls and monitors charging. We also implemented a landing platform with special landing mechanism for easier landing. Finally we suggest further options for future improvements.

Keywords: autonomous landing, autonomous charging, Parrot AR.Drone, quadrocopter, mobile robotics, computer vision

Poglavje 1

Uvod

Živimo v času, ko so računalniki, v takšni ali drugačni obliki, prisotni povsod okoli nas. Dnevno smo v kontaktu z namiznimi , prenosnimi in tabličnimi računalniki ter pametnimi telefoni. Manj zmogljive računalnike pa srečamo v televizorjih, štedilnikih, pralnih in pomivalnih strojih, v različnih prevoznih sredstvih in v igračah.

Napredek v tehnologiji pripomore do zmogljivejših, a varčnejših čipov in ostalih komponent po zelo dostopnih cenah. Velik napredek je tudi na področju lahkih in močnih materialov, kot so na primer karbonska vlakna. Spekter uporabe karbonskih vlaken je velik, saj so v primerjavi s kovino lažja a močna. Napredek na različnih področij pa privede do zmogljivejših, trpežnejših in cenejših izdelkov. Sledeče zasledimo tudi v robotiki, predvsem na področju mobilne robotike. V preteklosti je bila mobilna robotika omejena na premikanje po tleh ali v vodi, nebo pa je bilo skoraj nedosegljivo. Z razvojem zmogljivih brezkrtačnih elektromotorjev , ki imajo večji izkoristek delovanja kot krtačni motorji, zmogljivejših in lažjih hranilnikov energije, kot so na primer LiPo akumulatorji, lahkih materialov in elektronskih komponent je tudi nebo lažje dostopno. Veliko popularnost v mobilni robotiki uživajo kvadrokopterji (ang. quadrocopter), to so zračna plovila, ki delujejo podobno kot helikopterji, le da namesto enega rotorja uporabljajo štiri rotorje. Prednost kvadrokopterja pred helikopterjem je lažje upravljanje plovila in večja

robustnost, kar je v mobilni robotiki zelo zaželeno.

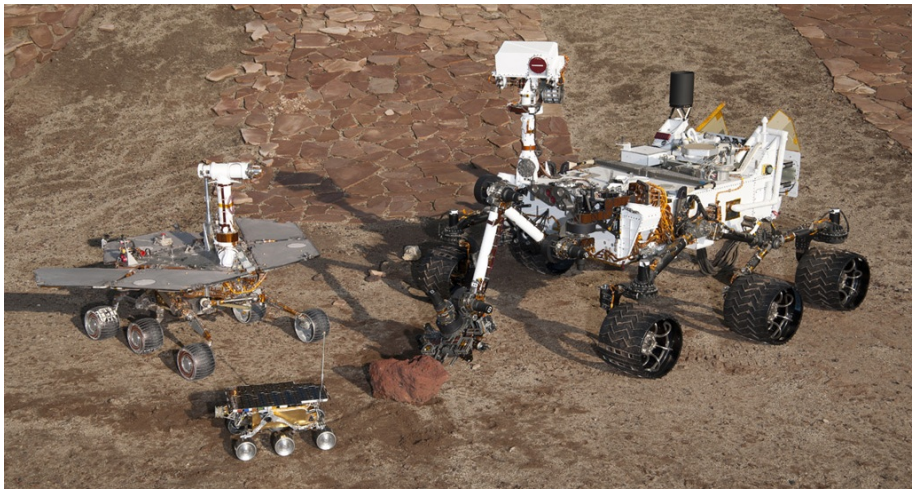
Zgodovina kvadrokopterjev se ni pričela bleščeče. Prvi preizkus leta s kvadrokopterjem sega v leto 1922, ko je George de Bothezat za United State Army Air Service [1] zgradil prvi kvadrokopter [2]. Zaradi zahtevnega in takrat omejenega krmiljenja, so leta 1924 projekt ukinili. Nadaljnji razvoj kvadrokopterjev se je pričel z množičnim razvojem brezpilotnih zračnih plovil ali angleško UAV (Unmanned Aerial Vehicle). Njegova glavna prednost pred letali s krili in helikopterji je visoka stabilnost v zraku, tako pri visokih in nizkih višinah, čemur pripomore vgrajena krmilna enota s senzorji. Zaradi dobre stabilnosti, manjših rotorjev in možnosti enostavne vgradnje zaščite za rotorje, lahko dostopajo tudi do težko dostopnih mest; lahko se premikajo celo po stanovanju. Oviro pri kvadrokopterjih predstavlja cena, ki pa z razvojem tehnologije pada in na trgu zasledimo veliko število različnih izvedb kvadrokopterjev.

Nadvse zanimiv izdelek je na prodajne police postavilo francosko Podjetje Parrot [24]. Čeprav je izdelek namenjen igranju, za svojo ceno ponuja veliko več. ARDrone [8] je igrača s štirimi rotorji, sprednjo in spodnjo kamero, višinomerom in vgrajenim majhnim računalnikom, ki poganja prilagojeno distribucijo Linux-a. Za krmiljenje se uporablja kar pametni mobilni telefon, lahko pa uporabimo tudi drugo napravo z brezžično (WiFi) povezavo. Zaradi enostavne, nezahtevne uporabe in izjemnega užitka med igranjem je ARDrone igranje ponesel na popolnoma nov nivo in postal zelo popularen. Svojo priljubljenost je pridobil tudi v akademskih krogih, kjer se ga uporablja v namene navigacije in na ostalih področjih raziskav s poudarkom na avtonomnosti plovil.

1.1 Motivacija

V zadnjih 400 letih je bil napredek na področju tehnologije, znanosti in ostalih področjih zelo izrazit. Galileo Galilei je bil prvi, ki je s pomočjo teleskopa podrobneje preučil luno in ostale planete v našem osončju. Sir Isaac New-

ton nam je v fiziki podal tri zakone in razložil zakaj luna ne pade z neba. Odkritja in razvoj elektromagnetizma in polprevodnikov, splošna in posebna teorija relativnosti ter ostala pomembna odkritja so pripomogla, da je v manj kot 400 letih, ko je Galileo Galilei teleskop usmeril proti luni, človek stopil na luno. Od takrat v vesolje z namenom raziskovanja pošiljamo satelite in sonde. Poleg velike zanesljivosti sistemov, ti potrebujejo tudi zanesljiv in konstanten vir energije. Kjer je to možno, se za vir uporablja sončne celice. Ker vira svetlobe vedno ni na voljo in, ker imajo sončne celice nizko zmogljivost je potrebno uporabiti tudi druge vire energije. Vesoljska agencija NASA je v ta namen robotsko vozilo Curiosity opremila z baterijami in s termoelektričnim generatorjem z radioaktivnim izotopom (Radioisotope Thermoelectric Generator ali RTG), ki bo dovajal potrebno energijo vsaj 14 let [3]. Zaradi zmogljivejšega vira energija je opremljen z večjim naborom orodij, zaradi česar je tudi velik, kar lahko razberemo s slike 1.1.



Slika 1.1: Testni modeli sond za raziskovanje Marsa in od leve proti desni predstavljajo Spirit in Opportunity, Sojourner (najmanjši) ter Curiosity. Slika povzeta po [4].

ARDrone za vir napajanja uporablja le eno baterijo. Uporaba sončnih celic, kot dodatni vir energije je mogoč a nepraktičen, saj ima AR.Drone

premalo površine, kamor bi lahko vgradili dovolj veliko število sončnih celic, da bi te opazno podaljšale avtonomijo letenja. Tudi vgradnja raznih vrst generatorjev ni smiselna, saj so ti težki in bi otežili letenje, če bi ARDrone le lahko vzletel. Zato je trenutno edini smiselni vir energije le baterija. Baterija v ARDronu je vrste LiPo, sestavljena iz treh členov, vsak člen ima nazivno napetost 3,7V, kar skupaj znese 11,1V, kapaciteta baterije pa znaša 1000mA. Pri zmerni uporabi baterija zadostuje za do 12 minut letenja, nato pa jo je potrebno napolniti. Napolnimo jo tako, da odstranimo pokrov ARDrona, odklopimo in odstranimo baterijo ter jo priklopimo na originalni polnilec, ki je priložen ob nakupu ARDrona. Za napajanje originalnega polnilnika potrebujemo vir enosmerne napetosti med 13V in 18V, zmogljivosti do 1,0A. V ta namen je priložen adapter, ki na vhodni strani potrebuje vir izmenične napetosti od 100V do 240V, frekvence 50 ali 60Hz, napetost na izhodu polnilca znaša 13,5V enosmerne napetosti, pri toku do 0,9A. Polnjenje baterije nato znaša do 90 minut [5].

Ideja je sledeča. ARdrone mora pred polnjenjem vedno pristati. Pristal bi lahko na posebni ploščadi, na kateri bi bili prisotni priključki za polnjenje in polnilnik. Proces pristajanja lahko avtomatiziramo in bi ARDrone pristal čisto samodejno. Pristajalna ploščad se lahko nahaja na nepremični površini ali pa je vgrajena na drugi mobilni platformi. Ta mobilna platforma je lahko osebno vozilo, mobilni robot ali kaj drugega. V ta namen moramo upoštevati omejen vir napetosti. Omejimo se na 12V enosmerne napetosti, ki je v mobilnih platformah zelo razširjen. Ob pristanku bi se polnjenje pričelo samodejno, brez odstranjevanja baterije iz ARDrona. Ko bi se baterija napolnila, pa bi bil ARDrone pripravljen za uporabo.

Praktična uporaba samodejnega pristajanja in polnjenja je velika. Namesto WiFi povezave lahko kvadrokopter opremimo z RC krmiljenjem in močnejšim oddajnikom za prenašanje videa, ter tako povečamo domet na 200m ali 500m [6], z boljšo opremo pa lahko dosežemo tudi nekaj 10km [7]. ARDrone opremljen s plazovno žolno lahko nato išče v snežnem plazu zasute osebe. Prednost uporabe zračnega plovila je, da leti nizko nad plaziščem

Slika 1.2: AR.Drone v času letenja.

ter mu teren ne predstavlja velike ovire za gibanje. Posledično veliko hitreje preišče večjo površino in hitreje najde v plazu zasute osebe, lokacijo pa nato sporoči reševalni ekipi. Naslednja možnost uporabe je tudi v gorah. Z zmogljivejšo sprednjo in spodnjo kamero ter ustrezno programsko opremo za prepoznavanje oseb bi lahko iskal v gorah izgubljene osebe, ali poškodovane osebe, ki ob klicu pomoči ne vedo kje točno se nahajajo. V obeh primerih je primerna hitra odzivnost reševalnega osebja, ki pa, čeprav so vedno v pripravljenosti, potrebujejo nekaj minut, da se razmeram primerno opremijo. Kvadrokopter lahko s pritiskom na gumb vzleti na dane koordinate, kjer nato iskanje opravi sam ali z navzočnostjo kontrolorja. Oviro predstavlja le slabo vreme, v katerem kvadrokopter ne more leteti. V primeru gozdnih ali travniških požarov ali preostalih naravnih nesrečah, bi kvadrokopter, nameščen na gasilnem ali reševalnem vozilu, letel nad krajem nesreče in reševalce obveščal s svežimi podatki in svežo sliko o kraju nesreče.

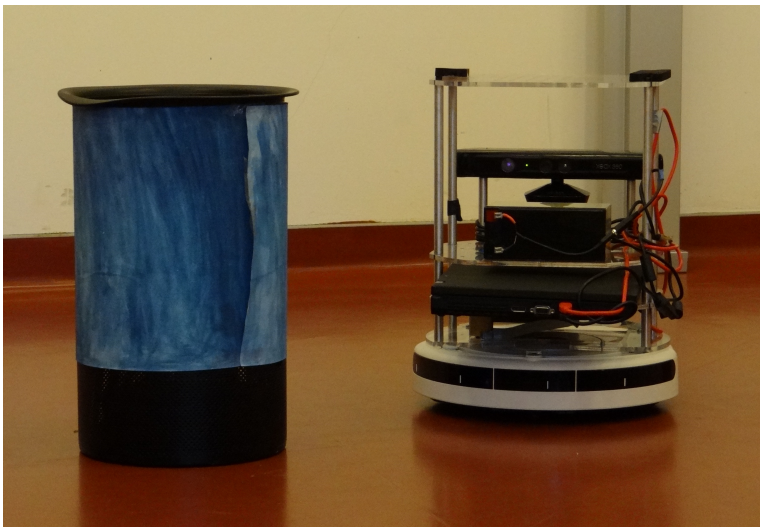
Bolj splošno opisano, mobilni robot, v mislih imamo robota, ki je omejen na gibanje po tleh, ima lahko vgrajene različne vire energije. Na primer, lahko je oblikovan tako, da ima čim večjo površino, na katero so pritrjene sončne celice. Pri takih robotih si to lahko privoščimo, saj oblika robota ni vedno striktno omejena. Dodamo lahko tudi eno ali več vrst agregatov, saj si lahko privoščimo malo težjega robota. Vsi viri energije napajajo baterije. Potreba po baterijah je obvezna, saj so viri energije navzgor omejeni, poraba energije pa ni konstantna in je lahko trenutna potreba po energiji večja, kot pa so jo zmožni viri dovesti. V času nizke zahteve po energiji, pa se vsa neuporabljena energija shrani za kasnejšo uporabo. Kvadrokopter nato to zalogo energije uporabi za polnjenje svoje baterije. Da bi bila simbioza med robotom in kvadrokopterjem popolna, lahko kvadrokopter dopolnjuje ali celo opravi opravila, ki jih mobilni robot ne more. To so na primer, prinese vzorec težko dostopnih predelov, kot na primer navpična stena, pečine ali dovolj velike razpoke in robotu prinese vzorec za testiranje. Omenjene predele lahko tudi samo posname in dodela obstoječi zemljevid ali pa za mobilni robot zgradi nov zemljevid.

1.2 Cilj diplomske naloge

Cilj diplomske naloge je razviti strojno in programsko opremo, ki bi omogočala avtonomno pristajanje in avtonomno polnjenje vgrajene baterije. Problem postane še bolj zanimiv, saj je načinov za dosego želenega cilja veliko, ravno tako pa je potrebno znanje iz večjega števila področij. Pri pristajanju moramo vedeti, kje mora ARDrone pristati, oziroma, kje je pristajalna ploščad. V ta namen si pomagamo z računalniškim vidom in eno izmed obeh kamer. Za polnjenje baterije potrebujemo ustrezen polnilnik in logiko, da trajno ne poškodujemo baterijo ali celo uničimo AR.Drone.

Pristajalno ploščad in krmilno logiko bomo vgradili v TurtleBot [45]. TurtleBot je odprtokodni projekt za razvoj avtonomnih vozil. Slika 1.3 prikazuje iRobot Roombo [46] z nadgradnjo TurtleBot in prenosnim računalnikom.

Avtonomni robot in kvadrokopter bosta nato lahko skupaj reševala naloge.

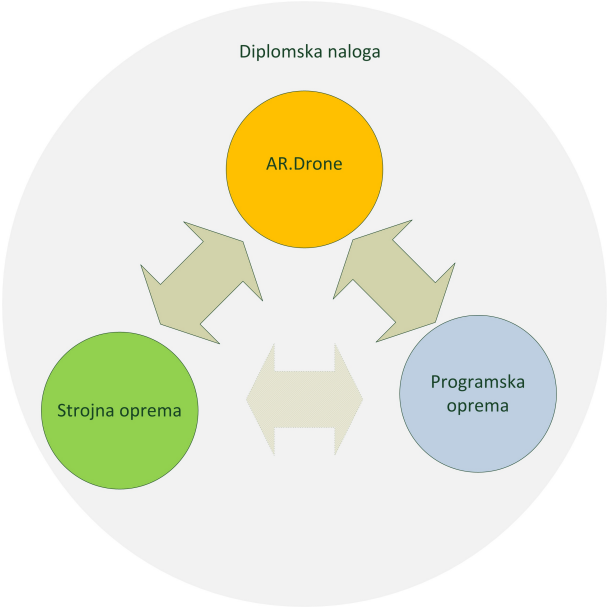


Slika 1.3: Mobilni robot iRobot Roomba s TutleBot nadgradnjo pri izoogibanju ovire.

1.3 Zgradba sistema za avtonomno pristajanje in avtonomno polnjenje baterije

Diplomsko nalogo smo v grobem razdelili na dva sklopa. Prvi sklop predstavlja strojna oprema, drugi del pa predstavlja programska oprema. Oba sklopa med seboj nista direktno povezana, saj rešujeta različne probleme. Prvi sklop, strojna oprema, poskrbi za ustrezno okolje v katerem lahko AR.Drone pristane in si napolni baterije ter nato nemoteno vzleti. Drugi sklop, programska oprema, poskrbi za krmiljenje AR.Drone med pristajanjem.

Slika 1.4 predstavlja zgradbo zgoraj opisane delitve diplomske naloge. Vsak od krogov ponazarja posamezen sklop, omenili pa smo že prvi sklop – strojna oprema – ter drugi sklop – programska oprema. Nič pa nismo omenili sklop AR:Drone. Ta sklop ponazarja kvadrokopter. Puščice med sklopi ponazarjajo medsebojno povezanost. Puščica, ki povezuje sklop strojna oprema



Slika 1.4: Grafični prikaz delitve diplome na posamezne sklope

in sklop programska oprema, je od preostalih dveh manj opazna ker pri izdelavi diplomske naloge nismo zasledili potrebe po komunikaciji med strojno opremo – mikrokrmilnikom – in programsko opremo.

1.4 Zgradba diplomske naloge

Preostanek naloge je razdeljen na šest poglavij. V poglavju 3 je predstavljena mobilna platforma Parrot AR.Drone. Opisane so tehnične specifikacije kvadrokopterja. Dotaknili se bomo tudi razvojnega okolja, s katerim lahko razvijamo poljubne aplikacije za AR.Drone.

V poglavju 3 govorimo o strojni opremi, ki smo jo izdelali. Opisali smo kaj vse predstavlja strojna oprema in kaj opravlja. Pri pristajalni ploščadi opišemo, kaj je njena naloga, kakšne omejitve se pri tem pojavijo in kako smo jih rešili. Podobno storimo tudi za krmilno logiko, kjer opišemo posamezne komponente, njihove naloge in omejitve.

V poglavju 4 opisujemo programsko opremo. Opišemo uporabljeno pro-

gramsko opremo, kako deluje oziroma kaj smo morali vedeti, da smo jo lahko uspešno uporabili. Opišemo delovanje izbrane programske opreme za vizualno zaznavanje, za kaj smo jo izbrali in kako deluje algoritem za zaznavanje oznak ter opišemo postopek kalibracije kamere. Nato razložimo krmiljenje kvadrokopterja s pomočjo informacije o lokaciji oznake. Predstavimo grafični vmesnik, kaj je njegova naloga in kaj prikazuje. Sledi še opis težav, s katerimi smo se srečali pri krmiljenju med pristajanjem kvadrokopterja.

Sledi še poglavje 5, v katerem smo strnili ugotovitve v sklep in navedli možnosti za izboljšavo sistema. Čisto na koncu so v prilogi dodane tehnične risbe pristajalne ploščadi in shematski načrti elektronskih vezij.

Poglavje 2

Mobilna platforma AR.Drone

V sledečem poglavju je podrobneje predstavljena mobilna platforma AR.Drone. V poglavju se dotaknemo zgradbe kvadrokopterja, poglobimo se v njegovo strojno opremo in zmogljivost. Sledi kratek opis razvojnega okolja, na podlagi katerega lahko zgradimo poljubno aplikacijo.

2.1 Tehnične specifikacije

AR.Drone je radijsko krmiljen kvadrokopter, katerega je razvilo francosko podjetje Parrot [24]. Prvotno je bil načrtovan za krmiljenje z napravami z nameščen iOS (iPhone, iPad in iPod Touch), kasneje pa so prišle aplikacije za Android, uradna podpora, ter operacijski sistemi Samsung BADA in Symbian, ki trenutno nimata uradne podpore. Za krmiljenje AR.Drona mobilna naprava potrebuje Wi-Fi povezavo, zaslon na dotik in senzorje nagiba. Javnosti je bil predstavljen leta 2010 v Las Vegasu na International Consumer Electronics Show (CES) [9] . Parrot je nekoliko kasneje javnosti predstavil tudi razvojno okolje [25] za različne platforme ter tako odprl razvijanje programske opreme javnosti. Na trg je leta 2012 prišla novejša različica, AR.Drone 2.0, z malo nadgrajeno strojno opremo. Prvi model kvadrokopterja prikazuje slika 2.1.

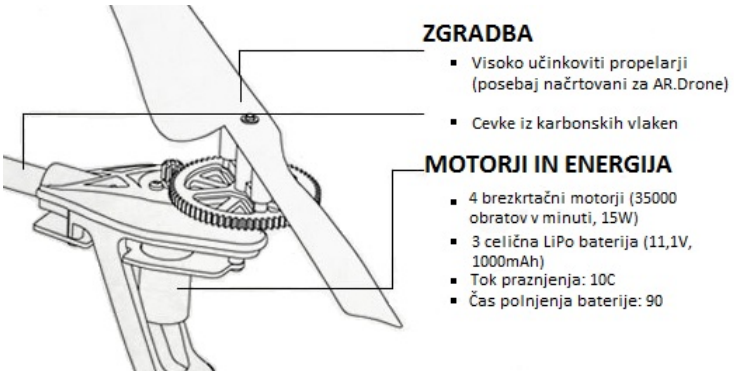
AR.Drone je prava tehnološka igrača saj združuje tako modelarstvo, ra-

Slika 2.1: Kvadrokopter s pokrovom z zaščito (levo) in s pokrovom brez zaščite (desno).

čunalništvo kot tudi elektrotehniko . Vgrajen ima centralno procesno enoto ARM9 s 468 MHz, 128 MB RAM-a in poganja prilagojeno različico operacijskega sistema Linux. Povezavo med mobilno napravo in AR.Drone vzpostavimo preko Wi-Fi povezave. Trenutno sta podprta tipa 802.11b/g. Teža kvadrokopterja pripravljenega za letenje z ogrodjem za notranjo uporabo z zaščito za rotorje, znaša 420g, z ogrodjem za zunanjo uporabo - brez zaščitepa znaša 380g. Dimenzije z zaščito znašajo $52,5 \times 51,5cm$ ter brez zaščite $45 \times 29cm$. Za boljšo predstavo in razumevanje je nekaj specifikacij navedenih na slikah 2.2, 2.3 in 2.4.

Pri sliki 2.2 je omenjen tok praznjenja baterije, ki znaša $10C$, kar je na prvi pogled nesmiselno. Predpona C stoji za kapaciteto (ang. Capacity) baterije, ki znaša $1000mA$ oziroma $1A$. Tok praznjenja znaša $10 \times 1A$ kar je $10A$. Čas polnjenja znaša 90 minut pri uporabi originalnega polnilnika, pri uporabi alternativnega polnilnika pa moramo biti previdni na polnilni tok, ki je lahko največ $1C$ oziroma $1A$.

Kvadrokopter preko brezžične povezave pošilja tako barvno sliko horizontalne kamere z resolucijo 640×480 slikovnih točk, kot tudi barvno sliko vertikalne kamere resolucije 176×144 slikovnih točk. Na sprejemni strani lahko nato izbiramo s katere kamere bomo spremljali video.



Slika 2.2: Opis osnovne zgradbe in elektronskih komponent. Slika povzeta po [5].

Slika 2.3: Prikaz in opis horizontalne kamere. Slika povzeta po [5].

Slika 2.4: Specifikacije in prikaz vertikalne kamere ter višinomera. Slika povzeta po [5].

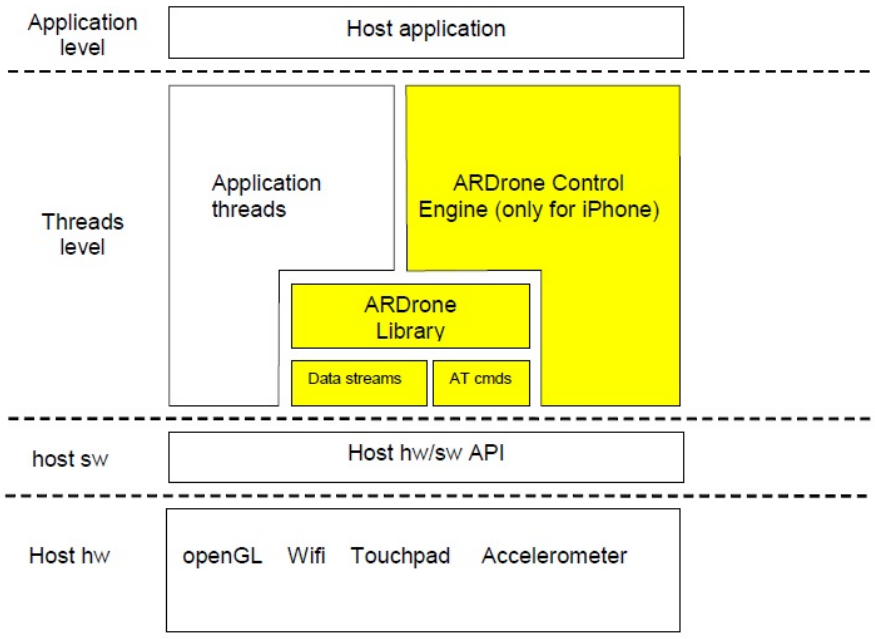
2.2 Razvojno okolje za AR.Drone

Poleg uradnih aplikacij za pametne mobilne telefone in tablične računalnike je podjetje Parrot javnosti ponudilo tudi razvojno okolje za računalnike. Razvojno okolje lahko prenesemo s spletne strani [25] in nam omogoča, da se na kvadrokopter povežemo prenosnim računalnikom ali osebnim računalnikom z WiFi povezavo. Nudi nam tudi osnovo, na kateri lahko ustvarimo lastno aplikacijo. Podrobno shemo zgradbe razvojnega okolja predstavlja slika 2.5. Najvišji nivo, imenovan »aplikacijska ravn« (ang. application level), predstavlja nivo kjer se nahaja uporabniška aplikacija. Tu lahko uporabnik napiše aplikacijo po svoji želji. Nivo nižje, imenovan »nivo niti« (ang. threads level), predstavlja jedro razvojnega okolja. V jedru se nahajajo vse glavne rutine in podrutine ter knjižnice potrebne za komunikacijo med računalnikom in kvadrokopterjem. Dodamo lahko tudi svojo podrutino ali prilagodimo obstoječo. Naslednji nivo, imenovan »programska oprema gostitelja« (ang. host sw), predstavlja programsko opremo, ki je nameščena na kvadrokopterju. Zadnji in najnižji nivo, imenovan »strojna oprema gostitelja« (ang. host hw), predstavlja strojno opremo na kvadrokopterju. Shema na sliki 2.5 ponazarja tudi po katerih nivojih potujejo podatki za krmiljenje in olajša razumevanje principa delovanja razvojnega okolja.

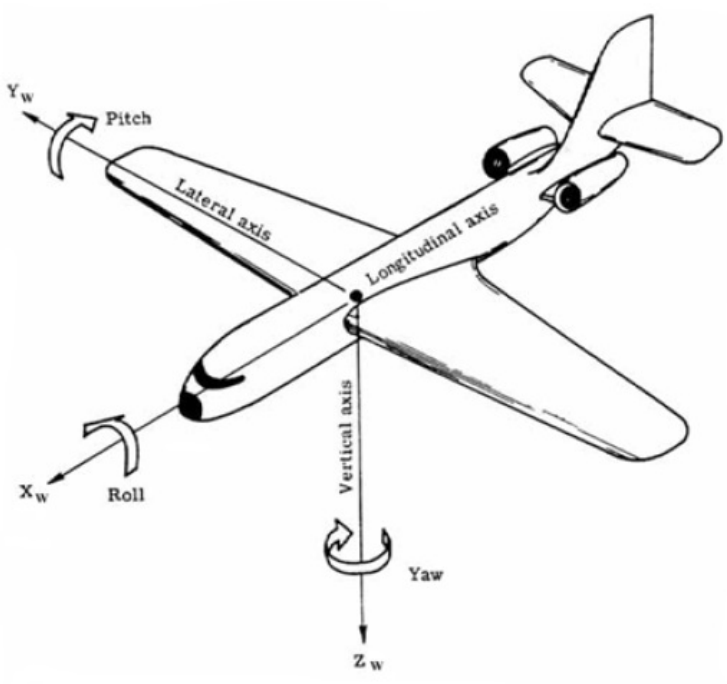
Pri izdelavi aplikacije nam razvojno okolje zelo olajša delo. Vsebuje že implementirane rutine za vzpostavitev komunikacijskega kanala s kvadrokopterjem, dekodira video vsebino, ki jo neprestano pošilja kvadrokopter, dekodiran video pa lahko nato poljubno obdelujemo. Tudi krmiljenje je enostavno, vse kar moramo storiti je, da v aplikaciji kličemo funkcijo:

ardrone_at_set_progress_cmd(int flag, float roll, float pitch, float gaz, float yaw);

Funkciji podamo argumente v obliki števila s plavajočo vejico in lahko zavzema vrednosti med -1 in 1. Podano število predstavlja del največjega dovoljenega premika. S pošiljanjem različnih vrednosti argumentov lahko



Slika 2.5: Shema nivojev razvojnega okolja in kvadrokopterja. Povzeto po [26].



Slika 2.6: Prikaz treh osi, na katere lahko vplivamo s pošiljanjem primernega ukaza. Povzeto po [26].

sestavimo različna gibanja kvadrokopterja. Kako podani argumenti vplivajo na premik kvadrokopterja prikazuje slika 2.6. Vrednost atributa gaz vpliva na vertikalno hitrost.

Poglavje 3

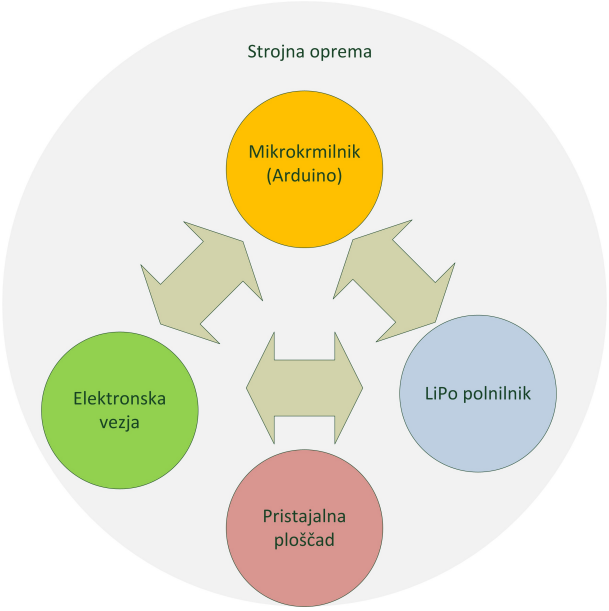
Strojna oprema

V sledečem poglavju sledi predstavitev koncepta in opis strojne opreme. Podrobneje so predstavljeni posamezni sklopi, kaj so njihove naloge ter kako delujejo. Predstavljene rešitve so koncept, predstavljajo neko osnovo, ki pa jo je možno nadgraditi ali celo zamenjati. Ponujajo vpogled na kaj vse moramo biti pozorni, kje so kritične točke za morebitne težave in kako bi jih lahko odpravili.

3.1 Zasnova strojne opreme

Osnovna zgradba strojne opreme je enostavna. Potrebujemo pristajalno ploščad, ki bo omogočala da je AR.Drone ob pristanku vedno na istem mestu in da je med polnjenjem stabilen, torej se ne more premakniti. Hkrati pa le ta ne sme ovirati pristajanja in vzletanja. Naslednji del opreme predstavlja krmilna logika, ki poskrbi za preverjanje prisotnosti AR.Drona, merjenje napetosti baterije in vklop ter izklop polnilnika baterije. Ostane še polnilnik, ki poskrbi za ustrezno polnjenje baterije.

Slika 3.1 prikazuje delitev strojne opreme na posamezne sklope. Dva sklopa, pristajalna ploščad in LiPo pomnilnik smo že omenili. Preostala sklopa, elektronska vezja in mikrokrmilnik pa skupaj predstavljata krmilno logiko. Za ločitev smo se odločili, ker mikrokrmilnik ni dovolj zmogljiv da



Slika 3.1: Prikaz delitve sklopa strojne opreme na posamezne podsklope.

bi neposredno krmilil tokovno zmogljivejše porabnike, kar nadomestimo z elektronskimi vezji in ustreznimi komponentami.

Sklopi so sestavljeni modularno. Vsak sklop sestoji iz vsaj enega modula. Modularna zgradba je uporabljena zaradi lažjega odkrivanja morebitnih napak ali okvar. Omogoča tudi enostavno zamenjavo posameznih modulov ali njihovo nadgradnjo, ne da bi pri tem znatno vplivali na ostale module. Za primer lahko navedemo, da lahko uporabimo katerikoli mikrokrmilnik, le da ima na razpolago dovolj ustreznih priključnih nožic.

3.2 Pristajalna ploščad

Kot že ime pove, je pristajalna ploščad namenjena pristajanju in tudi vzletanju AR.Drona. Pri tem le ta ne sme v nikakršni obliki ovirati tako pristajanje kot vzletanje. Pri pristajanju je potrebno biti še posebej pozoren na to, da AR.Drone vedno malo niha – se premika naprej, nazaj, levo in desno – kar ni zaželeno. To nihanje je sorazmerno majhno a je vedno prisotno in lahko po-

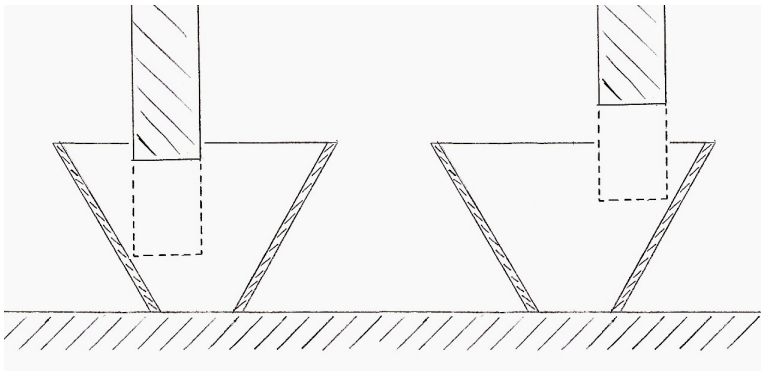
meni neuspeh pri pristajanju. Da nihanje med pristajanjem čim manj vpliva na uspešnost pristajanja, je potrebno sestaviti ustrezen mehanizem. Pri izdelavi ustreznega mehanizma moramo upoštevati tudi, da je kvadrokopter v času neaktivnosti ali v času polnjenja baterije prisoten na mobilni platformi, ki pa se lahko v tem času prosto giba po prostoru. Teren, po katerem se premika mobilna platforma pa je lahko zelo razgiban in lahko mobilna platforma naleti na ovire ki povzročijo nenaden nagib tako mobilne platforme kot tudi pristajalne ploščadi in lahko pride do zdrsa kvadrokopterja s ploščadi.

Problem nihanja med pristajanjem in možnost zdrsa ob nenadnem nagibu smo rešili z votlim stožcem. Ideja je sledeča. Na pristajalno ploščad postavimo štiri stožce, za vsak krak AR.Drona en stožec. Vsak stožec je obrnjen na glavo, tako da je najširši del stožca na vrhu, najožji del stožca pa je pritrjen na pristajalno ploščad. Ker je najširši del stožca obrnjen proti kvadrokopterju, natančneje proti kraku kvadrokopterja in je stožec votel, se verjetnost, da bo AR.Drone z nožico zadel odprtino v stožcu poveča. Zaradi poševne stene stožca nožica kvadrokopterja po steni drsi navzdol in na koncu vedno pristala na sredini stožca. Najožji del stožca se natanko ujema z velikostjo pristajalne nogice kvadrokopterja. Na ta način tudi preprečimo, da bi ob nagibu mobilne platforme kvadrokopter zdrsnil s ploščadi.

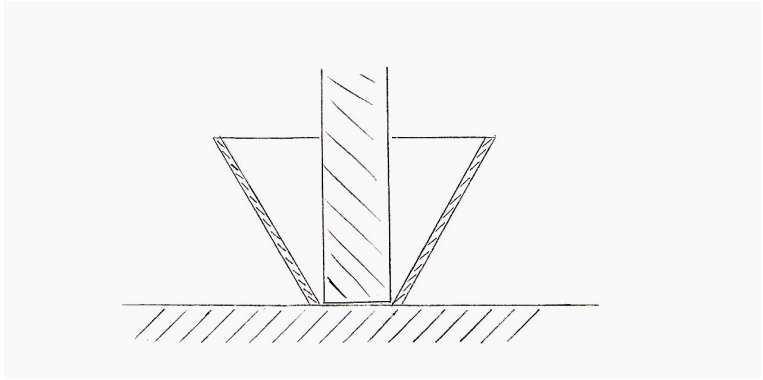
Sliki 3.2 in 3.3 nazorno prikazujeta princip delovanja ideje. V primerih so že odstranjene konice stožca. Najožji del stožca se ujema s širino pristajalne noge kvadrokopterja. Tako dosežemo da je pristajalna noga vedno na istem mestu in se le ta ne mora premikati. Na ta način dosežemo tudi, da je lahko pristajalna ploščad med pristajanjem pod kotom in bo lahko kvadrokopter še vedno pristal ter si napolnil baterije. Še več, ploščad bi se med pristajanjem lahko premikala, seveda če lahko kvadrokopter krmilimo dovolj natančno.

Slika 3.4 prikazuje krak kvadrokopterja, ko je le ta že pristal. Odprtina pri stožcu je potrebna, da lahko podaljšamo stranico stožca in tako povečamo prostor za pristajanje. Obseg najširšega dela stožca lahko povečamo tudi tako, da povečamo naklon stranice in pri tem ohranimo višino stožca.

Pristajalna ploščad je v obliki kvadrata, s stranicami dolžine $38cm$. Služi



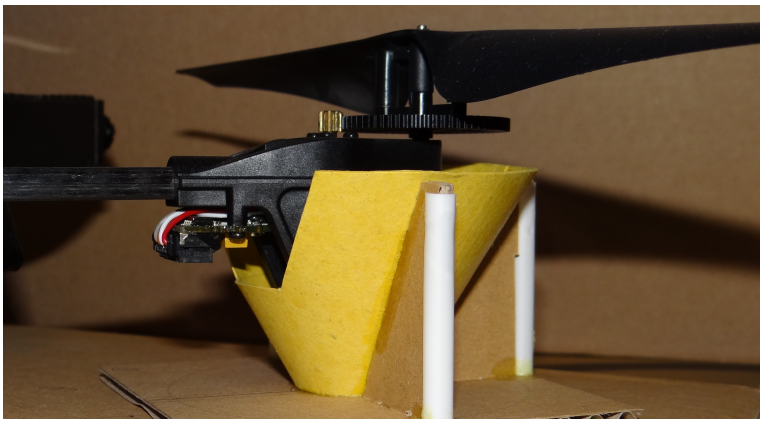
Slika 3.2: Prikaz ideje in dveh mogočih dogodkov med pristajanjem.



Slika 3.3: Pristajalna noga se vedno nahaja na dnu, kjer ima omejeno premikanje.

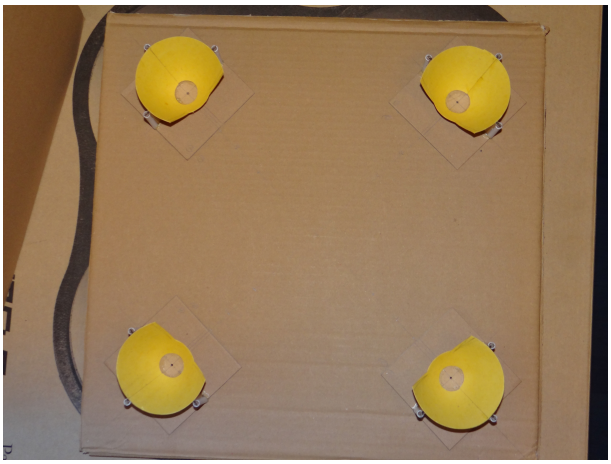
kot platforma na katero namestimo stožce, oznako za orientacijo in osvetlitev oznake ter potrebne kontakte za polnjenje baterije. Slika 3.5 prikazuje pristajalno ploščad, na kateri so že postavljeni vsi štirje stožci.

Osvetlitev oznake ni nujno potrebna, toda odpravi marsikatero nevšečnost, ki se lahko prigodi v času pristajanja. V primeru vira svetlobe direktno nad kvadrokopterjem povzroči, da senca kvadrokopterja med pristajanjem postopoma zakriva oznako. To privede do neenakomerne osvetlitve oznake in posledično odpoved prepoznavanja oznake. Problem je še izrazitejši v primeru, ko je v prostoru več virov svetlobe različnih intenzitet in se med virom



Slika 3.4: Končno izdelani stožec z zarezo in podporniki.

svetlobe in oznake nahaja ovira, ki delno zakriva snop svetlobe. Uspešnost prepoznavanja oznake se znatno zmanjša tudi v primeru slabotne osvetlitve. V ta namen smo se odločili za uporabo dodatne osvetlitve. Tu se pojavi več možnosti. Najenostavnejša rešitev je vgradnja osvetlitve na spodnjo stran kvadrokopterja tako, da le ta ne moti kamere ali ultrazvočnega merilnika višine. Rešitev je enostavna in hitro izvedljiva ima le dve slabosti. Za vir energije bi uporabili baterijo kvadrokopterja ali dodali novo baterijo in dodatno obtežili kvadrokopter. Naslednja slabost je, da bi bila osvetlitev v času aktivnosti vedno vklopljena. Slednje bi se dalo rešiti z globljim posegom v programsko opremo nameščeno na kvadrokoptorju in vklapljati osvetlitev samodejno. Lahko bi sami izdelali majhno vezje, ki bi vklopilo osvetlitev če pade napetost baterije v ustrezno območje. Naslednja možnost je pritrditev dodatne osvetlitve na pristajalno ploščad, oznaka pa bi bila osvetljena s strani. Tu omejitev predstavlja kvadrokopter, ker če pristane in ima nameščeno ogrodje za letenje v prostorih, na ploščadi ni dovolj prostora za vgraditev osvetlitve s strani. Naslednja ideja je zelo zanimiva in jo srečamo vsakič, ko pogledamo v LCD ekran. Ideja je bila sestaviti osvetlitev, ki uporablja vse prednosti osvetljevanja LCD ekranov [42], [43] in [44]. Prednost, v primerjavi s predhodno opisanima idejama, je enakomerna osvetlitev po celotni površini, ker pa v debelino meri le približno centimeter,

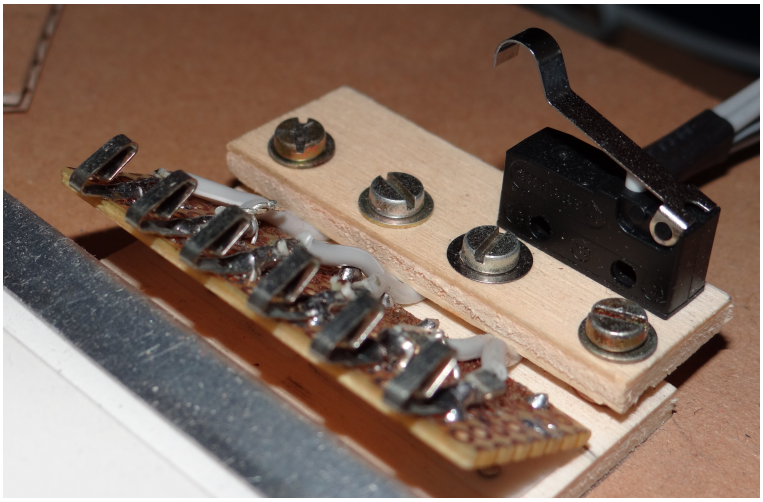


Slika 3.5: Pristajalna ploščad z vgrajenimi vsemi štirimi stožci.

ga lahko vgradimo direktno pod kvadrokopter in pri tem ne moti ne vzletanje ne pristajanje. Za vir svetlobe smo uporabili svetleče diode skupne zmogljivosti 2,8W pri napetosti 12V. Vgrajene ima 4 držalne ročice, ki pritiskajo oznako na osvetljeno podlago, da jo veter, ustvarjen med letenjem nad oznako, ne premakne ali celo odpihne. Tovrstna pridržanja oznake nam dovoljuje, da lahko uporabimo oznake različne velikosti, natisnjene na papir, prozorno folijo ali izrezane iz kartona. Končni izdelek predstavlja slika 3.6.

Slika 3.6: Uporabljen način osvetlitve oznake.

Za kontakt s kvadrokopterjem s strani pristajalne ploščadi smo uporabili kontakte, ki se ob pritisku nanje upognejo. Tako dosežemo, da se ob pristanku kvadrokopterja vsi kontakti dotikajo kontaktne površine. Kontakti so prikazani na sliki 3.7 in od leve proti desni predstavljajo (–) in (+) za odklop baterije, nato sledi (−) baterije, (+) napetost celice 1, (+) napetost celice 2 in (+) napetost celice 3. Na sliki 3.7 je prikazan tudi stikalo. Ob pristanku kvadrokopterja se stikalo sklene in tako krmilna logika ve, da je kvadrokopter pristal.



Slika 3.7: Stikalo za preverjanje prisotnosti kvadrokopterja in kontakti za polnjenje baterije ter odklop le te.

3.3 Krmilna logika

Omenili smo že, da je naloga krmilne logike preverjanje napetosti baterije, vklapljanje in izklapljanje polnilnika ter preverjanje prisotnosti kvadrokopterja na pristajalni ploščadi. To pa še ni vse. Za uspešno delovanje mora krmilna logika znati veliko več.

Podrobneje opišimo okolje, v katerem bo delovala krmilna logika. AR.Drone se v času polnjenja nahaja na pristajalni ploščadi, ta pa se nahaja na mobilni

platformi. V našem primeru je to TurtleBot, lahko pa bi bil tudi avto ali kako drugo prevozno sredstvo. To nam pove, da se bo vsa elektronika napajala iz omejenega vira napetosti. Bolj točno, Turtlebot ima vgrajen dodatni 12V svinčev akumulator. Pomanjkljivost vseh shranjevalnikov energije je, da imajo omejeno kapaciteto. Naloga krmilne logike je, da preverja napetost akumulatorja ter obvešča o njegovem stanju. Poleg tega v primeru nizke napetosti akumulatorja vklop polnilnika LiPo baterije ni priporočljiv, saj se lahko polnilnik zaradi prenizke napetosti akumulatorja izklopi preden je polnjenje LiPo baterije zaključeno. Ker se akumulator polni ročno, se njegovo stanje napetosti signalizira preko svetleče diode (ang. LED). Odvisno od stanja napetosti svetleča dioda utripa različno hitro in sicer, če je akumulator poln utripa s periodo 2 sekundi, ko je že malo izpraznjen utripa s periodo 1s, ko pa je prazen utripa s periodo 0,5 sekunde in v tem primeru se prepreči tudi vklop polnilnika baterije.

Poleg predhodno opisanih omejitev, mora krmilna logika vedeti tudi, da je kvadrokopter pristal. To storimo tako, da na pristajalno ploščad vgradimo senzor. V našem primeru bo to tipka, ki sklene tokokrog, ko kvadrokopter pristane. Ko vemo, da je kvadrokopter pristal, lahko preverimo napetost baterije. Napetost baterije preverjamo z razlogom. Če bi vedno takoj vklopili polnilec, čeprav je baterija skoraj polna, bi ji zmanjšali življenjsko dobo in bi jo morali kmalu zamenjati. Da se temu izognemo torej najprej preverimo napetost. Ponovno lahko pride do več možnosti. V primeru da lahko izmerimo napetost lahko pride do treh primerov. V prvem primeru je baterija lahko dovolj polna in polnjenje ni potrebno. V drugem primeru je napetost dovolj nizka da je smiselno vklopiti polnilec. V tretjem primeru je napetost prenizka, kar nakazuje na to, da je baterija poškodovana – ali iztrošena – in jo je potrebno zamenjati, polnjenje pa ni priporočljivo in se tudi ne vklopi. Vsako od teh stanj ponazarja dodatna LED dioda, ki utripa v intervalu 2, 1 in 0,5 sekunde – vrstni red ustreza predhodnim trem opisom stanj baterije. V primeru, da ne moremo razbrati napetosti baterije, kljub temu, da je tipka sklenjena, pomeni, da je kvadrokopter pristal napačno ali pa da je prišlo do

kake druge težave – slab kontakt kvadrokopterja s priključki na pristajalni plošči. V tem primeru začne LED dioda utripati v intervalu 0,5 sekunde. Signaliziranje je pomembno, da lahko hitro odstranimo vzrok napake.

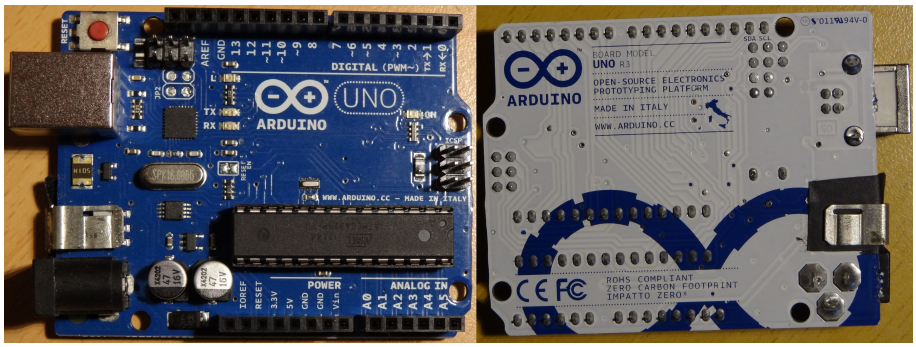
V primeru, ko je baterija premalo prazna, da bi krmilna logika vklopila polnilec, hkrati pa je preveč prazna, da bi kvadrokopter lahko opravil zadano nalogo, dodamo tipko. Naloga tipke je, da lahko ročno vklopimo polnjenje, kar storimo tako, da jo neprekinjeno tiščimo 5 sekund. V času tiščanja svetleča dioda, ki ponazarja stanje LiPo baterije, petkrat utripne. Tako lahko aktiviramo polnjenje tudi v primeru, ko je napetost akumulatorja pod priporočljivo mejo uporabe.

Sedaj je polnilec baterije vklopljen in polni baterijo kvadrokopterja. Toda, kdaj vemo da je polnjenje končano, da lahko polnilnik izklopimo. Uporabimo enostaven polnilec, ki svoja stanja signalizira preko svetlečih diod, ter s krmilno logiko preverjamo stanja teh diod. Ko razberemo, da je polnilec končal s polnjenjem ga nato izklopimo. Tako se pri ponovnem pristajanju kvadrokopterja polnjenje ne bo začelo nenadzorovano. Na ta način lahko tudi razberemo, ali je bilo polnjenje uspešno ali je prišlo do napake in po potrebi ustrezno ukrepamo.

3.3.1 Mikrokrmilnik

Arduino je odprtokodno elektronsko prototipno vezje [10]. Izdelana je bila z mislijo o enostavni uporabi in trpežnosti. Programska oprema je brezplačna in zelo enostavna za uporabo. Ponuja veliko knjižnic in primerov [11], ki jih enostavno vključimo v kodo. Primeri so sestavljeni tako, da podrobno nakažejo zmogljivost in način uporabe. Trenutno je programska oprema podprta na operacijskih sistemih Windows, Mac OS X in distribucijah Linuxa [11]. Programiranje poteka v okrnjenem c podobnem programskem jeziku s pridihom zbirnega jezika [13]. Vsi modeli in razširitve [14] so kvalitetno izdelane in enostavni za uporabo.

Izmed vseh modelov, ki so trenutno na razpolago, je za potrebe diplome osnovni model, imenovan Arduino Uno Rev3 [15], dovolj zmogljiv. Uporablja



Slika 3.8: Pogled na Arduino uno z zgornje in spodnje strani.

mikrokrmilnik ATmega328 in deluje pri 16 MHz. Razpolaga s 14 nožicami, ki lahko služijo kot vhod ali izhod, od teh lahko 6 nožic uporabimo za pulzno-širinsko modulacijo [48] (ang. pulse-width modulation PWM). Vsaka od teh nožic zmore na izhod dati do 40mA. Ima tudi 6 analognih vhodov, USB konektor za povezavo z računalnikom ter priključek za zunanje napajanje. Za program je na razpolago 32KB spomina, kar je veliko več potrebujemo. Vezje za delovanje uporablja 5V enosmerne napetosti in se lahko napaja iz dveh virov. Prvi vir je preko USB priključka, ki je lahko priklopljen v računalnik ali polnilnik z USB izhodom. Naslednji vir napajanja priklopimo v vgrajeni priključek za napajalnik. Priporočena vhodna napetost znaša od 7V do 12V, lahko pa uporabimo tudi napetosti vse do 20V, pri čemer pa moramo biti previdni na pregrevanje regulatorja napetosti.

Programiranje mikrokrmilnika je zelo enostavno in ne zahteva nobenega dodatnega znanja. Skico, kot se imenuje programska koda, prenesemo na mikrokrmilnik tako, da le tega preko USB priklopimo na računalnik, kliknemo gumn za nalaganje, na sliki 3.9 je to gumb s puščico usmerjeno v desno, počakamo da se prenos opravi in že lahko krmilimo želene naprave.

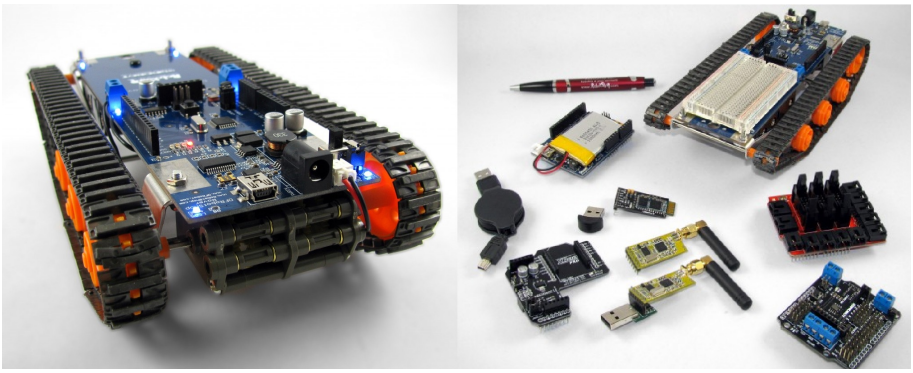
Na ta način lahko programiramo tako ploščice Arduino kot tudi velik nabor ostalih mikrokrmilnikov podjetja ATMEL. Programska oprema namreč skico prevede v strojni jezik po principu, ki velja za mikrokrmilnike ATMEL.

Uporabnost ploščice Arduino pa se tu še ne konča. Na ploščico lahko

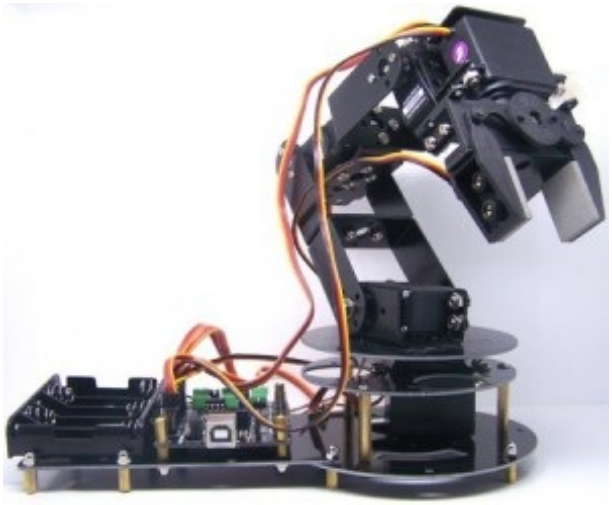
Slika 3.9: Programska oprema za programiranje ploščice Arduino, s skico uporabljeno v diplomski nalogi.

priklopimo veliko naprav. Podatke lahko izpisujemo na eno ali dvovrstični šestnajst segmentni ekran, ki ga priklopimo na nožice mikrokrmilnike. S pomočjo pulzno-širinsko modulacije lahko krmilimo do 6 elektromotorjev ali do 6 servomotorjev. Primer krmiljenja štirih elektromotorjev prikazuje slika 3.10, krmiljenje servomotorjev pa prikazuje slika 3.12. Lahko mu dodamo tudi ultrazvočni merilnik razdalje ali kak drug merilnik razdalje in veliko več. Zelo priljubljen je v robotiki, slika 3.11, saj lahko sestavimo zmogljiv robot, z velikim naborom senzorjev. Izmerjene podatke lahko preko USB vmesnika pošiljamo na računalnik, ali pa ukaze pošiljamo iz računalnika in tako krmilimo želeno napravo (robota, enostaven CNC stroj ali tiskalnik in podobno).

Slika 3.10: Doma izdelani kvadrokopter. Slika povzeta po [16].



Slika 3.11: Primer izdelave robota, ki temelji na Arduino. Slika povzeta po [17].



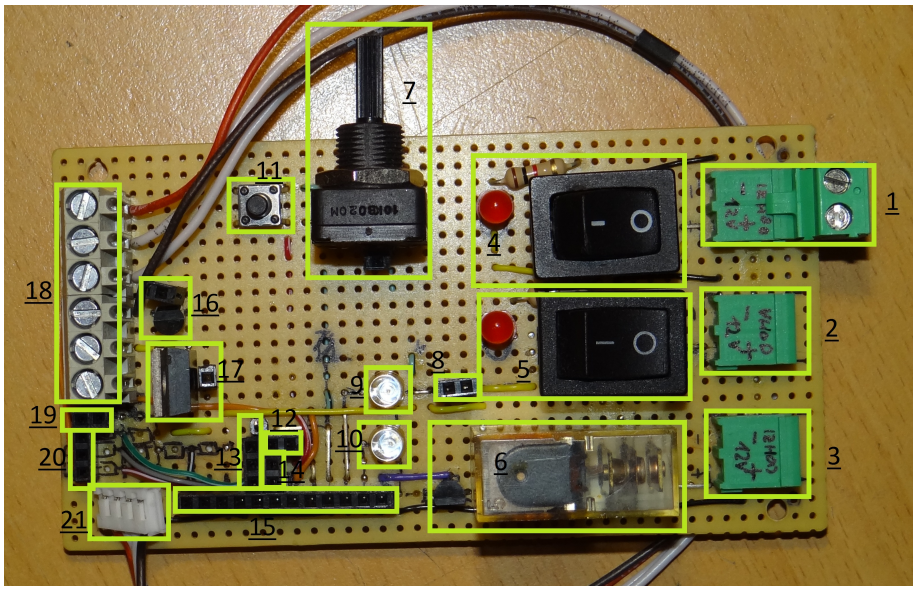
Slika 3.12: Primer uporabe Arduina, ki krmili robotsko roko. Slika povzeta po [18].

3.3.2 Elektronska vezja

Za potrebe uresničitev vseh zahtev, da smo lahko realizirali zastavljen cilj, smo morali izdelati še nekaj dodatnih elektronskih vezij. Njihova naloga je preprosta, predvsem morajo vsebovati ustrezne komponente, ki so dovolj zmogljive za opravljanje določene funkcije. Za izdelavo vezij smo uporabili prototipno ploščico [19]. Za tak pristop smo se odločili, ker smo v teku izdelave diplomske naloge sproti dodajali različne komponente, kakšno tudi nadomestili z drugo komponento ali v celoti odstranili. Prednost prototipne ploščice je ravno v tem, da lahko uporabimo različne komponente, ki jih enostavno vstavimo v že izvrtane luknje, komponente pa nato povežemo med seboj.

Slika 3.13 prikazuje glavno vezje s komponentami. Vezje se priklopi na vir napetosti, 12V akumulator, ter služi kot dodatek mikrokrmilniku. S pomočjo vezja in ustreznih komponent lahko krmilimo porabnike z višjo napetostjo in z večjimi tokovi. Preko uporovnega vmesnika lahko tudi direktno priključimo višje napetosti na vhodne nožice mikrokrmilnika, ne da bi ga

pri tem poškodovali. Ta pristop smo uporabili za namene merjenja napetosti akumulatorja, LiPo baterije in stanja obeh svetlečih diod na polnilniku. Shematski načrt vezja se nahaja v dodatku A.2.



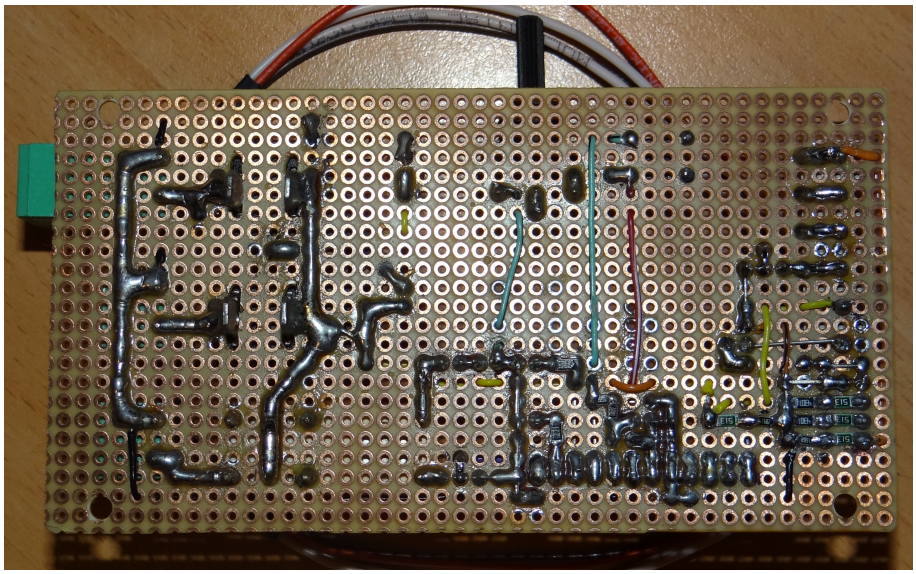
Slika 3.13: Elektronsko vezje kot vmesnik za mikrokrmilnik. Prikazana je zgornja stran. Opisi posameznih komponent se nahajajo v tabeli 3.1.

Na sliki 3.14 lahko vidimo spodnjo stran vezja. Poleg povezav med komponentami se tu nahajajo, razen dveh, vsi upori. SMD upore smo uporabili zaradi dveh razlogov. Prvi razlog je, da ni potrebe po tokovno zmogljivih uporih, saj so tokovi zelo majhni, od $0,2mA$ do $20mA$ in ni velikih toplotnih izgub. Drugi razlog je velikost SMD uporov. Ti so od klasičnih uporov veliko manjši, končno vezje pa je manjše.

V času polnjenja je lahko LiPo baterija priklopljena le na polnilec. To pomeni, da na baterijo ne sme biti priklopljen noben porabnik, v našem primeru je to kvadrokopter. Torej potrebujemo vezje, s katerim bomo lahko baterijo v času polnjenja odklopili. Pri izbiri komponent moramo upoštevati sorazmerno visok tok, ki ga kvadrokopter porablja med delovanjem. Sodeč po sliki 3.15, je nazivna poraba kvadrokopterja 8A pri 12V. V ta namen

Številka komponente	Opis
1	Priključek za dodatne porabnike, vklopi se ga s stikalom 4.
2	Priključek za vir napajanja.
3	Priključek za priklop polnilnika baterije.
4	Stikalo za vklop napajalne napetosti na izhodnem priključku 1. Ob vklopu sveti rdeča LED dioda.
5	Stikalo za vklop vezja. Ob vklopu sveti rdeča LED dioda.
6	NPN tranzistor za vklop releja, ki vklopi LiPo polnilnik na priključku 3.
7	Potenciometer za nastavljanje jakosti osvetlitve. Izhod je povezan na Arduino uno vhod A4.
8	12V priključek za napajanje Arduino uno. Na levi strani je (-), desno (+).
9	LED svetilka, ki signalizira stanje LiPo baterije.
10	LED svetilka, ki signalizira stanje akumulatorja.
11	Tipka, s katero ročno aktiviramo polnjenje LiPo baterije.
12	Priključek na katerega priklopimo tipko, ki preverja prisotnost AR.Drona na ploščadi.
13	GND, uporabno za lažji priklop merilnih enot.
14	5V
15	Priključki za povezavo vezja z Arduino uno.
16	NPN tranzistor za mehki odklop baterije in vezja. Vhod je priklopljen na Arduino uno nožico 7 in je v nizkem stanju le v času poljenje baterije. Priključki od leve proti desni (kam se priklopi na Arduino uno): A3, A2, A1, A0, 5V, GND, 8, 9, 10, 11, 12, 13.
17	MOS tranzistor za reguliranje svetilnosti osvetlitve. Vhod je priklopljen na Arduino uno nožico 6.
18	Priključna mesta za priklop kablov, ki povezujejo pristajalno ploščad z vezjem. Od zgoraj navzdol si priključki sledijo: Od 1 (+) do 4 (-) priklop 3S baterije ter 5 (+) in 6 (–) napajanja osvetlitve.
19	Priključek za mehki odklop baterije v AR.Dronu. Desena stran priključka je GND, leva stran je priklopljena na Arduino uno nožico 8.
20	Priključki za merjenje napetosti. Od zgoraj navzdol: vhod napetosti zelene diode polnilnika, vhod napetosti rdeče diode polnilnika, vhod napetosti LiPo baterije.
21	Priključek za priklop na izhod polnilnika LiPo baterije.

Tabela 3.1: Opsi posameznih delov vezja s slike 3.13.



Slika 3.14: Elektronsko vezje s spodnje strani.

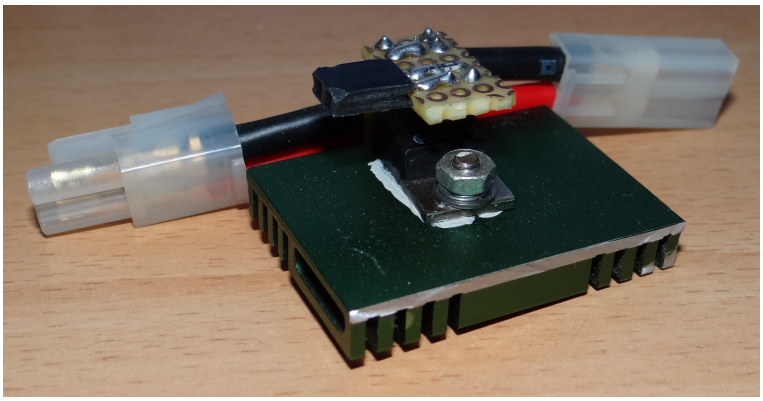
smo uporabili močnostni MOS tranzistor, BUZ11 , ki zmore tokove do 30A in ima notranjo upornost do 0.04Ω [20].

Na sliki 3.16 je prikazano vezje za odklop baterije in kvadrokopterja. Zaradi galvanske ločitve LiPo baterije in mikrokrmilnika, ki krmili tranzistor, smo morali dodati še element SFH617A-2 [21]. Shematski načrt vezja se nahaja v prilogi A.2.

V poglavju 1.3 smo omenili, da v diplomski nalogi nismo uporabili možnosti komunikacije med programsko opremo in mikrokrmilnikom. Komunikacijo bi lahko izkoristili, če bi programski opremi dodali funkcionalnost, da bi kvadrokopter dali v stanje pripravljenosti. V tem stanju bi lahko ostal vse dokler ne bi želeli uporabiti kvadrokopterja. Ko pa bi želeli uporabiti kvadrokopter, bi le pritisnili na gumb in program bi poslal ukaz mikrokrmilniku naj vklopi kvadrokopter. Vklop kvadrokopterja bi lahko tako izvedli tudi na daljavo ali v primeru popolne avtonomnosti kvadrokopterja, bi se vklop izvedel samodejno, ko bi dobil novo nalogo.

Notranja upornost MOS tranzistorja je majhna a kljub temu je ne moremo zanemariti. Zaradi notranje upornosti tranzistorja in visokih tokov

Slika 3.15: Nazivna napetost in tok delovanja.



Slika 3.16: Vmesnik za odklop LiPo baterije od kvadrokopterja.

nastane na tranzistorju padec napetosti. Vsa izguba energije se pretvori v odvečno toploto, ki jo moramo odvajati, da trajno ne poškodujemo tranzistor.

$$I = \frac{U}{R} \tag{3.1}$$

$$R_D = \frac{U}{I} = \frac{12V}{8A} = 1,5\Omega \tag{3.2}$$

Enačbo za izračun notranje upornosti kvadrokopterja (3.2) izpeljemo iz enačbe (3.1). Pri izračunu upoštevamo nazivni tok in napetost, ki sta na-

vedena na kvadrokoptorju, kar prikazuje slika 3.15. Izračunana nadomestna upornost kvadrokopterja, R_D, kjer indeks D predstavlja AR.Drone, predstavlja najnižjo možno notranjo upornost.

Upornost tranzistorja preberemo iz specifikacije [20]. Upoštevamo zaporedno vezavo obeh nadomestnih uporov in izračunamo nov tok, I_N, kjer indeks N predstavlja nov tok, ki teče skozi oba porabnika, kar predstavlja enačba (3.3).

$$I_N = \frac{U}{R_D + R_M} = \frac{12V}{1,54\Omega} = 7,79A \tag{3.3}$$

$$U_M = I_N \times R_M = 0,31V \tag{3.4}$$

$$P_M = I_N \times U_M = 2,42W \tag{3.5}$$

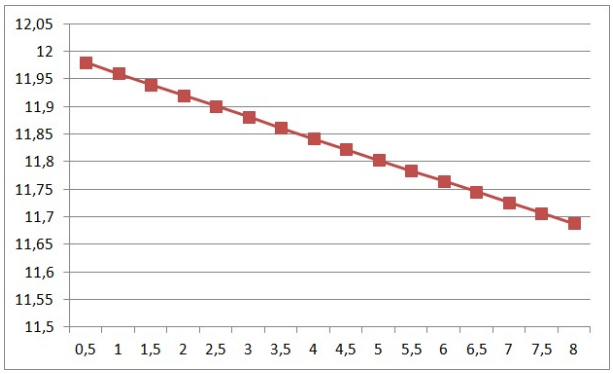
Enačba (3.4) prikazuje padec napetosti, U_M (indeks M predstavlja MOSFET tranzistor), ki nastane na tranzistorju. Padec napetosti je majhen a če upoštevamo tudi nizek vir napetosti, padec ni več zanemarljiv. Nastala izguba energije, ki se pretvori v odvečno toploto in jo moramo odvajati, prikazuje enačba (3.5).

Slika 3.17 prikazuje graf, ki ponazarja vpliv padca napetosti na tranzistorju pri konstantni vhodni napetosti $12V$ in naraščanju toka vse do $8A$. Ker ima kvadrokopter merilnik kapacitete baterije, padec napetosti na tranzistorju povzroči, da kvadrokopter izmeri nižjo napetost baterije kot je ta v resnici.

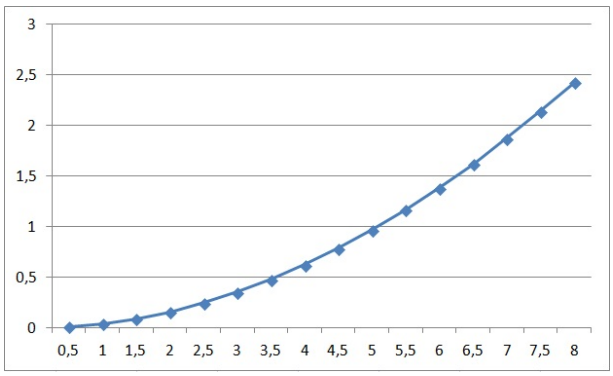
Slika 3.18 prikazuje, kako naraščajo izgube na tranzistorju. Tudi tu smo upoštevali konstantno napajalno napetost ter tokovno porabo vse do $8A$. Izračunani rezultati predstavljajo zgornjo mejo, saj v resnici vhodna napetost, oziroma napetost baterije, ni konstantna in skozi čas pada.

Slika 3.19 prikazuje vezje, za odklop baterije, priklopljeno na baterijo in kvadrokopter. Povezana je tudi baterija s polnilnim priključkom, dodaten priklop na vezju služi za signaliziranje tranzistorju, ki nato prekine tokokrog.

Izgube na tranzistorju privedejo do krajše avtonomije baterije. Lahko bi izbrali tranzistor z nižjo notranjo upornostjo in tako zmanjšali izgube.



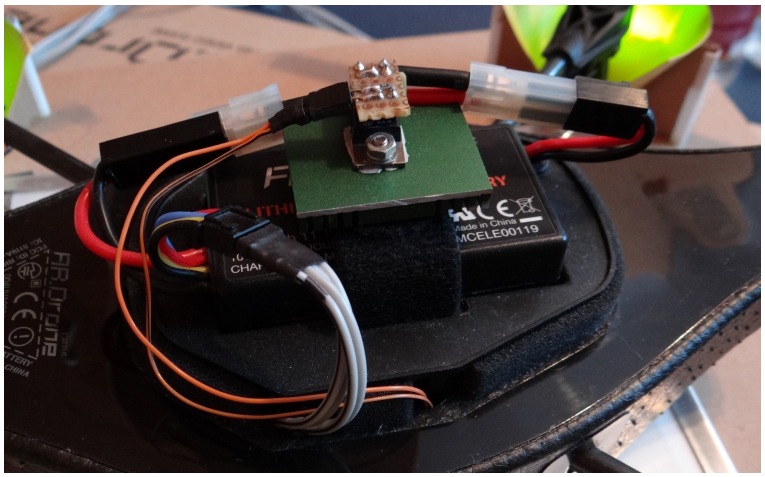
Slika 3.17: Vpliv na tranzistorju nastalega padca napetosti na napetost, ki jo zazna kvadrokopter.



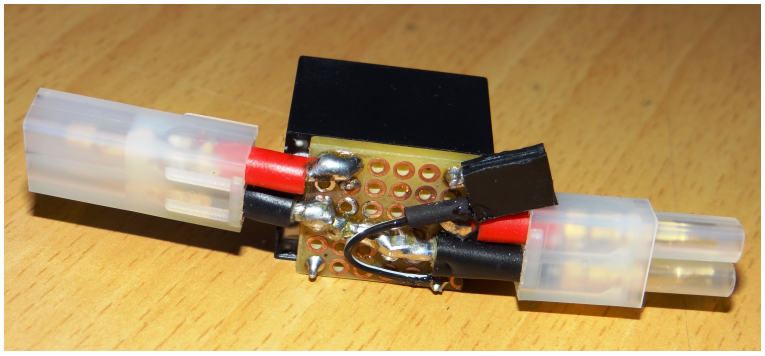
Slika 3.18: Energija, ki se na tranzistorju pretvori v toploto.

Izgube popolnoma odstranimo z uporabo releja. Ker rele deluje kot mehansko stikalo, na njem izgub praktično ni. Pozorni moramo biti le, da izberemo tokovno dovolj zmogljiv rele. V primeru izbire tokovno ne dovolj zmogljivega releja, bi le ta začel delovati kot upor z nezanemarljivo upornostjo in ga lahko tudi trajno poškodujemo. Primer nadomestnega releja prikazuje slika 3.20.

Za kontakt s polnilnimi in kontrolnimi priključki na pristajalni ploščadi smo uporabili ravno ploščico prevlečeno z bakrom. Ker baker na zraku skozi čas oksidira, smo ploščico prevlekli s cinom za spajkanje. Na sliki 3.21 je kontaktna ploščica s selotejpom pritrjena na ogrodje kvadrokopterja, na

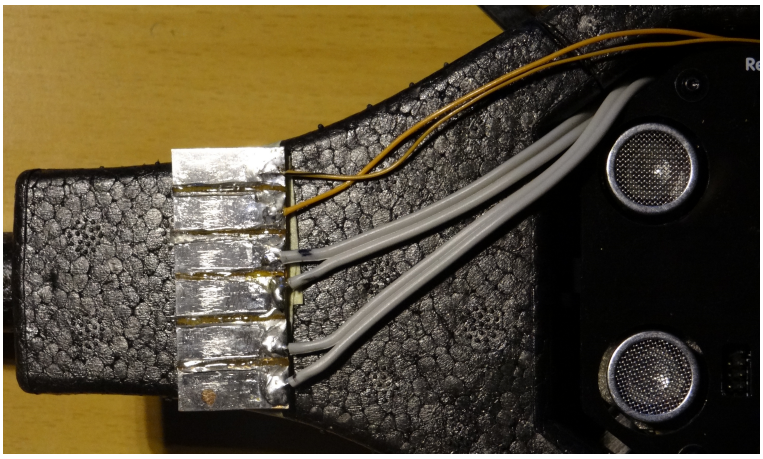


Slika 3.19: Vmesnik za odklop baterije v času aktivnosti.



Slika 3.20: Rele z vsemi potrebnimi priključki.

ploščico pa so prispajkane žice. Od spodaj navzgor si sledijo + kontakt za celico 3, nato + kontakt za celico 2, + kontakt za celico 1, nato – baterije ter še + in – kontakta za krmiljenje odklopa baterije.



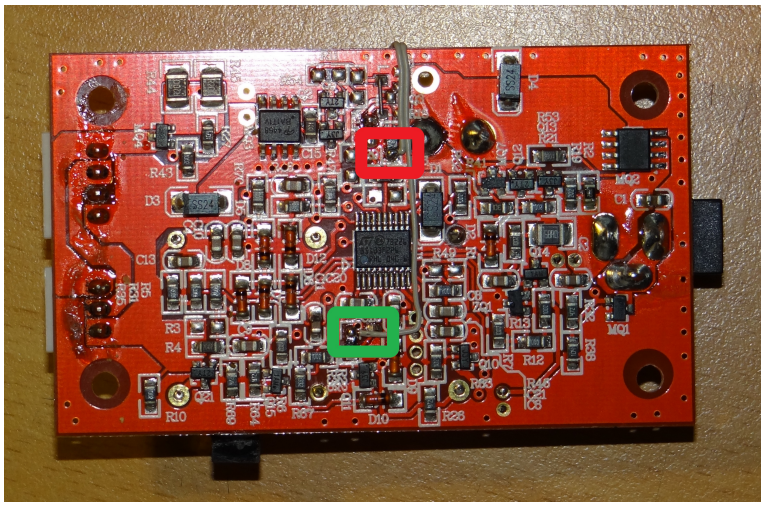
Slika 3.21: Kontaktna ploščica, povezana z LiPo baterijo in vezjem za odklop baterije.

3.3.3 Polnilnik

Brez ustreznega polnilnika polnjenje LiPo baterije ni priporočljivo. Kot prvo mora omogočati polnjenje LiPo baterije s $3S$ priključkom. Upoštevati moramo tudi vir napajanja, kar je v našem primeru $12V$ akumulator. Akumulator z nazivno napetostjo $12V$ ima pri neobremenjenem izhodu v času največje napolnjenosti približno $12,8V$ [22] , med obratovanjem pa lahko napetost na obremenjenem izhodu pade tudi pod $11V$. Originalni polnilnik, priložen AR.Dronu, za obratovanje potrebuje napajanje v razponu od $13V$ do $18V$, kar pa ne ustreza našim zahtevam. V ta namen je najbolje uporabiti polnilnike ki so namenjeni za terensko uporabo in jih lahko direktno priključimo na svinčev akumulator. Izbrali smo modelarski polnilnik podjetja Graupner s sledečo specifikacijo. Obratovalna napetost se nahaja v razponu od $10V$ do $18V$ enosmerne napetosti ter poleg tri celičnih baterij ($3S$) omogoča polnjenje tudi dvocelične baterije ($2S$), česar v našem primeru ne bomo potrebovali. Podrobnejše informacije o polnilniku lahko preberemo na spletni strani [23].

Polnilnik smo malo predelali. Da krmilna lahko logika ve, kdaj je polnjenje zaključeno, mora biti povezana s polnilnikom. Polnilnik stanje aktivnosti

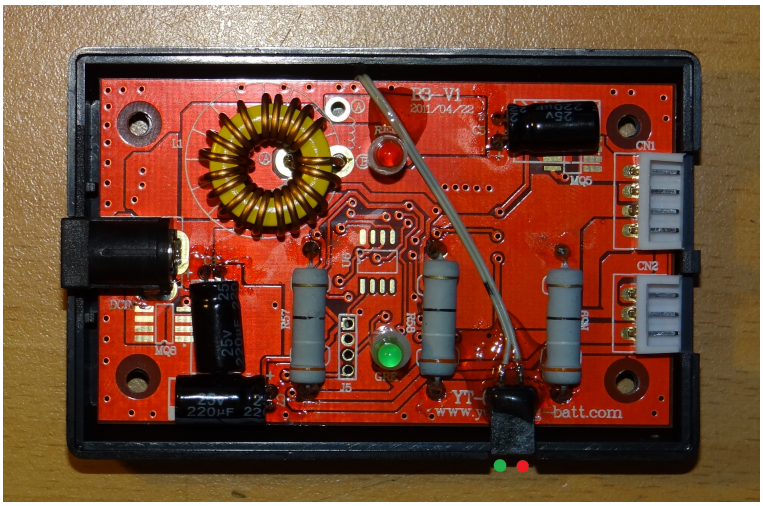
signalizira preko dveh svetlečih diod (ang. LED), zelene in rdeče. V času polnjenja sveti rdeča dioda, zelena pa ne sveti. Ko je polnjenje zaključeno sveti zelena dioda ter rdeča ne. Da lahko krmilna logika prepozna stanje obeh diod, smo dodali priključek, preko katerega spremljamo napetost na posamezni svetleči diodi. Poseg v polnilnik prikazujeta sliki 3.22 in 3.23.



Slika 3.22: Zelen okvir označuje mesto prispajkane žice za zeleno diodo, rdeč okvir pa za rdečo diodo.

Slika 3.22 prikazuje mesto kamor smo prispajkali žici. Obe žici sta prispajkani pred zaščitnim uporom svetleče diode, zaradi česar moramo biti pri nadaljnjem rokovanju previdni. Nobena od žic ne sme priti v stik z ozemljitvijo ali napetostjo višje od $5V$. V primeru da pride do kratkega stika, bomo čip poškodovali do meje, kjer ga lahko samo še zavržemo.

Slika 3.23 prikazuje pritrditev obeh žic in konektorja. Da lahko do konektorja dostopamo tudi ko je ohišje zaprto, smo v ohišje narediti ustrezno veliko odprtino. Barvni piki ob konektorju ponazarjata barvo diod.

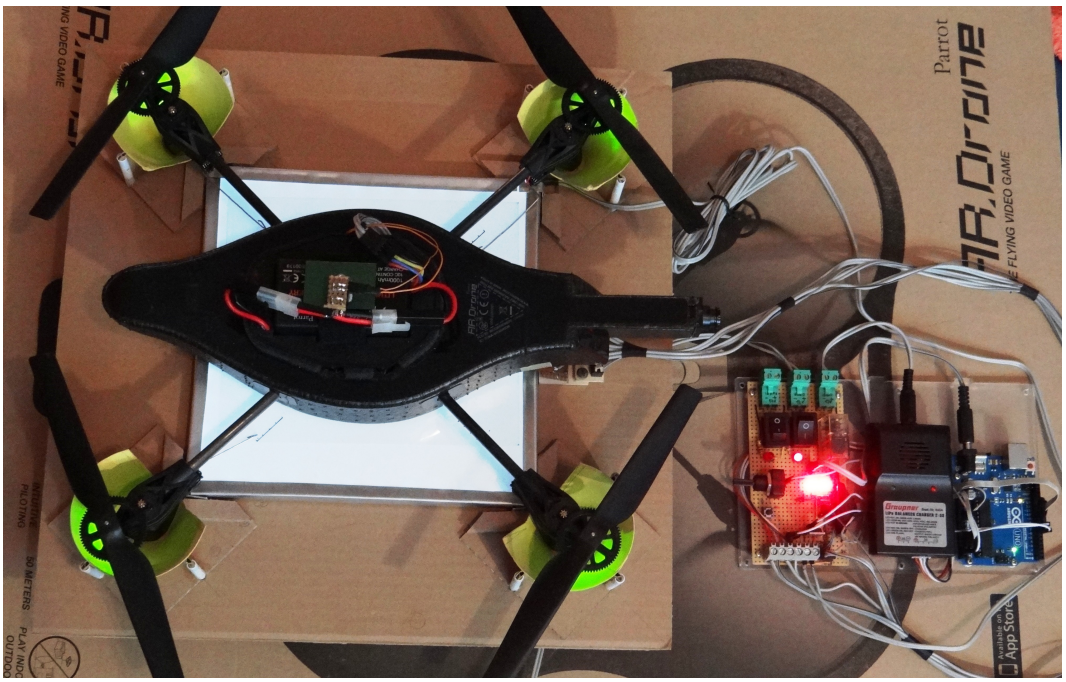


Slika 3.23: Pritrditev žici in konektorja.

3.4 Končni izdelek strojne opreme

Končni produkt strojne opreme je sestavljen modularno. Vsak modul opravlja točno določeno nalogo in je lahko pri tem bolj ali manj uspešen. Modularna zgradba nam omogoča hitro in enostavno popravilo, zamenjavo ali nadgradnje posameznih modulov. Moduli, ki so izpostavljeni fizičnim kontaktom se lahko poškodujejo ali celo uničijo. Največjemu fizičnemu stresu so izpostavljeni pristajalni stožci. Ti so izdelani iz kartona in podprti le s treh strani. Med pristajanjem se lahko pripeti, da nogica kvadrokopterja pristane na robu stožca in tam tudi ostane, na mestu kontakta pa se karton deformira.

Produkt opravlja zastavljeno nalogo. Torej, omogoča pristajanje kvadrokopterja, stožci med polnjenjem ali neuporabo kvadrokpter zadržujejo na mestu in tako preprečujejo zdrs s ploščadi. Slednje se lahko pripeti zaradi ne ravnega terena, po katerem se premika avtonomno mobilno vozilo. V slabši vidljivosti k boljši prepoznavnosti na pomoč priskoči dodatna osvetlitev, katere jakost lahko nastavljamo. Ravno tako uspešno upravljata svojo nalogo mikrokrmilnik in polnilnik baterije. Končni izdelek prikazuje slika 3.24.



Slika 3.24: Strojna oprema z aktivirano osvetlitvijo, kvadrokopterjem in ponazarjanjem stanj LiPo baterije in vira napetosti.

Poglavje 4

Programska oprema

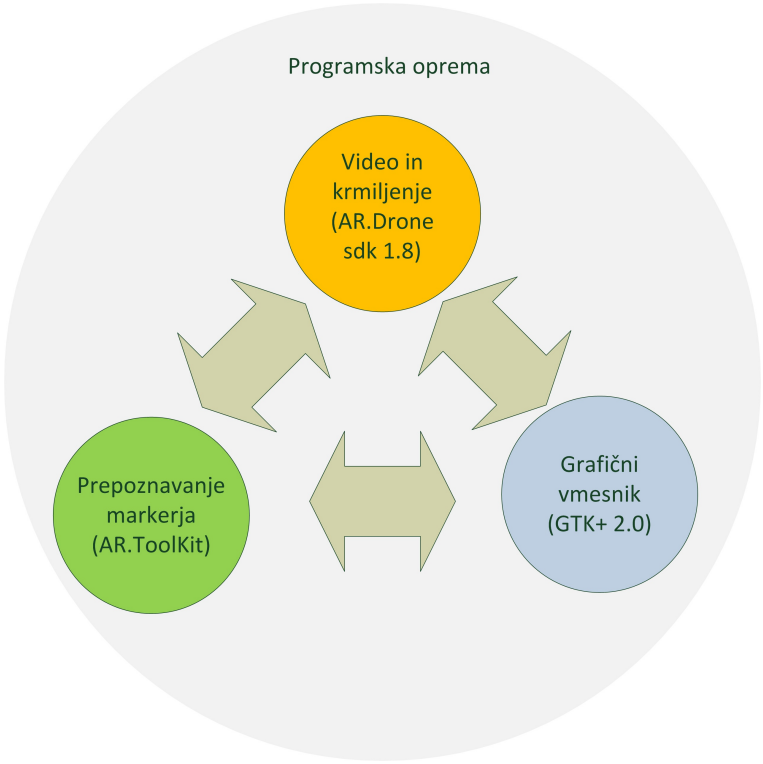
V sledečem poglavju se dotaknemo uporabljene programske opreme. Podrobneje spoznamo ARToolKit, osnovni princip, njegovo uporabnost in omejitve pri uporabi. Sledi vključitev ARToolkita v AR.Drone SDK, kako s pomočjo dobljenih podatkov krmilimo kvadrokopter ter predstavitev grafičnega vmesnika.

4.1 Razdelitev programske opreme

Ravno tako kot sklop strojna oprema, se tudi programska oprema deli na podsklope. Značilnost teh podsklopov je, ravno tako kot pri podsklopih strojne opreme, njihova samostojnost podsklopov. Ta samostojnost je mišljena tako, da lahko posamezen podsklop nadomestimo z novejšim, na primer, lahko zamenjamo AR.Drone SDK 1.8 z novejšo verzijo 2.0 in lahko uporabimo obstoječo kodo, brez potrebe po drastičnem posegu v kodo.

Sklop programska oprema se deli na tri podsklope, kar prikazuje slika 4.1. Prvi podsklop je naslovljen kot Video in krmiljenje. Sklop pravzaprav predstavlja AR.Drone SDK 1.8. SDK poskrbi za nastavitev osnovnih parametrov in komunikacijo med kvadrokopterjem in računalnikom. Na tem sloni naša celotna aplikacija. Drugi podsklop predstavlja prepoznavanje oznake. V tem podsklopu poskrbimo za prepoznavanje oznake na pristajalni ploščadi, z do-

bljenimi podatki pa nato ustrezno krmilimo kvadrokopter. Za prepoznavanje oznake smo se odločili da uporabimo ARToolkit [30], z implementiranim algoritmom za določanje lokacije oznake. Tretji in zadnji podsklop predstavlja grafični vmesnik. Predstavlja vez med uporabnikom in prvim ter drugim podsklopom.



Slika 4.1: Prikaz delitve sklopa programska oprema na posamezne podsklope.

4.2 AR.Drone SDK

Podjetje Parrot je javnosti poleg aplikacij za mobilne naprave predstavilo tudi razvojno okolje, ki ga lahko uporabimo na različnih platformah. Trenutno so podprti operacijski sistemi iOS, Android, Windows in Linux. Za naše potrebe smo uporabili SDK 1.8 [25], na razpolago pa ju tudi že SDK 2.0. SDK 1.8 je bil izbran zaradi številčne uporabe in velike pomoči na forumih in

raznih spletnih straneh, kot na primer [27], [28] in [29] ,na katrih najdemo številne nasvete in pomoč. Razvojno okolje in vse dodatne knjižnice smo namestili na Ubuntu 10.04.

Vsa koda in postopki ki bodo prikazani v nadaljevanju, veljajo za operacijski sistem Ubuntu in ne moremo trditi, da bo vse v enaki meri delovalo tudi na preostalih distribucijah Linux.

Ob namestitvi razvojnega okolja, le ta že vsebuje osnovno ogrodje, ki poskrbi za ustrezno povezavo s kvadrokopterjem in računalnikom ter inicializacijo potrebnih parametrov. Ravno tako že vsebuje vse potrebno za krmiljenje kvadrokopterja, kot so na primer vzlet, pristajanje, zasilni izklop, krmiljenje in podobno a le, če si lastimo enega od dveh podprtih igralnih ploščkov. V primeru, da si ne lastimo podprtega igralnega ploščka, si moramo vzeti čas in ustrezno spremeniti kodo. Kratka navodila za uspešno prilagoditev kode, za uporabo poljubnega igralnega ploščka, se nahaja v poglavju A.3. Poleg krmiljenja je realizirano tudi dekodiranje videa, ki ga neprestano pošilja kvadrokopter, a le ta še ni prikazan. Razlog za to je v tem, da osnovno ogrodje služi le kot podlaga, na kateri lahko vsakdo razvija aplikacije po svoji želji.

AR.Drone SKD 1.8 si prenesemo s spletne strani [25] na svoj računalnik. Na isti spletni strani najdemo tudi vodnik za razvijalce. V vodniku je bolj obsežno predstavljeno delovanje razvojnega okolja, kako se dekodira prejeti video, podrobno so razložene funkcije za krmiljenje in nastavljanje ostalih parametrov in podobno. Za namestitev razvojnega okolja sledimo navodilom [26], sama namestitev in uporaba pa v Ubuntu 10.04 poteka brez zapletov. Za prenos razvojnega okolja in vodnika za razvijalce je potrebna predhodna registracija.

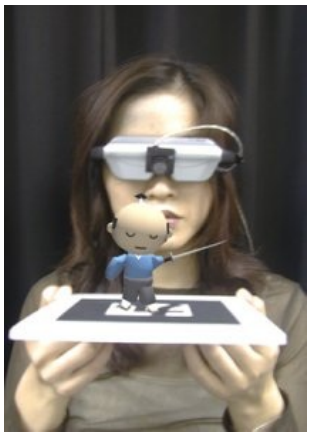
4.3 Ocenjevanje lege pristajalne ploščadi

Pri pristajanju kvadrokopterja na pristajalni ploščadi si pomagamo z vizualnim zaznavanjem. Za vizualno zaznavanje smo izbrali orodje ARToolKit [30]. Zanj smo se odločili zaradi enostavne uporabe, velike hitrosti algoritma

za zaznavanje in velike natančnosti. V poglavju 4.3.1 predstavimo ARToolKit, kako je zgrajen, v katere namene ga lahko uporabimo in kako deluje. V poglavju 4.3.2 podrobneje opišemo delovanje algoritma za zaznavanje. Na koncu, v poglavju 4.3.3, je opisan postopek kalibriranja kamere.

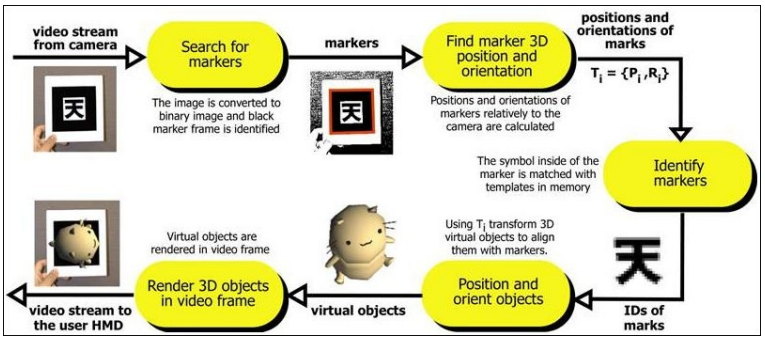
4.3.1 ARToolKit

ARToolKit [30] knjižnica temelji na C in C++ jeziku in omogoča programerju na enostaven način razvijati aplikacije o obogateno realnost (ang. Augmented Reality ali AR). Obogatena resničnost deluje na način, da s pomočjo računalniške grafike na zajeto sliko izriše nov lik, kar ponazarja slika 4.2. Princip ima velik potencial tako v industriji kot tudi v akademskih krogih.



Slika 4.2: Primer uporabe obogatene realnosti. Slika povzeta po [30].

Poenostavljen primer delovanja prikazuje slika 4.3. Z videokamero zajeto sliko podamo v prvo stopnjo, kjer se na sliki označijo vsi pravokotniki. Sledi izračun lokacije in orientacije pravokotnika (oznake) ter nato še identifikacija vzorca. V primeru da je vzorec pravi, se nad njim s pomočjo računalniške grafike izriše poljuben lik. Najtežji del pri obogateni resničnosti predstavlja izračun lokacije oznake. Zaradi neidealnih pogojev in uporabo različnih kamer lahko pride do velikega odstopanja pri izračunih.



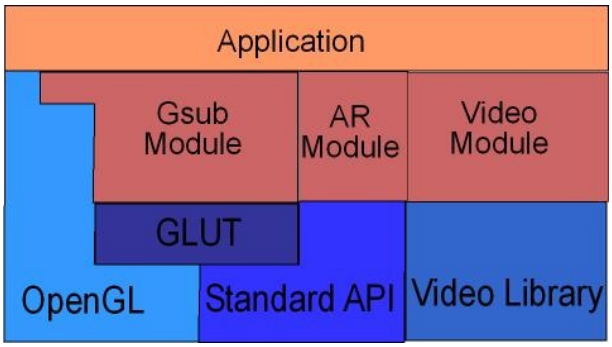
Slika 4.3: Proces, ki poteka za izris lika na oznaki. Slika povzeta po [31].

Proces se izvede za vsako sliko, ki jo posname kamera. Za potrebe diplomske naloge pa ne potrebujemo celotnega procesa. Za krmiljenje kvadrokopterja potrebujemo le podatke, kje se oznaka nahaja, zato lahko izris lika izpustimo.

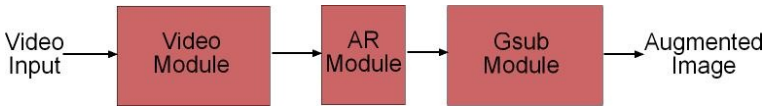
Zgradba ARToolkita je modularna, kar poenostavi njegovo uporabo. Shemo modularne zgradbe predstavljajo slike 4.4, 4.5 in 4.6. Zaradi modularne zgradbe lahko uporabimo, zamenjamo ali spremenimo vsak modul posebej. AR.Drone SDK že dekodira video, ki ga pošilja kvadrokopter, v RGB format. Ravno ta format ARToolkit potrebuje, da lahko izvaja iskanje oznake, kar lahko razberemo iz slike 4.6. Torej »Video Module« ne potrebujemo. Ravno tako ne potrebujemo »Gsub Module« za izris. Potrebujemo le »AR Module«, ki poskrbi za iskanje oznake in vrne relativno pozicijo oznake glede na kamero.

Slika 4.6 nazorno prikazuje obliko videa zapisa od zajemanja, obdelave in izrisa. Kvadrokopter pošilja video, ki ga z vgrajeno funkcijo v AR.Drone SDK samodejno pretvori v RGB. RBG sliko nato podamo v funkciji za sledenje, na sliki 4.6 to ponazarja »TRACKING«. Funkcija vrne matriko 3×4 kjer je podana relativna lokacija oznake glede na kamero kvadrokopterja.

Za nekomercialne namene prenesemo ARToolkit 2.72.1 s spletne strani [33].



Slika 4.4: Modularna zgradba ARToolkita. Slika povzeta po [32].

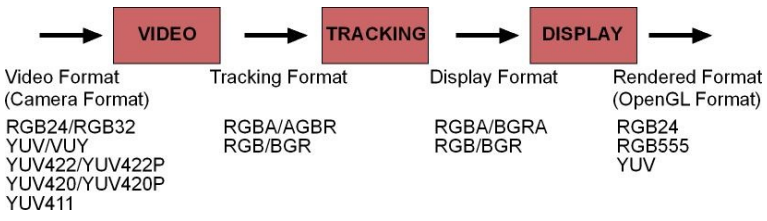


Slika 4.5: Poenostavljena zgradba ARToolkita. Slika povzeta po [32].

4.3.2 Delovanje algoritma za prepoznavanje oznake

Pri določanju lokacije oznake največjo oviro predstavlja relacija med oznako in kamero. Slika 4.7 prikazuje neznanko v procesu določanja lokacije oznake.

Osnovo za določanje lokacije predstavlja kvadrat, oznaka, katerega velikost je vnaprej podana. Koordinatni sistem oznake poznamo, ravno tako tudi koordinatni sistem kamere, ne poznamo pa transformacijske matrike med obema sistemoma (T_{cm}), kar predstavljata enačbi (4.1) in (4.2).



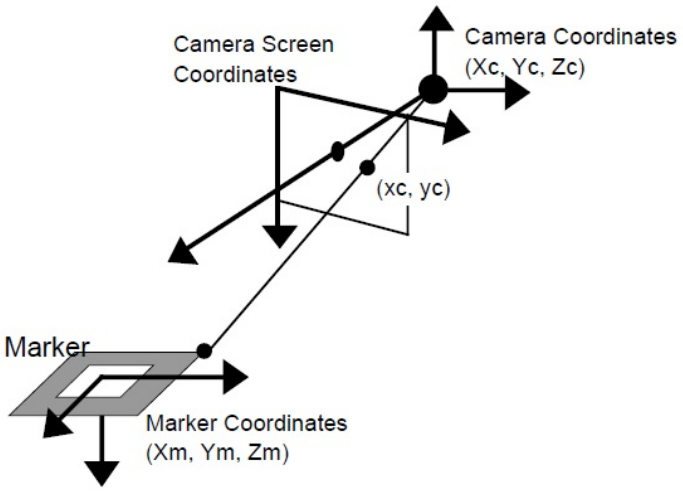
Slika 4.6: Video formati na posamezni stopnji. Slika povzeta po [32].

Slika 4.7: V procesu določanja lokacije oznake v kordinatnem sistemu kamere se išče transformacijska matrika T_{cm}. Slika povzeta po [35].

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{4.1}$$

$$= \begin{bmatrix} & V_{3\times3} & & W_{3\times1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_{cm} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{4.2}$$

Na zajeti sliki se nato poišče kvadrat. To se stori tako se z ustreznim pragom (ang. threshold) na sliki opravimo segmentacijo. Vsaki regiji, kjer se lahko sekajo 4 črte, te dobljene podatke o sekajočih se črtah nato shranimo za kasnejšo obdelavo. Shranjene regije se nato normalizirajo in vzorec v njih primerja z nastavljenim vzorcem. Vzorec lahko predstavlja slika ali besedilo. Za proces normalizacije se uporabi enačba (4.3). Vse spremenljivke v transformacijski matriki so determinirane z določanjem ujemanja, štirih predhodno shranjenih točk, v koordinatnem sistemu oznake in kamere. Po tem se lahko prične normalizacija z uporabo sledeče matrike:



Slika 4.8: Povezanost kordinatnega sistema oznake in kamere. Slika je povzeta po [36].

$$\begin{bmatrix} hx_c \\ hy_c \\ h \end{bmatrix} = \begin{bmatrix} N_{11} & N_{12} & N_{13} \\ N_{21} & N_{22} & N_{23} \\ N_{31} & N_{32} & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ 1 \end{bmatrix} \tag{4.3}$$

Ko sta na sliko projicirani dve vzporedni premici, ti premici v koordinatnem sistemu kamere predstavljata spodnji enačbi:

$$a_1x + b_1x + c_1 = 0, \qquad a_2x + b_2x + c_2 = 0 \tag{4.4}$$

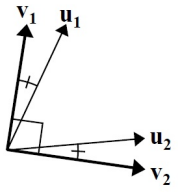
S podano perspektivno projekcijsko matriko P, dobljeno s kalibracijo kamere po enačbi (4.5), lahko enačbo (4.4) zapišemo kot (4.6) in (4.7) , kjer x in y zamenjamo z X_c in Y_c.

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} hx_c \\ hy_c \\ h \\ 1 \end{bmatrix} = P \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{4.5}$$

$$a_1P_{11}X_c + (a_1P_{12} + b_1P_{22})Y_c + (a_1P_{13} + b_1P_{23} + c_2)Z_c = 0 \quad (4.6)$$

$$a_2P_{11}X_c + (a_2P_{12} + b_2P_{22})Y_c + (a_2P_{13} + b_2P_{23} + c_2)Z_c = 0 \quad (4.7)$$

Glede na vektorja ravnine n_1 in n_2, je smer vektorja obeh vzporednih strani kvadrata podana s produktom $n_1 \times n_2$. Dobimo vektorja u_1 in u_2, ki naj bi bila pravokotna, kar pa zaradi napake v procesiranju slike nista. Sledeče popravimo tako, da v isti ravnini določimo pravokotna si vektorja v_1 in v_2, ki napako zgladita, kot to prikazuje slika 4.9. Vektor, ki je tako pravokoten tako na v_1 in v_2, je v_3. Vektor v_3 je pravokoten na ravnino oznake, iz česar sledi, da je rotacijska matrika $V_{3\times3}$, v transformacijski matriki T_{cm} (enačba (4.2)), enaka $[V_1^t, V_2^t, V_3^t]$. Sedaj ko imamo rotacijsko komponento $V_{3\times3}$, po enačbi (4.1) in (4.2) izračunamo še vrednosti za W_x, W_y in W_z.



Slika 4.9: Primer popravila vektorjev. Slika je povzeta po [36].

Tako dobljena transformacijska matrika ni nujno točna. Napako transformacijske matrike lahko zmanjšamo na sledeč način. Poleg zgoraj opisanega procesa, koordinate štirih točk, ki predstavljajo oglišča oznake, pretvorimo iz koordinatnega sistema oznake v koordinatni sistem kamere s predhodno dobljeno transformacijsko matriko. Transformacijska matrika se nato optimizira glede na razliko obeh transformiranih koordinat. Z večanjem števila iteracij, se točnost povečuje. To velja le v primeru, da se tako kamera kot oznaka ne premikata, ali pa premikata zelo počasi. V nasprotnem primeru, ko ti premiki niso majhni, dobimo bolj zglajene podatke, ki pa niso nujno točni.

4.3.3 Kalibracija kamere

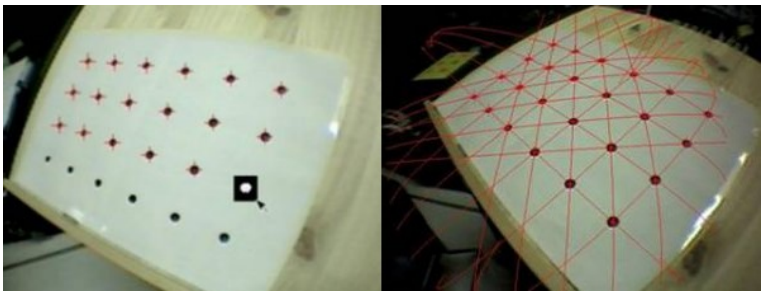
Kot smo že omenili, ARToolkit na podani sliki išče oznako, na podlagi katere nato izračuna relativno lokacijo oznake glede na kamero. Vsaka oznaka je sestavljena iz dveh delov. Prvi del predstavlja pravokotnih in je pri vseh enak. Drugi del predstavlja poljuben lik znotraj pravokotnika. ARToolkit že vsebuje nekaj primerov oznak, ki jih lahko uporabimo ter tudi prazen pravokotnik, v katerega lahko narišemo znak – številka, črka, beseda, slika itd.

Na trgu je velika dostopnost različnih kamer. Poleg razlike v kvaliteti slike, slikovnih točk in hitrosti zajemanja slik, se kamere razlikujejo tudi po lečah. Naloga vsake leče je usmeriti snop svetlobe na svetlobni senzor kamere. Pri tem pa leča popači sliko, kar nazorno prikazuje tudi slika 4.10. Čeprav ARToolKit že ponuja privzete nastavitve za kamere, ki se ujemajo z veliko različnimi kamerami, je za pridobivanje točnejših rezultatov priporočljivo kamero v uporabi kalibrirati. V ta namen imamo na razpolago dva pristopa [41]. Prvi je zahtevnejši a ponuja boljše rezultate. Drugi je nekoliko lažji in hitrejši a ponuja manj natančne rezultate. Za kalibriranje horizontalne kamere kvadrokopterja smo uporabili prvi pristop.

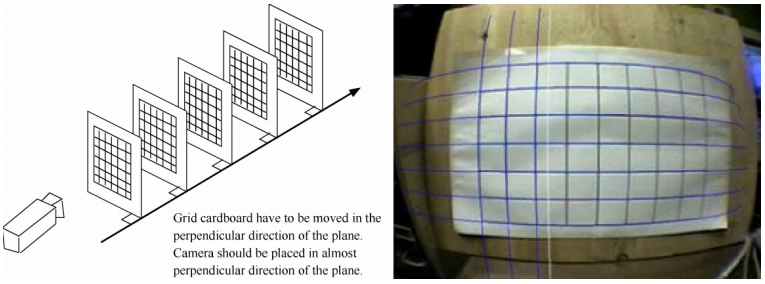
Zahtevnejši pristop kalibriranja je sestavljen iz dveh korakov. V prvem koraku se pridobi parametre popačenje slike. To popačitev slike ustvari zaobljenost leč, ki je z večanjem vidnega kota vse večja. Primer postopka prikazuje slika 4.10, kjer je na levi strani prikazan zajem slike in označevanje točk, na desni strani slike pa je prikazan končni rezultat, kjer rdeče črte nakazujejo ukrivljenost.

Drugi korak se uporabi za iskanje fokusne dolžine in ostalih parametrov. Način kalibriranja v drugem koraku nazorno prikazuje slika 4.11.

Pri celotnem procesu kalibriranja moramo biti zelo previdni. Če smo storili manjšo napako ta vpliva za točnost kalibriranja in posledično na točnost izračunov o lokaciji oznake. Kalibracijo je zato priporočljivo izvesti večkrat, primerjati rezultate in izbrati najboljšo kalibracijo.



Slika 4.10: Prvi del kalibriranje kamere. Sliki povzeti po [41].



Slika 4.11: Postopek kalibriranja v drugem koraku. Sliki povzeti po [35].

4.4 ARToolKit in AR.Drone SDK

Da lahko v razvojnem okolju uporabimo funkcionalnosti ARToolKita, knjižnicam razvojnega okolja dodamo še knjižnice ARToolKita. Dodati je potrebno vsaj knjižnice za prepoznavanje oznake in izračun lokacije prepoznane oznake, po želji pa lahko dodamo vse knjižnice. Nadaljnja uporaba je zelo preprosta. Kodo za zaznavanje v osnovi sestavlja nekaj vrstic, te so:

```
If(arDetectMarker(new_pix,thresh,&marker_info,&marker_num)<0 ){
  // pojavila se je napaka, ne nadeljujemo
}
else{
  k = -1;
  for( j = 0;  j < marker_num; j++ ){
    if ( patt_id == marker_info[j].id ){
```

```
      if( k == -1 ) k = j;
      else if( marker_info[k].cf < marker_info[j].cf ) k = j;
    }
  }
  If( k >= 0 ){
    If( mode == 0 ){
      arGetTransMat(&marker_info[k],patt_center,patt_width,
patt_trans );
      mode = 1;
    }
    else{
      arGetTransMatCont( &marker_info[k], patt_trans, patt_center,
patt_width, patt_trans );
    }
  }
}
```

Funkcija *arDetectMarker* sprejme slednje parametre:

- *new_pix* je tabela tipa *unsigned char* velikosti $176 \times 144 \times 3$ in predstavlja sliko zapisano v RGB formatu.
- *thresh* je tipa *int* in predstavlja prag za določitev črne in bele slikovne točke.
- *marker_info* je tipa *ARMarkerInfo* in vsebuje podatke o nastavljeni oznaki.
- *marker_num* je tipa *int* in vsebuje število najdenih oznak v trenutni sliki.

Naslednja pomembna funkcija sta *arGetTransMat* in *arGetTransMatCont*, ki pa sta skoraj identični. Prva funkcija se uporabi le ko zajamemo prvo sliko z

oznako, drugo funkcijo pa uporabljamo pri vseh ostalih slikah, saj kot vhodni parameter upošteva tudi predhodno matriko o lokaciji kamere. Posledično je naslednja matrika o lokaciji bolj točna. Parametri ki jih prejme funkcija *arGetTransMatCont*:

- *marker_info* je tipa *ARMarkerInfo* in vsebuje podatke o oznaki.
- *Patt_Trans* je dvodimenzionalna tabela velikosti 3×4 tipa *double* in vsebuje lokacijo oznake v kordinatnem sistemu kamere.
- *Patt_center* je enodimenzionalna tabela z dvema vrednostima tipa *double*, ki določata lokacijo vzorca v oznaki
- *Patt_width* je tipa *int* in predstavlja dolžino stranice oznake v milimetrih.

Dobljene podatke o lokaciji lahko uporabimo po želji. Lahko jih uporabimo za izris poljubnega lika ali pa jih uporabimo za krmiljenje kvadrokopterja.

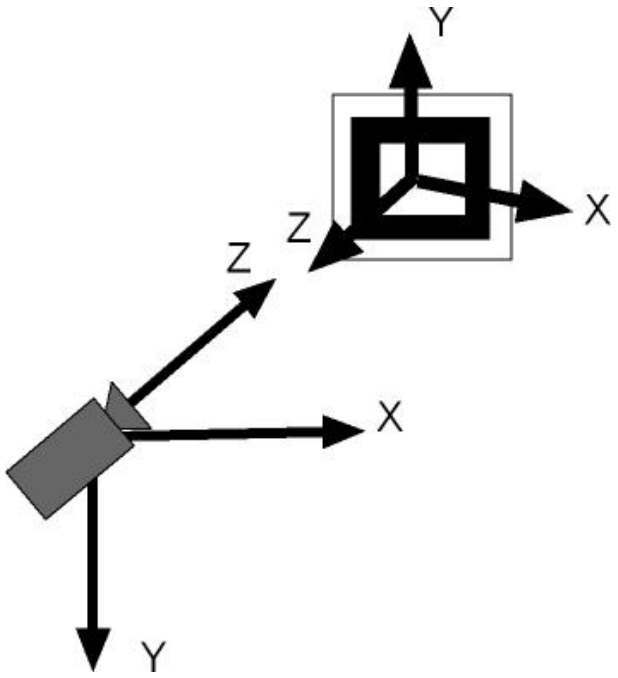
4.5 Koordinatni sistem in krmiljenje

Za dosego cilja potrebujemo pravilno krmiljenje kvadrokopterja. Z uporabo razvojnega okolja se krmiljenje poenostavi. Kvadrokopter krmilimo tako da kličemo funkcijo:

ardrone_at_set_progress_cmd(int flag, float roll, float pitch, float gaz, float yaw);

Podani argumenti predstavljajo število v plavajoči vejici in lahko zavzamejo vrednost med -1 in 1. Vrednost predstavlja odstotek predhodno nastavljene največje vrednosti. Slika 2.6 prikazuje, kako podani argumenti vplivajo na premik kvadrokopterja.

Koordinatni sistem se razlikuje od koordinatnega sistema, ki ga uporablja ARToolKit. Slednji uporablja koordinatni sistem vertikalne kamere kvadrokopterja, z izhodiščem v sredini slike, kar nazorno prikazuje slika 4.12.



Slika 4.12: Prikaz kordinatnega sistema v kamere in oznake. Slika povzeta po [37].

Lokacijo oznake iz koordinatnega sistema kamere pretvorimo v koordinatni sistem kvadrokopterja po enačbi 4.8.

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_K \\ Y_K \\ Z_K \end{bmatrix} \tag{4.8}$$

Kvadokopter premikamo po njegovi osi x tako, da ga rotiramo okoli njegove y koordinate. Če ga rotiramo v smeri urinega kazalca, se kvadrokopter začne premikati vzvratno. Obratno je v primeru, da ga rotiramo v nasprotni smeri urinega kazalca. Tedaj se prične premikati naprej. Na vpliv smeri rotacije in koliko ta znaša vplivamo tako, da podamo ustrezno vrednost parametru *pitch*. Če podamo vrednost 0, se kvadrokopter povrne v vodoravno stanje. Vrednosti med 0 in 1 pomeni rotacijo v smeri urinega kazalca, vrednosti med -1 in 0 pa povzročijo rotacijo v nasprotni smeri urinega kazalca.

Tabela 4.1: Vpliv vrednosti parametrov na posamezno os.

Parameter	Vrednost	Učinek
pitch	med 0 in 1	Pomik vzratno po osi x.
	med −1 on 0	Pomik naprej po osi x.
roll	med 0 in 1	Pomik desno po osi y.
	med −1 on 0	Pomik levo po osi y.
gaz	med 0 in 1	Pomik navzgor po osi z.
	med −1 on 0	Pomik navzdol po osi z.
yaw	med 0 in 1	Zasuk v desno po osi z.
	med −1 on 0	Zasuk v levo po osi z.

Za koliko stopinj se nagne je odvisno od predhodno nastavljenega največjega naklona in podane vrednosti. Odklon se nato izračuna tako, da se obe vrednosti zmnoži. Ta princip krmiljenja se uporablja tudi za pomik po osi y in z ter rotaciji okoli osi z. Pri pomiku po osi z je razlika le, da je predhodno nastavljena največja vertikalna hitrost, pri rotaciji okoli osi z pa je predhodno nastavljena največja kotna hitrost. Opisano prikazuje tabela 4.5.

Pri krmiljenju kvadrokopterja smo predpostavili, da se njegov naklon ne spreminja, oziroma se spreminja zelo malo. Tako dobimo natančnejše podatke o lokaciji oznake. Pomeni tudi, da moramo argumentom podajati nizke vrednosti. Upoštevali smo tudi, da pristajalni stožci dovoljujejo, da je kvadrokopter med pristajanjem od idealne pozicije oddaljen največ 3 centimetre.

Zaporedje izračunov ukazov, ki se zgodijo za vsako iteracijo slike, prikazuje algoritem 1. Vrednosti vhodnih spremenljivk X_A, Y_A in Z_A dobimo po enačbi 4.8, vrednost *orientacija_oznake* pa iz orientacije oznake. Spremenljivko *odmik* nastavimo po potrebi in predstavlja največji dovoljen odmik kvadrokopterja iz idealne pozicije.

Kvadrokopter moramo pred pristankom čim bolj približati pristajalni ploščadi. Bližje kot je, večja je verjetnost za uspešen pristanek. Kvadrokop-

Algorithm 1 iteracijo za eno sliko

Vhodne spremenljvke: $X_A,\ Y_A,\ Z_A$ *in* $orientacija\_oznake$

if $X_A > odmik$ **then**
  $pitch = procent\_naklona$
else if $X_A < -odmik$ **then**
  $pitch = -procent\_naklona$
else
  $pitch = 0$
end if
if $Y_A > odmik$ **then**
  $roll = procent\_naklona$
else if $Y_A < -odmik$ **then**
  $roll = -procent\_naklona$
else
  $roll = 0$
end if
if $Z_A > visina$ **then**
  $gaz = procent\_hitrosti$
else if $Z_A < visina$ **then**
  $gaz = -procent\_hitrosti$
else
  $gaz = 0$
end if
$yaw = orientacija\_oznake/180$
Izhodne spremenljivke: $Vrednosti\ premikov\ pitch,\ roll,\ gaz\ in\ yaw.$

ter približamo pristajalni ploščadi tako, da postopoma zmanjšujemo vrednost spremenljivke *visina*. Vrednost spremenljivki znižamo samo v primeru, da je kvadrokopter že dosegel trenutno nastavljeno višino in se nahaja v radiju 3cm od idealne pozicije. Višino nato znižujemo vse dokler ni kvadrokopter na višini 400mm ali manj. Tedaj je dovolj blizu pristajalne ploščadi in kličemo funkcijo z argumentom 0:

ardrone_tool_set_ui_pad_start(0);

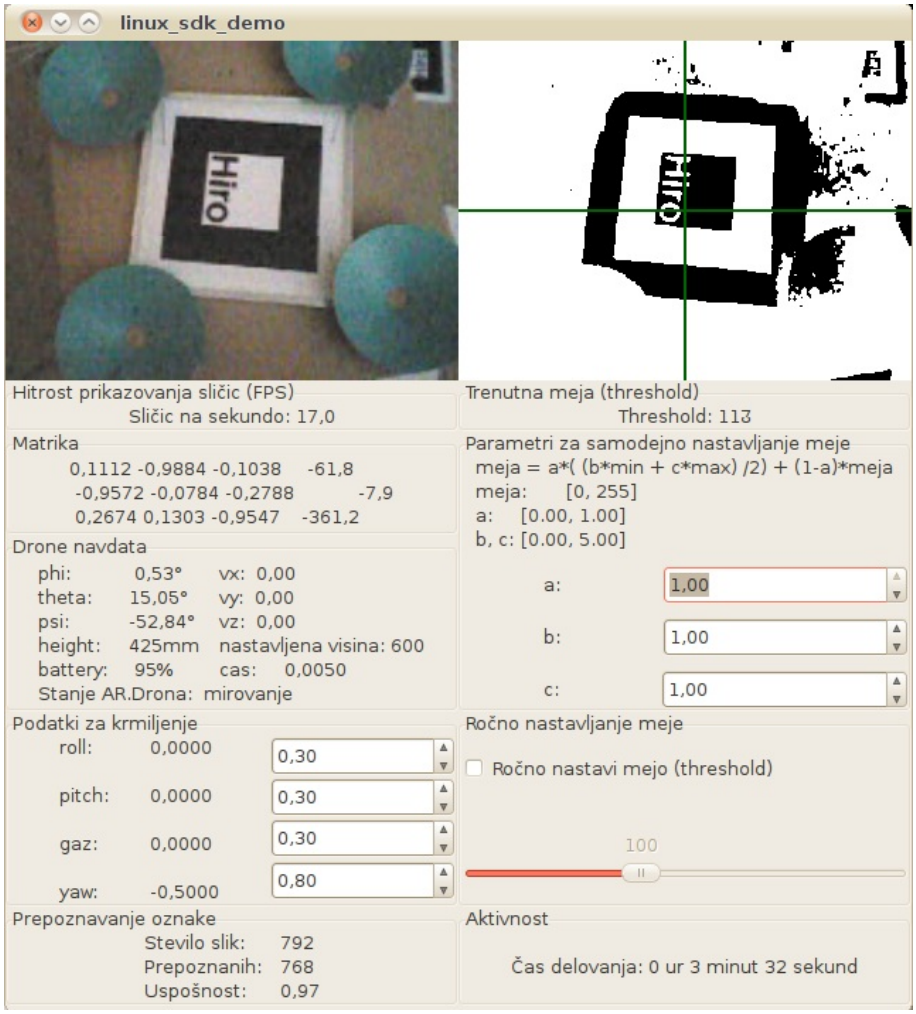
Po klicu funkcije se kvadrokopter prične počasi spuščati proti pristajalni ploščadi.

4.6 Grafični vmesnik

Grafični vmesnik smo naredili z razlogom, da olajšamo delo uporabniku in hkrati povečamo funkcionalnost programa. Vmesnik je izdelan v GTK+ 2.0 [38] in je nadgradnja enostavnega vmesnika [39], ki je prikazoval video ter je imel dva gumba za vzletanje in pristajanje kvadrokopterja. Vmesnik smo nadgradili tako, da prikazuje dve sliki, prva je barvna, druga črno-bela ter približno ponazarja video, kot ga vidi ARTookit, prikazuje nekaj pomembnih parametrov ter nam omogoča, da določene parametre nastavimo tudi sami.

Na vrhu se prikazujeta videa, levi je nespremenjen, desni pa prikazuje video po opravljeni segmentaciji. Slednjemu smo dodali še horizontalno in vertikalno zeleno črto s presečiščem v sredini videa. Pod levim videom se nahajajo:

- *Hitrost prikazovanja sličic (FPS)*: Izpisuje oceno prejetih sličic v sekundi.
- *Matrika*: Prikazuje podatke o lokaciji oznake v koordinatnem sistemu kamere.
- *Drone navdata*: Prikazuje podatke o nagibu, višini, stanju baterije in ocenjeni hitrosti kvadrokopterja. Na koncu je prikazano trenutno stanje kvadrokopterja.



Slika 4.13: Grafični vmesnik, ki prikazuje video in ostale podatke.

- *Podatki za krmiljenje*: Prikazane so vrednosti, ki jih pošiljamo kvadrokopterju. Poleg tega lahko nastavimo največjo vrednost, ki velja v primeru ročnega krmiljenja.

- *Prepoznavanje oznake*: V času lebdenja in pristajanja se izpisuje število prejetih slik, število slik, na katerih je bila zaznana oznaka in statistika.

Pod desnim videom se nahajajo:

- *Trenutna meja (threshold)*: Prikazuje trenutno vrednost pragu za segmentacijo slike.

- *Parametri za samodejno nastavljanje meje*: Samodejni izračun vrednosti pragu se izvede tako, da se v zajeti sliki vzame povprečje najsvetlejše in najtemnejše slikovne točke. Na vrednost lahko vplivamo z nastavitvijo treh parametrov, nova vrednost pa se vedno nahaja med 0 in 255.

- *Ročno nastavljanje meje*: V primeru aktiviranja, lahko z drsnikom ročno nastavimo vrednost za prag pri segmentaciji.

- *Aktivnost*: Prikazuje čas povezanosti s kvadrokopterjem.

4.7 Samodejno pristajanje

Samodejno pristajanje nam ne deluje dobro. Pri krmiljenju smo upoštevali napako o podani lokaciji oznake, kar pa ni bilo dovolj. Algoritem za pristajanje deluje in je dobro zasnovan, težava se pojavi pri krmiljenju. V idealnem svetu, kjer je kvadrokopter idealno zračno plovilo in ni prisotnih vplivov iz okolice, lahko kvadrokopter krmilimo tako, da se počasi premika k oznaki. To pa ne velja za resnični svet.

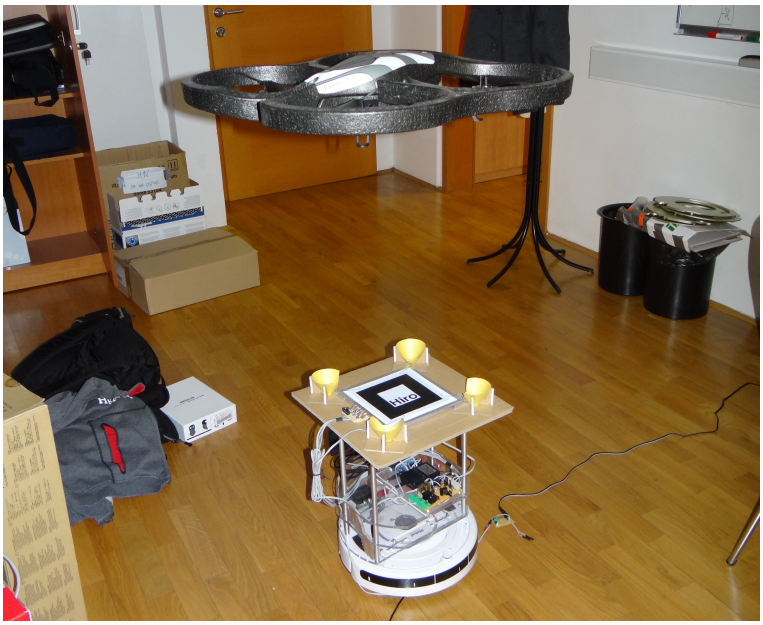
Pri krmiljenju moramo upoštevati tudi ostale dejavnike, ki vplivajo na nestabilnost kvadrokopterja. Najbolj izrazita dejavnika sta veter in vgrajeni senzorji nagiba. Največji vpliv vetra se pojavi na odprtih prostorih, kjer je prisoten naravni veter ali v majhnih in zaprtih prostorih, kjer kvadrokopter

ustvari vrtinčenje zraka. Najbližje idealnim razmeram ustreza velik prostor z enakomerno ogretim zrakom. Vgrajene senzorje nagiba kvadrokopter uporablja za izravnavo svoje lege. Ti senzorji niso idealni in so zato meritve nenatančne. Zaradi napak v meritvah se kvadrokopter prične počasi premikati.

Vpliv nezaželenih dejavnikov lahko zmanjšamo s pošiljanjem ukazov za večji nagib. Tako povečamo odzivnost krmiljenja, zaradi velike odzivnosti pa prične kvadrokopter vesti nemirno. To se izkaže za zelo škodljivo. Z večanjem višine se zaradi zakasnitve videa povečuje napaka o lokaciji oznake. Zaradi hitro spreminjajoče se lege oznake v videu se zmanjša tudi točnost transformacijske matrike T_{cm} (poglavje 4.3.2). Z nižanjem višine postane težava v velikosti slike, ki jo zajame vertikalna kamera. Vertikalna kamera ima majhen vidni kot (poglavje 2.1), kar pri nizkih višinah in velikih odklonih povzroči, da kamera ne zajame celotno oznako. Ker ARToolKit za določanje lokacije potrebuje celotno oznako, ne moremo popraviti lokacije kvadrokopterja. To se pripeti tudi v primeru kratkotrajne izgube video vsebine.

Kvadrokopter se pri krmiljenju prične vesti bolj mirno ko mu pošljemo ukaze za majhen nagib. Točnost izračuna lokacije se z večanjem višine ne znižuje drastično, saj se oznaka na zajetem videu premika počasi in je transformacijska matrika T_{cm} bolj točna. Tudi izguba manjšega števila slik v videu ne predstavlja velike ovire, saj se kvadrokopter premika počasi in imamo za krmiljenje na voljo več časa. Zaradi majhnega spreminjanja naklona kvadrokopterja, lahko vertikalna kamera tudi pri zelo nizki višini zajame celotno oznako, zaradi česar lahko kvadrokopter krmilimo bljižje pristajalni ploščadi, kar pa poveča verjetnost za uspešen pristanek. Se pa poveča vpliv ostalih dejavnikov, ki vplivajo na nekontrolirano premikanje kvadrkopterja.

Težava je v določanju najprimernejšega krmiljenja. Poskusili smo več prijemov za krmiljenje. Najbolj učinkovito je krmiljenje z majhnimi nagibi kjer upoštevamo tudi hitrost premikanja kvadrokopterja in njegove lege. V primeru prevelike hitrosti ali prevelikega nagiba pošljemo ukaz z ustreznimi parametri, ki upočasnijo in uravnajo kvadrokopter. Podatki o hitrosti in



Slika 4.14: AR.Drone med poskušanjem lebdenja nad pristajalno ploščadjo.

naklonu vsebujejo napako, zato je zelo težko določiti najprimernejše vrednosti za stabilizacijo kvadrokopterja, zato je to v načrtu za kasnejše delo.

Poglavje 5

Sklep

V diplomski nalogi smo izdelali sistem, ki omogoča avtonomno pristajanje kvadrokopterja na mobilni platformi za samodejno polnjenje akumulatorja. Izdelali smo zanimiv koncept pristajalne ploščadi, ki je enostaven in zadovoljivo služi v pomoč pri pristajanju in preprečevanju zdrsa kvadrokopterja s ploščadi. Izdelali smo tudi namensko krmilno vezje, ki je pri svoji nalogi zelo uspešno. Pri izdelavi krmilnega vezja smo posegli v nam manj poznano področje elektrotehnike. Pravi tehnični izziv pa je predstavljalo krmiljenje kvadrokopterja s pomočjo vizualnega zaznavanja. Za zaznavanje oznake smo uporabili ARToolKit, ker je natančen in poda lokacijo in orientacijo oznake. Oviro pri tem predstavlja le oznaka. Zaradi njene kompleksneše zgradbe v slabših pogojih natančnost dobljenih podatkov pade ali pa jih ARToolKit ne more priskrbeti. Naslednji velik izziv je bilo krmiljenja kvadrokopterja z dobljenimi podatki. Kvadrokopter ni idealno plovilo in se tudi ne nahaja v idealnem svetu, zato so vedno prisotne motnje. Če kvadrokopterju ne pošiljamo ukazov za premik se ta vseeno premika. Ti premiki so posledica zunanjih vplivov, predvsem vetra, ki ga v manjših prostorih ustvari kar sam. Svoj del pri neželenem premikanju dodajo tudi notranji senzorji, s pomočjo katerih se stabilizira. AR.Drone je namenjen igri zato je opremljen z manj natančnimi senzorji.

5.1 Možnosti za izboljšavo

Možnosti za nadgradnjo je veliko, saj je to le koncept. Nekaj predlogov smo že omenili v predhodnih poglavjih in se jih bomo še enkrat dotaknili. Čeprav je koncept stožcev za pristajanje zanimiv in enostaven, je pri pristajanju potrebna sorazmerno velika natančnost. S strojne strani, bi lahko ta problem delno rešili z uporabo širših stožcev. Trenutni premer, ki znaša približno 6cm, bi lahko povečali na 12cm. Tako bi podvojili uspešnost pristajanja, hkrati pa nadgradnja ni zahtevna. Lahko tudi preuredimo pristajalno ploščad, odstranimo stožce in kontakte za polnjenje ter uporabimo tehnologijo za brezžično polnjenje. Poleg tega bi morali v kvadrokopter vgraditi tudi LiPo polnilnik s krmilnim vezjem. Slednje se da narediti zelo majhno in težko le nekaj gramov. Večjo oviro predstavlja sprejemna tuljava. Lažja tuljava pomeni manj ovojev tuljave, kar bi pripomoglo k dolgemu polnjenju. V nasprotnem primeru bi bilo polnjenje z večjo tuljavo hitrejše, toda naraste tudi dodatna teža. Vseeno bi bilo vredno preizkusiti tudi ta koncept.

Pri trenutnem načinu polnjenja bi lahko kontakte na pristajalni ploščadi zamenjali z mehkejšimi, s čimer bi se izognili izgubljanju stika. Na hitrost polnjenja ne moremo veliko vplivati. LiPo baterije, trenutno najbolj dostopne na trgu, lahko polnimo s tokom 1C, kar nanese na $1A \times (kapaciteta\ baterije)$. Lahko pa uporabimo baterijo z večjo kapaciteto, katero bomo z dovolj zmogljivim polnilnikom napolnili približno tako hitro kot originalno baterijo. Polnilni čas ostane skoraj nespremenjen, podaljša pa se čas letenja.

Naslednja možnost nadgradnje lahko izvedemo na strani mikrokrmilnika. Trenutno za signaliziranje stanja obeh baterij, LiPo in akumulatorja, ponazarjamo s pomočjo svetlečih diod. Slednje bi lahko nadomestili z manjšim LCD zaslonom, ki lahko prikazuje 2 vrstici po 16 znakov. Z vgradnjo dodatnih tipk in nadgradnjo programske kode mikrokrmilnika bi lahko sproti nastavljali nastavitve, razbrali napetost na obeh baterijah, čas polnjenja in podobno. Mikrokrmilnik lahko tudi direktno povežemo z računalnikom, ki krmili kvadrokopter. Tako bi lahko preko računalnika vklopili kvadrokopter ali aktivirali polnjenje baterije.

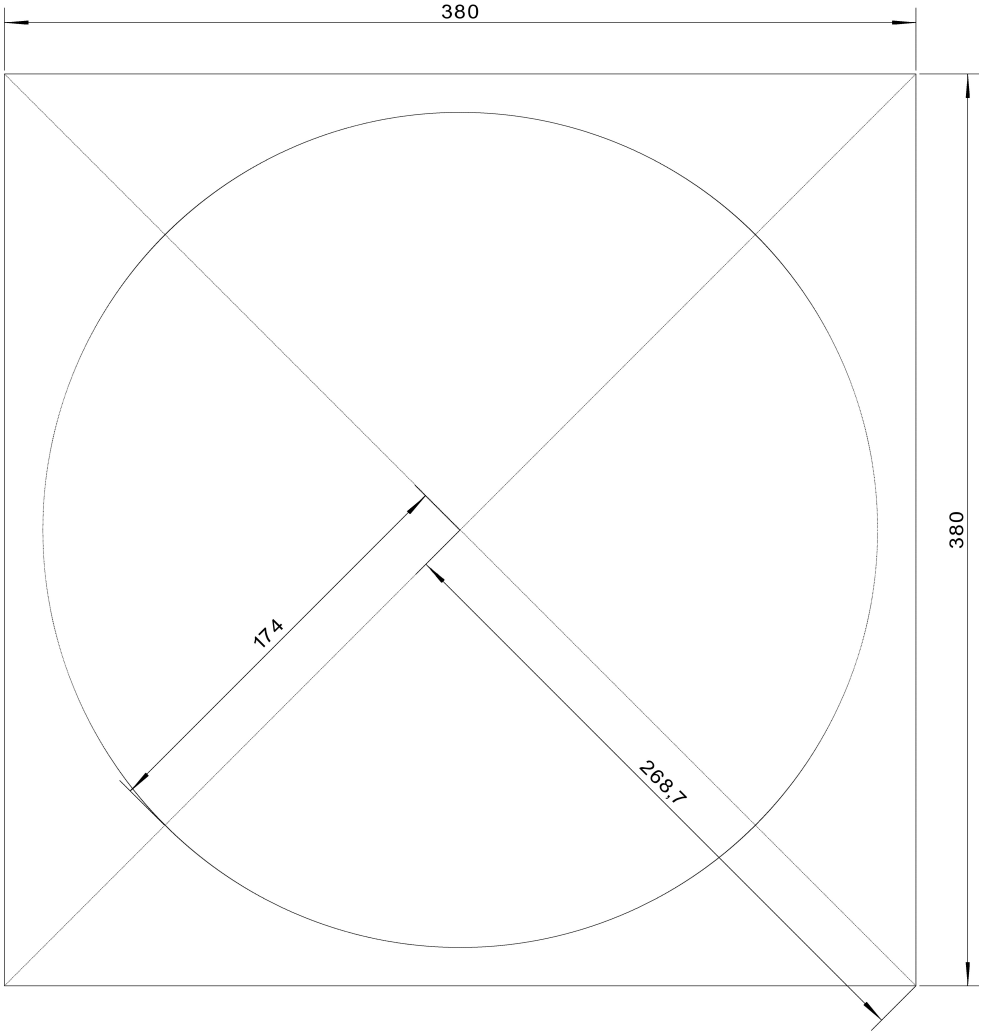
Na strani programske opreme bi lahko poizkusili tudi ostale pristope za vizualno zaznavanje. Pri procesu pristajanja bi lahko za natančnejše krmiljenje uporabljali tudi podatke o trenutni oceni naklona in hitrosti kvadrokopterja. To bi kvadrokopter napravilo bolj robustnega v slabših pogojih, kot so rahel veter ali drugi vplivi iz okolice.

Dodatek A

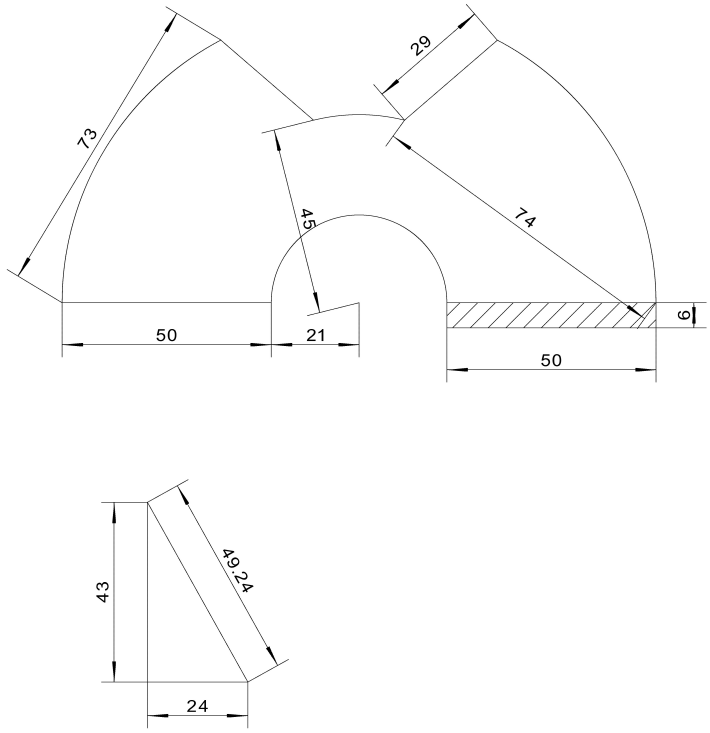
V sledečem poglavju se nahajajo priloge o merah pristajalne ploščadi, stožca in podpornika. Te mere lahko uporabimo za izdelavo nove pristajalne ploščadi. Priložene so tudi sheme elektronskih vezij, ki smo jih izdelali v sklopu diplome. Na koncu se nahaja še kratek vodnik, kako dodamo podporo za nepodprti igralni plošček.

A.1 Tehnične risbe

Vse mere na shemah so navedene v milimetrih. Za izdelavo Pristajalne ploščadi in pripadajočih komponent lahko poleg kartona uporabimo tudi druge, trpežnešje materiale.



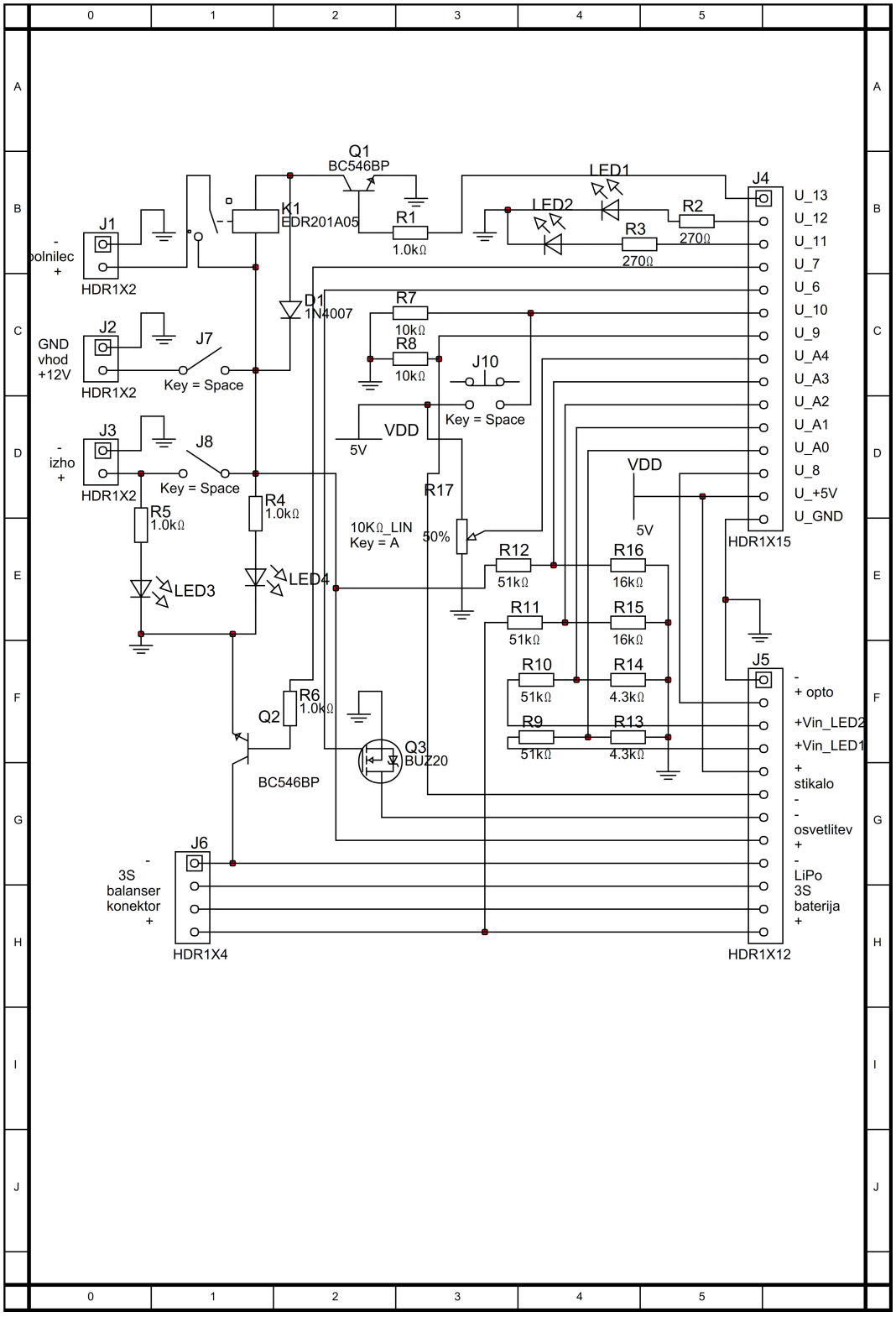
Slika A.1: Načrt pristajalne ploščadi.



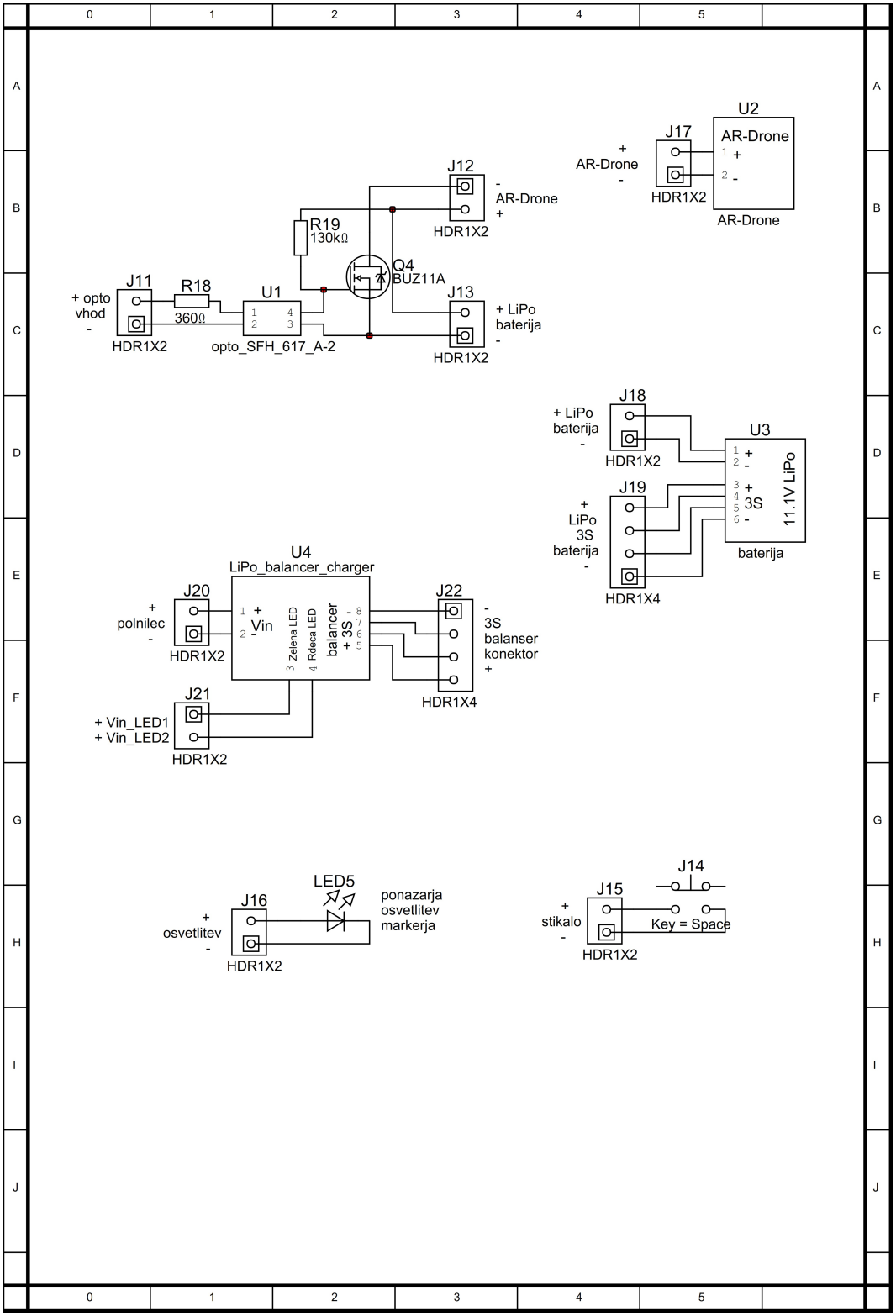
Slika A.2: Načrt stožca in podpornika.

A.2 Shematski načrt vezij

Pri izdelavi elektronskih vezij smo uporabili komponente, ki so se v danem trenutku zdele najbolj primerne. Posledično se je izkazalo, da bi lahko uporabili druge, bolj primerne komponente, kar pa lahko storimo pri naslednji izdelavi elektronskih vezij. Shemo lahko tudi dogradimo, da ustreza točno določenim zahtevam ali predelamo po potrebi. Slika A.3 prikazuje shematski načrt za glavno ploščico, medtem ko slika A.4 prikazuje shematske načrte ostalih, manjših, elektronsih vezij.



Slika A.3: Shema glavnega elektronskega vezja.



Slika A.4: Shema preostalih elektronskih vezij.

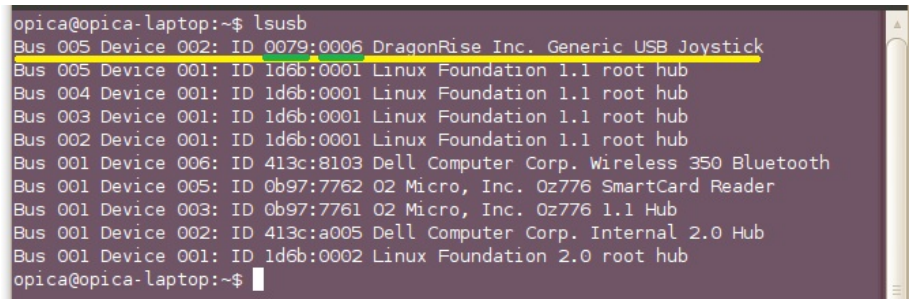
A.3 Uporaba poljubnega igralnega ploščka

Razvojno okolje SDK 1.8 ima ima omejeno podporo igralnim ploščkom. Če si podprtega igralnega ploščka ne lastimo imamo na voljo dve možnosti. Lahko si kupimo podprti igralni plošček ali pa preuredimo kodo. V nadaljevanju se nahaja kratek napotek kako hitro in enostavno dodamo podporo za plošček, ki si ga lastimo. Podporo za igralni plošček naredimo na sledeč način. Igralni plošček priklopimo na računalnik. V terminal vpišemo ukaz *lsusb*. Prikaže se izpis vseh USB naprav, ki so trenutno priklopljene na računalnik. Primer izpisa kar prikazuje slika A.5. Poiščemo vrstico z želeno napravo. Izpišemo 8 mestno ID številko, brez »*:*«, torej, zapis iz oblike *XXXX:XXXX* prepišemo v obliko *0xXXXXXXXX*, kar predstavlja šestnajstiški zapis. Odpremo datoteko *gamepad.h*, ki se nahaja v *SDK1.8/Examples/Linux/sdk_demo/Sources/UI*. Nato v vrstici 6:

#define GAMEPAD_LOGICTECH_ID 0x046dc21a

zamenjamo z:

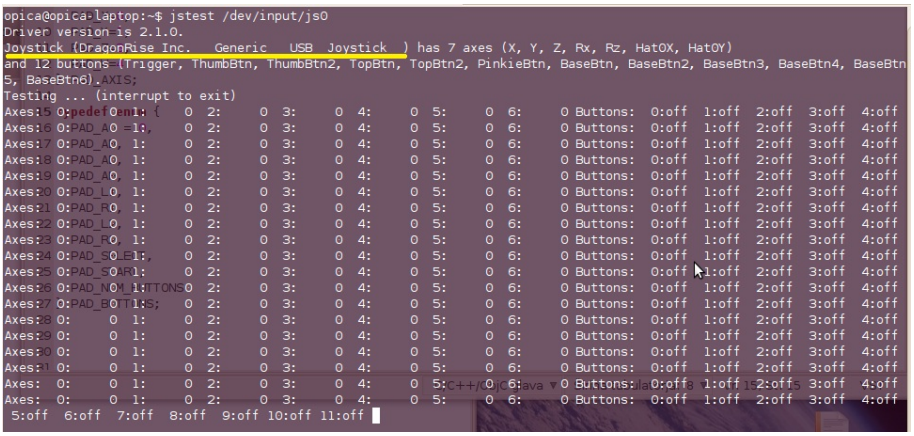
#define GAMEPAD_LOGICTECH_ID 0xXXXXXXXX



Slika A.5: Prikaz izpisa *lsusb*.

Sedaj lahko uporabimo poljubni igralni plošček, ki prvotno ni podprt.

Pred uporabo je priporočljivo preveriti kaj igralni plošček pošilja računalniku in pod katero številko se posamezne tipke predstavijo računalniku. Če tega še nismo storili, si namestimo *joystick*. Namestimo ga na sledeč način. V terminal napišemo *sudo apt-get install joystick* in zaključimo z namestitvijo. Po namestitvi v terminal napišemo *ls /dev/input/js** kar nam izpiše vse »*joystick*« naprave. V primeru, da imamo priklopljen le naš igralni plošček se bo izpisalo le */dev/input/js0*, v nasprotnem primeru, pa bo vsaka naslednja naprava izpisana kot */dev/input/js1* in tako dalje. Več naprav nam lahko izpiše tudi v primeru da ima računalnik vgrajene notranje senzorja za nagib. Naposled v terminal napišemo še ukaz *jstest /dev/input/js0* ali namesto *js0* uporabimo ustrezen *js**, če prvič nismo dobili pravega izpisa. Primer izpisa prikazuje slika A.6, za katerega vemo da je pravilni, saj vsebuje ime naprave. Sedaj lahko v datoteki *gamepad.h* določimo vrstni red tipk in osi.



Slika A.6: Prikaz izpisa *jstest*.

Po ponovnem prevajanju kode in zagonu prevedene kode lahko sedaj z igralnim ploščkom krmilimo kvadrokopter kot če bi uporabili igralni plošček igralne konzole Playstation 3 ali če bi krmilili kvadrokopter preko telefona.

Literatura

[1] United States Army Air Service - predstavitev in zgodovina (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/United_States_Army_Air_Service

[2] De Bothezat helikopter - predstavitev in zgodovina eksperimentalnega kvadrokopterja (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/De_Bothezat_helicopter

[3] Curiosity - opis misije in tehničnih značilnosti (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/Curiosity_rover#Specifications

[4] Predstavitev zgodovine sond za raziskovanje Marsa (2013). Dostopno na:
http://marsrovers.jpl.nasa.gov/gallery/press/opportunity/20120117a.html

[5] AR.Drone tehnične specifikacije (2013). Dostopno na:
http://ardrone.parrot.com/parrot-ar-drone/en/technologies

[6] Avdio video oddajnik z večjim dometom (2013). Dostopno na:
http://www.dronesvision.com/lawmate-2-4ghz-500mw-wireless-av-transmitter/

[7] Pilotiranje po principu prvoosebnega pogleda (ang. First-person view ali FPV)(Wikipedia) (2013). Dostopno na:

http://en.wikipedia.org/wiki/Radio-controlled_aircraft#Video_piloting_.28first-person_view.29

[8] Opis in povzetek o kvadrokopterju AR.Drone (Wikipedia)(2013). Dostopno na:
http://en.wikipedia.org/wiki/Parrot_AR.Drone

[9] Computer Electronic Show (CES) - predstavitev in zgodovina mednarodnega tehnološkega dogodka (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/International_Consumer_Electronics_Show

[10] Arduino - uradna spletna stran (2013). Dostopno na:
www.arduino.cc

[11] Primeri programov, namenjenih za Arduino (2013). Dostopno na:
http://arduino.cc/en/Tutorial/HomePage

[12] Razvojno okolje za pisanje skic za Arduino (2013). Dostopno na:
http://arduino.cc/en/Main/Software

[13] Podatkovni tipi in funkcije za pisanje skic (2013). Dostopno na:
http://arduino.cc/en/Reference/HomePage

[14] Arduino elektronska vezja trenutno na voljo (2013). Dostopno na:
http://arduino.cc/en/Main/Hardware

[15] Arduino Uno - podroben opis tehničnih specifikacij in predstavitev (2013). Dostopno na:
http://arduino.cc/en/Main/ArduinoBoardUno

[16] Primer domače izdelave kvadrokopterja (2013). Dostopno na:
http://mad-science.wonderhowto.com/inspiration/arduino-air-force-diy-robotic-cardboard-quadcopters-0134247/

[17] Robot krmiljen s pomočjo Arduina (2013). Dostopno na:
`http://www.devinmiller.biz/2012/04/25/new-category-arduino-robot/`

[18] Primeri uporabe ploščice Arduino v robotiki (2013). Dostopno na:
`http://www.mhobbies.com/blog/robotics-tweezers-work-at-cellular-standard/`

[19] Različni formati prototipnih ploščic (2013). Dostopno na:
`http://www.produktinfo.conrad.com/datenblaetter/525000-549999/529593-da-01-de-LOETPUNKTRASTERPLATTE_FR2.pdf`

[20] Tehnična dokumentacija za N-kanalni MOSFET tranzistor BUZ11 (2013). Dostopno na:
`http://www.datasheetcatalog.org/datasheets/50/118235_DS.pdf`

[21] Tehnična dokumentacija elementa SFH617A (2013). Dostopno na:
`http://www.vishay.com/docs/83740/sfh617a.pdf`

[22] Princip delovanja svinčevega akumulatorja (2013). Dostopno na:
`http://www2.arnes.si/~afirma/pouk/akku.htm`

[23] Tehnična dokumentacija polnilnika podjetja Graupner (2013). Dostopno na:
`http://www.manualslib.com/manual/402148/Graupner-Lipo-Balancer-Charger-2-3s.html?page=8#manual`

[24] Predstavitev in zgodovina podjetja Parrot (2013). Dostopno na:
`http://en.wikipedia.org/wiki/Parrot_(company)`

[25] AR.Drone razvojno okolje (2013). Dostopno na:
`https://projects.ardrone.org/`

[26] Vodnik za razvoj aplikacij z razvojnim okoljem za AR.Drone (2013). Dostopno na:

http://projects.ardrone.org/attachments/download/365/
ARDrone_SDK_1_7_Developer_Guide.pdf

[27] Uradni forum z napotki za rokovanje z razvojnim okoljem in AR.Dronom (2013). Dostopno na:
https://projects.ardrone.org/projects/ardrone-api/boards

[28] Neuradni forum z napotki za rokovanje z razvojnim okoljem in AR.Dronom (2013). Dostopno na:
http://www.ardrone-flyers.com/

[29] Kratek vodnik za razvoj aplikacij za kvadrokopter AR.Drone (2013). Dostopno na:
http://gauth.fr/ar-drone-tutorial/

[30] ARToolKit - domača stran programske opreme (2013). Dostopno na:
http://www.hitl.washington.edu/artoolkit/

[31] ARToolKit - opis in razlaga delovanja prepoznavanja oznake (2013). Dostopno na:
http://www.hitl.washington.edu/artoolkit/documentation/
userarwork.htm

[32] ARToolKit - razlage zgradbe programske opreme (2013). Dostopno na:
http://www.hitl.washington.edu/artoolkit/documentation/
devframework.htm

[33] ARToolKit - prenos programske opreme (2013). Dostopno na:
http://www.hitl.washington.edu/artoolkit/download/

[34] ARToolKit - napotki za namestitev programske opreme na operacijski sistem Linux (2013). Dostopno na:
http://www.hitl.washington.edu/artoolkit/documentation/
usersetup.htm#comp_linux

[35] ARToolKit - predstavitev in prikaz delovanja algoritma za zaznavanje oznake (2013). Dostopno na:
`http://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf`

[36] ARToolKit - predstavitveni dokument z opisom delovanja zaznavanja oznake (2013). Dostopno na:
`http://www.hitl.washington.edu/artoolkit/Papers/IWAR99.kato.pdf`

[37] ARToolKit - kordinatni sistem (2013). Dostopno na:
`http://www.hitl.washington.edu/artoolkit/documentation/cs.htm`

[38] GTK+ - predstavitev in napotki za uporabo (2013). Dostopno na:
`http://developer.gnome.org/platform-overview/stable/gtk`

[39] Vodnik za izdelavo grafičnega vmesnika v razvojnem okolju za AR.Drone (2013). Dostopno na:
`http://gauth.fr/2011/09/create-an-ar-drone-graphical-application/`

[40] ARToolKit - podrobnejši prikaz delovanja zaznavanja oznake (2013). Dostopno na:
`http://www.hitl.washington.edu/artoolkit/documentation/vision.htm`

[41] ARToolKit - postopek kalibracije kamere (2013). Dostopno na:
`http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm`

[42] Način osvetlitve LCD zaslonov (Wikipedia) (2013). Dostopno na:
`http://en.wikipedia.org/wiki/Backlight#LED_backlights`

[43] Osvetlitev LCD zaslonov s svetlečimi diodami (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/LED-backlit_LCD_display

[44] Princip delovanja osvetlitve LCD zaslonov (2013). Dostopno na:
http://www.iitk.ac.in/asid06/proceedings/papers/TC2_1-I.pdf

[45] Uradna spletna stran TurtleBot (2013). Dostopno na:
http://turtlebot.com/

[46] Uradna spletna stran podjetja iRobot (2013). Dostopno na:
http://www.irobot.com/en/us/robots/home/roomba.aspx

[47] kinect1) Opis in predstavitev Kinect (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/Kinect

[48] Razlaga delovanja pulzno-širinske modulacije (Wikipedia) (2013). Dostopno na:
http://en.wikipedia.org/wiki/Pulse-width_modulation