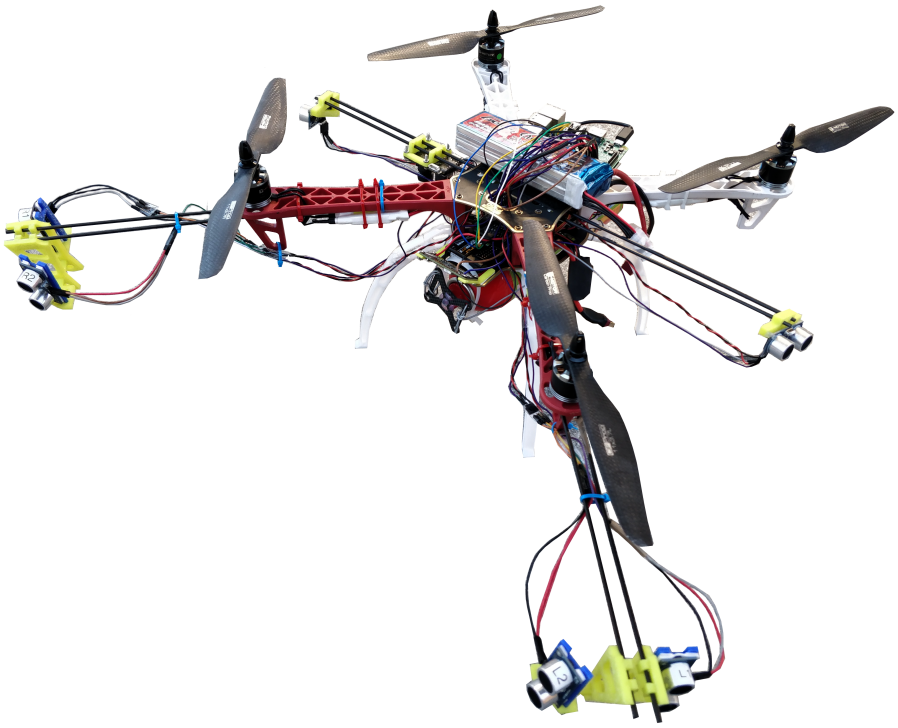
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Konstruktion av autonom drönare för eldbekämpning

Kandidatarbete vid Data- och Informationsteknik

ANGELICA STRANDBERG, DAVID ILIEFSKI-JANOLS,
JOHAN LABERG NILSSON, MAGNUS KARLSSON,
SIMON MARE, VIKTOR LINDBLOM

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

Kandidatarbete

Konstruktion av autonom drönare för eldbekämpning

ANGELICA STRANDBERG, DAVID ILIEFSKI-JANOLS,
JOHAN LABERG NILSSON, MAGNUS KARLSSON,
SIMON MARE, VIKTOR LINDBLOM

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2018

Konstruktion av autonom drönare
för eldbekämpning
ANGELICA STRANDBERG, DAVID ILIEFSKI-JANOLS,
JOHAN LABERG NILSSON, MAGNUS KARLSSON,
SIMON MARE, VIKTOR LINDBLOM

Handledare: Roger Johansson, Institutionen för Data- och Informationsteknik
Examinator: Arne Linde, Institutionen för Data- och Informationsteknik

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Sverige
Telefon +46 (0)31-772 1000

Omslag: Bilden föreställer den drönare som använts under detta projekt, utrustad med sensorer, beräkningsenhet och släckanordning i form av släckspray

Typsatt i LaTeX
Göteborg, 2018

Construction of an autonomous drone
for firefighting

Angelica Strandberg, David Iliefski-Janols, Johan Laberg Nilsson,
Magnus Karlsson, Simon Mare, Viktor Lindblom
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

This work comprises the development and construction of a drone designed to autonomously navigate a small area, aiming to identify and extinguish a minor fire source. A flight-ready drone was used as a base, and was extended with hardware for data-gathering and software for flight control.

The implemented hardware mounted on the drone consisted of ultrasonic sensors for distance measurements, a flow sensor for velocity measurements, a thermal camera for fire identification, a fire extinguishing spray and a Raspberry Pi for communicating with the components as well as with the drone's flight controller. The implemented software was executed on a separate computer and communicated wirelessly with the Raspberry Pi, controlling the drone by processing the gathered data through the use of control systems and an artificial neural network.

The work resulted in a drone that could autonomously take off, maintain a given height while maneuvering to avoid walls and simple obstacles, and land on command. Although partially implemented, the goal of identifying and extinguishing fire was never reached, due to time restrictions following problems and inconsistencies with the hardware components.

This report is written in Swedish.

Keywords: UAV, Drone, Autonomous, AI, Neural Network, Firefighting,
Quadcopter, Robot

Konstruktion av autonom drönare
för eldbekämpning

Angelica Strandberg, David Iliefski-Janols, Johan Laberg Nilsson,
Magnus Karlsson, Simon Mare, Viktor Lindblom
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Sammandrag

Detta arbete innefattar utveckling och konstruktion av en drönare designad att autonomt navigera ett litet utrymme i syfte att identifiera och släcka en mindre eldkälla. En flygfärdig drönare användes som bas och utökades sedan med hårdvara för datainsamling samt mjukvara för styrning.

Hårdvaran som implementerades och monterades på drönaren bestod av ultraljudssensorer för avståndsmätning, en flödessensor för hastighetsmätning, en värmekamera för eldidentifiering, en släckspray samt en Raspberry Pi för kommunikation med komponenterna och drönarens flight controller. Den implementerade mjukvaran exekverades på en separat dator och kommunicerade trådlöst med Raspberry Pi:n i syfte att kontrollera drönaren genom att processera den insamlade datan med hjälp av reglersystem och ett artificiellt neuralt nätverk.

Arbetet resulterade i en drönare som autonomt kunde lyfta, bibehålla en given höjd medan den manövrerade för att undvika väggar och simpla hinder, samt landa på kommando. Målet att identifiera och släcka eld, även om det var delvis implementerat, nåddes aldrig på grund av tidsrestriktioner till följd av problem och inkonsekvenser hos hårdvaran.

Nyckelord: UAV, Drönare, Autonom, Eldbekämpning, Brandsläckning, AI,
Neuralt Nätverk, Robot

Förord

Detta projekt gjordes som ett kandidatarbete på Chalmers Tekniska Högskola under vårterminen 2018. Vi vill rikta ett stort tack till vår handledare Roger Johansson för all hjälp under projektets gång med rapporten såväl som med drönarprototypen, samt till Lars Norén för hans hjälp med komponenter till drönaren. Vi vill också tacka Elektrosektionens Teletekniska Avdelning, ETA, för alla hårdvarubidrag till prototypen. Slutligen vill vi tacka David Frisk för LaTeX-mallen som rapporten är skriven i.

Notis: för alla figurer och foton i rapporten gäller, om inget annat anges, att de är tagna eller skapade av författarna själva.

Angelica Strandberg, David Iliefski-Janols, Johan Laberg Nilsson,
Magnus Karlsson, Simon Mare och Viktor Lindblom, Göteborg, Maj 2018

Förkortningar

API *Application Programming Interface*
Applikationsprogrammeringsgränssnitt inkluderar grundläggande funktioner och kommandon som kan användas vid programmering.

GPIO *General Purpose Input Output*
En port som kan konfigureras till att hantera indata eller utdata på ett datorsystem.

GPU *Graphical Processing Unit*
En typ av processor som huvudsaklingen används vid bild- och videorendering.

LIDAR *Light Detection and Ranging*
En optisk avståndsmätare.

PWM *Pulse-Width Modulation*
Pulsbreddsmudulation används till att styra elektriska apparater och innebär att apparatens effekt varieras med hjälp av en pulssignals pulsbredd.

UDP *User Datagram Protocol*
Ett nätverksprotokoll för snabb dataöverföring, men som inte garanterar att all data når mottagaren.

Innehåll

Kapitel 1

Inledning

De senaste åren har tekniker för autonom styrning, det vill säga självstyrning, av fordon fått mycket uppmärksamhet. Stora bilföretag som Volvo utvecklar bilar som ska klara sig i trafiken utan mänsklig styrning [1] genom att istället använda sensorer och ett datorsystem som tar beslut om hur styrningen ska ske.

Det är dock inte enbart bilar som utvecklas för självstyrning, utan även autonoma drönare (obemannade, flygande farkoster). Företag som Amazon har till exempel tagit fram system för att låta autonoma drönare leverera paket till människor [2], medans andra använder dem till att inspektera infrastruktur efter skador [3].

Manuellt flugna drönare har redan i dagsläget en utbredd användning inom ett flertal områden. Genom att eliminera behovet av en pilot vidgas de potentiella användningsområdena, och dessutom kan redan etablerade processer där drönare används i många fall också effektiviseras. Ett av dessa fall är eldbekämpning, vilket utgör en fokalpunkt i denna rapport.

Eldbekämpning genom användandet av drönare har tidigare förekommit av både privatpersoner [4] och myndigheter [5]. Genom självstyrning minskas de mänskliga resurser som krävs för flygning samtidigt som den potentiella precisionen under själva flygningen ökar, vilket öppnar möjligheter som till exempel att flyga koordinerade formationer av drönare för att bära vattentankar. Dessa skulle utgöra ett alternativ till de bemannade farkoster som idag används vid bekämpning av skogsbränder [6]. Genom ökad flygprecision skulle även bränder på svårnavigerade ytor såsom arenor, byggnadsplatser eller höghus också kunna bekämpas med hjälp av drönare.

Det kan tänkas att lokala brandstationer skulle kunna ha en uppsättning autonoma drönare som alltid står redo. Då dessa inte behöver någon förberedelse eller styrning och inte heller behöver förhålla sig till vägar skulle de kunna agera som ett snabbare svar vid utryckningar. Genom att utrustas med kameror och någon form av släckutrustning skulle de kunna ge tidig information om branden och potentiellt påbörja släckningsarbetet.

1.1 Syfte

Rapporten syftar till att beskriva, dokumentera, utforska samt analysera konstruktionen och utvecklingen av en prototyp för en eldbekämpande autonom drönare. Den färdigutvecklade prototypen ska under flygning kunna söka av ett område och navigera runt hinder samt lokalisera, identifiera och släcka en eldkälla.

1.2 Problembeskrivning

För att uppfylla syftet av att autonomt flyga, lokalisera och följaktligen släcka en eldkälla krävs utveckling och implementation av flera olika delsystem. Vilka delsystem som krävs klargörs genom uppdelning av projektet i flertalet delmål vilka redogörs för nedan.

1. **Anpassning av drönare**
   En mekanisk konstruktion behöver tas fram där samtliga komponenter samt släckanordning monteras så att drönaren kan utföra sitt uppdrag. Lyftkraften på drönaren behöver även anpassas så att den kan flyga stabilt med de extra komponenterna. Alla komponenter behöver även förses med en stabil strömförsörjning.

2. **Positionsbedömning i tre dimensioner**
   Ett system behöver utvecklas som kan bestämma drönarens avstånd till marken och eventuella objekt i omgivningen. Systemet behöver även vara medvetet om vilken riktning och hastighet drönaren färdas i för att göra självstyrning av drönaren möjlig.

3. **Autonom avsökning**
   Drönarens hastighet och höjd ovanför marken ska kunna styras autonomt. Dessutom ska ett system för att drönaren ska kunna navigera och söka av ett förutbestämt område utvecklas. Systemet ska också hjälpa drönaren att undvika hinder under flygningen.

4. **Identifiering av eldkälla**
   En algoritm ska implementeras som kan identifiera en eldkälla och skilja den från annat som utstrålar värme, som till exempel motorer, element och glödlampor. När en eldkälla är identifierad ska drönarens avsökning avslutas och släckning påbörjas.

5. **Släckning av eldkällan**
   Ett system för att släcka en eld behöver tas fram. Systemet ska först sikta på elden och därefter aktivera släckanordningen. När släckprocessen lyckats ska detta bekräftas och drönarens uppdrag avslutas.

Målet med projektet kan anses vara uppnått när alla delmål är uppfyllda och alla system därmed fungerar tillsammans.

1.3 Avgränsningar

Det finns ingen övre gräns på omfattningen och komplexiteten hos ett autonomt system, men då projektet är både tidsmässigt och budgetmässigt begränsat är vissa avgränsningar nödvändiga. Dessa avgränsningar innefattar var drönaren ska kunna flyga samt vad som ska utvecklas och implementeras på den, och redovisas nedan.

För det första kommer prototypen utvecklas utifrån en sedan tidigare flygfärdig drönare som bas, detta eftersom både kontrollsystem för drönarens stabilitet i luften samt dess elektriska system är tidskrävande och komplicerade att implementera. Dessutom sätts vissa begränsningar i hur mycket och vilken utrustning som kan monteras på drönaren utifrån dess fysiska kapaciteter så som lyftkraft, storlek och strömförsörjning.

Ytterligare en avgränsning som kommer göras är att systemet endast ska söka av ett begränsat område av enkel komplexitet. Detta förenklade område kommer endast att innehålla icke rörliga hinder som är i jämförbar storlek eller större än drönaren, inte innehålla hinder som kan befinna sig ovanför eller under drönaren, och drönaren kommer att befinna sig på en konstant höjd. Detta förenklar framförallt den omgivningsuppfattning systemet behöver ha och minskar på så sätt komplexiteten och kostnaden för sensorer, samtidigt som den autonoma navigationen begränsas till att endast behöva styra drönaren i två dimensioner.

Utöver detta kommer flygning av drönaren i den uppbyggda miljön att ta plats inomhus för att inte behöva ta hänsyn till väder eller till transportstyrelsens regelverk gällande flygning av drönare, då detta regelverk enbart gäller utomhus [7].

Vidare begränsas typer av möjliga eldkällor i den fördefinierade miljön till marschaller och värmeljus, då öppen eld är svårare att kontrollera och återskapa. Dessutom kommer endast en eldkälla åt gången vara aktiv i området.

1.4 Rapportens upplägg

För att få en bättre inblick i vilka delsystem och komponenter som behöver utvecklas och vad som redan finns att tillgå presenteras dessa först i kapitel 2. Här finns både en överblick över hur systemet ska byggas som stort och vad projektet innefattar i detalj, samt teori som krävs vid utvecklingen. Därefter beskrivs utvecklingen och implementationen i kapitel 3, där även mindre tester som var nödvändiga för att kunna vidareutveckla produkten beskrivs. I kapitel 4 presenteras resultaten från verifikation och sluttester av delmålen, vilka slutligen diskuteras i kapitel 5 tillsammans med möjligheter för vidareutveckling av projektet.

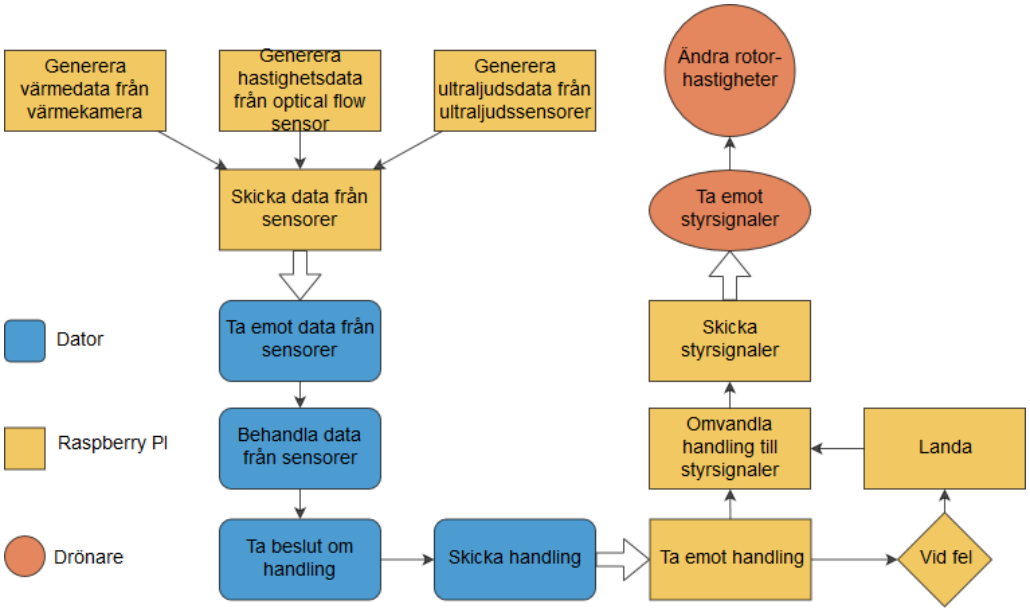
Kapitel 2

Komponenter och system

För att kunna lösa de fem olika delarna av projektet som redogjordes för i avsnitt 1.2 krävs att flera hårdvaru- och mjukvarukomponenter kopplas samman till olika delsystem. De delsystem som behöver utvecklas och de komponenter som behöver användas samt beskrivningar av dessa presenteras i detta kapitel.

De komponenter och system som sitter på drönaren kopplas in till en *Raspberry Pi v3* (benämns vidare som Raspberry Pi). Det finns flera anledningar till varför Raspberry Pi används. För det första är det en liten och lätt enhet som har en stor beräkningskraft för ett litet pris, och för det andra finns trådlösa uppkopplingsmöjligheter implementerade på chipet. Dessutom finns väldokumenterade och anpassningsbara operativsystem att tillgå vilket underlättar för utvecklingen av projektet samt den parallellism som systemet kan komma att behöva. Vidare finns olika programmeringsgränssnitt färdiga, till exempel för olika typer av kommunikationsprotokoll.

De system som inte sitter på drönaren exekveras på en separat laptop. Hur de olika delsystemen kopplas samman illustreras i Figur 2.1.



Figur 2.1: *Översikt över systemets uppbyggnad och vilka deluppgifter som bygger upp den autonoma drönaren.*

2.1 Komponenter för lyftkraft och strömförsörjning

Det första delmålet berör drönarens komponenter och strömförsörjning. I detta avsnitt kommer de delar som krävs för att ge drönaren dess lyftkraft samt de delar som levererar och reglerar strömmen till alla komponenterna att presenteras. Detta innefattar motorer, propellrar, batteri och en krets för att anpassa batterispänningen till de elektronikkomponeneter som monteras på drönaren, vilket även innefattar Raspberry Pi och sensorer.

Motorer och propellrar är de delar på en drönare som generar lyftkraft, vilket bestämmer drönarens maximala tillåtna vikt och därigenom säkerställer manövrerbar och säker flygning. Lyftkraften bör vara minst dubbelt så stor som drönarens totalvikt [8].

Drönarens originella propeller- och motorkombination bedöms ge otillräcklig lyftkraft. Genom en avvägning där drönarens totalvikt, storlek, ramkonstruktion och strömförbrukning tas i beaktning används istället *T-motor CF 11x3.7 tums* propellrar och *T-Motor MT2212-750kV* motorer [9].

Fördelen med denna propeller är att den är av kolfiber och därmed styvare, vilket ger bättre prestanda vid högre varvtal [10]. Storleken på propellern är precis så stor som ramen tillåter för en maximering av lyftkraft [11]. Stora propellrar är också mer energieffektiva och ger därmed längre flygtid [10], [11].

Den valda motorn har som fördel att den drar låg ström även vid full belastning vilket möjliggör ett lättare batteri. En nackdel är dock att motorn egentligen leverar lägre lyftkraft än rekommenderat, men enligt företaget RCFlight [12] bör det ändå fungera givet att inga avancerade flygmanövrar ska genomföras. För en mer djupgående analys av motorval,

lyftkraft och andra aspekter se Appendix A.1.

Vidare behövs ett lämpligt batteri för att förse alla komponenter med ström och motorerna med rätt spänning. Valet av batteri behöver ta hänsyn till batterityp och dess kapacitet, spänning, urladdningsström och vikt för att ge bra flygtid och lyftkraft. Utav motorerna krävs att batteriet kunde leverera ungefär $40\,A$ kontinuerlig ström och att spänningen var $14,8\,V$. Trots att många olika batterityper finns är det få som har egenskaper som gör dem lämpade för drönaren i projektet, ofta på grund av att deras vikt är för stor för en given kapacitet eller för att deras urladdningsström är för låg. En batterityp som dock har dessa egenskaper och som är vanligt förekommande hos drönare är LiPo batterier. För att uppnå de krav som satts och få en så lång flygtid som möjligt valdes ett batteri med en kapacitet på $3300\,mAh$, urladdningsström på $82,5A$ och vikt på $366\,g$ [13]. Detta ger en uppskattad flygtid på ungefär 9 minuter beroende på flygningens karaktäristik.

Samtliga komponenter som ska sitta på drönaren behöver också en stabil ström-försörjning för att fungera. För att uppnå detta krävs det en konstant spänningnivå som kan regleras. Det finns färdiga lösningar till detta implementerade på diverse integrerade kretsar. Antingen kan en linjär spänningsregulator användas, eller en likspännings-omriktare. Då linjära spänningsregulatorer behöver kylning vid högre laster, som kan bli stor och tung, ska en likspännings-omriktare användas. En mer noggran motivering om valet samt strömförsörjningens dimensionering hittas i Appendix A.2.

2.2 Komponenter för motorstyrning

Som nämndes i avsnitt 1.3 fanns det vid projektets start en flygfärdig drönare. Denna drönare hade en flight controller samt varvtalsregulatorer tillgängliga, vilka båda användes för att styra de nya motorerna. Dessutom fanns det inget behov av att byta ut dem. I detta avsnitt kommer dessa delars funktion på drönaren endast att förklaras i sin korthet.

I projektet används en flight controller av typen APM 2.6 som har två funktioner på drönaren. Den ena är att stabilisera flygingen via data från en accelerometer samt ett gyroskop och den andra är att utifrån styrsignalerna från Raspberry Pi:n styra motorerna så att drönaren flyger som förväntat, mer information om hur detta sker finns i avsnitt 2.5.1.

Vidare används varvtalsregulatorer från DJI i projektet. Modellen som används kan hantera upp till $30\,A$ ström och har en uppdateringsfrekvens på 30 till $450\,Hz$ [14]. Uppdateringsfrekvensen styr hur ofta som varvtalsregulatorn kan ställas in för att ge motorerna en ny rotationshastighet vilket påverkar drönarens stabilitet i luften. Enligt manual kan varvtalsregulatorerna arbeta med en batterispänning på $11,1V$ eller $14,8V$ [14] vilket krävs för att de ska fungera med de valda motorerna och batteriet. Deras funktion på drönaren är att styra motorernas rotationshastighet genom att med en viss frekvens skicka spännings- och strömpulser till motorns ingångar.

2.3 Komponenter för montering

För att montera komponenterna som behövs för att kunna uppfylla delmålen som beskrivs i avsnitt 1.2 behövs en ram och en annordning som kan bära och aktivera släcksprayen. Dessutom behöver fästen för sensorerna utvecklas, se kapitel 3. Valet av ram samt en beskrivning av vilka delar som behövs för en implementation av släckmekanismen redogörs för nedan.

Vid val av ram har dess material betydelse då ramens vikt utgör en stor del av drönarens totalvikt. Dessutom behöver den vara tålig och vibrationsdämpande för att inte störa känslig elektronik samtidigt som den tillåter montering av alla delar som behövs på drönaren. Vanligast är plastmaterial, men även aluminium och kolfiber förekommer. I projektet används en ram av modellen *FlameWheel F450* gjord med fäste för fyra motorer som är av plastmaterial. Valet av denna ram grundar sig främst i att den fanns tillgänglig vid start. Dess diagonala mått är 450 mm [15] vilket ger tillräckligt utrymme för montering av de system och komponenter som projektet behöver. Storleken är en god kompromiss för att möjliggöra tillräckligt stora propellrar, men samtidigt möjliggöra flygning inomhus.

På ramen ska även släckmekanismen fästas. Då eldkällan som drönaren ska kunna släcka är av mindre sort används en släckspray som använder skum som släckmetod. Valet av släckspray motiveras av dess låga vikt i jämförelse med andra brandsläckare [16]. Denna släckspray ska monteras på så sätt att tyngdpunkten på drönaren inte märkbart flyttas samt så att släckningsarbetet kan ske så enkelt som möjligt utan att skada kringliggande elektronik.

För att aktivera släcksprayet används en servomotor då servomotorer är enkla att styra med hjälp av GPIO-pinnar (General Purpose Input Output) samt att det finns små och lätta modeller. I detta projekt används en som väger 9 g och har ett vridmoment på $1,6$ $kgcm$ [17]. Servomotorn måste placeras på ett sådant sätt att den kan applicera en tillräcklig kraft på släcksprayet för att aktivera den.

2.4 Sensorer för datainsamling

För att uppfylla de olika delmålen som beskrivs i problembeskrivningen behöver drönaren kunna känna av sin omgivning. I det här avsnittet diskuteras vilka typer av sensorer som behövs för att delmålen ska kunna nås, hur dessa fungerar och hur de ska användas.

För att uppfylla delmål två behöver drönaren kunna upptäcka väggar och hinder samt avståndet till dessa. Tre olika sätt för att samla in avståndsdata är att använda sig av LIDAR (Light Detection and Ranging), ultraljudssensorer eller IR-sensorer för avståndsmätning. LIDAR använder laser för att mäta avståndet och har både längre räckvidd än ultraljudssensorerna och mindre risk för att störas av propellrarna på drönaren. IR-sensorer för avståndsmätning fungerar likt LIDAR genom att skicka ut ljus och mäter sedan intensiteten av reflektionen för att avgöra avståndet till reflektionspunkten. Som namnet antyder gör IR-sensorer denna mätning med infraröd strålning [18]. Då prissättningen på LIDAR-

sensorer ligger utanför projektets budget och IR-sensorer bedöms ha för stor variation och osäkerhet i mätningar, detta beroende på omständigheter som till exempel reflektionsmaterial [19], kommer ultraljudssensorer att användas för avståndsmätning. Dessa bedöms ha tillräcklig räckvidd, exakthet och framförallt en prissättning som gör att drönaren kan utrustas med flera stycken.

I projektet används ultraljudssensorer av märket *PING))) Ultrasonic Sensor* [20] samt *HC-SR04* [21]. Båda sensorerna fungerar snarlikt och har en praktisk räckvidd på upp till $3,2\,m$. När de aktiveras skickas en ultraljudspuls, sensorn reagerar sedan när den får tillbaka den reflekterade pulsen. Genom att mäta tiden det tar från det att sensorn skickar pulsen tills dess att den får tillbaka pulsen så kan avståndet räknas ut till framförliggande föremål enligt

$$\mathrm{s} = \frac{\mathrm{t} \cdot v_{\mathrm{luft}}}{2}, \tag{2.1}$$

där v_{luft} är ljudets hastighet i luft som uppgår till $343\,m/s$ [22]. Sensorerna mäter med andra ord tiden det tar för ljudet att färdas till ett föremål och tillbaka. Teoretiskt innebär detta att en mätning kan ta upp till $18,5\,ms$ [20]. Därtill har ultraljudssensorerna en mätvinkel på ungefär 15° från centrum beroende på hur målet ser ut [20],[21]. För vissa material på målet i kombination med vissa vinklar på ytan blir mätningarna mindre tillförlitliga. Mätningar mot ytor på över 45° vinkel uppfattas inte alls av sensorerna.

Utöver avståndsmätning måste också drönarens hastighet kunna mätas för att delmål två ska kunna uppfyllas. I projektet görs detta genom att använda en *PX4Flow Optical Flow Sensor* [23], fortsättningsvis benämnd som optisk flödessensor. Sensorn monteras direkt på drönaren och kräver inga externa sändare eller mottagare och den fungerar även inomhus, till skillnad från till exempel GPS.

Flödessensorn fungerar genom att ta en serie bilder och jämföra positionen av pixelformationer mellan två efterföljande bilder. Flödessensorn är även utrustad med en ultraljudssensor som mäter höjden ovanför marken. Med hjälp av skillnaden på bilderna och höjden från marken räknar sensorn ut en ungefärlig hastighet [24]. För att detta ska fungera bra krävs dock att drönaren flyger över en väl upplyst och texturerad yta, lik den i Figur 3.7.

Vidare behöver det finnas en sensor som både kan upptäcka en eldkälla samt skilja den från andra värmekällor. Detta för att kunna uppfylla delmål fyra som handlar om identifiering av en eldkälla. Identifieringen görs med hjälp av en värmekamera av modellen *Adafruit 3622 Thermal Camera*. Kameran kan ge tillförlitliga temperaturvärden i intervallet 0°C till 80°C med en spridning på ± 2,5°C.

Därtill har värmekameran en upplösning på 8x8 värmevärden med en uppdateringsfrekvens på $10\,Hz$ [25]. Dessutom har den en ungefärlig bildvinkel på 30°, vilket gör att den på avståndet $1,5\,m$ kan upptäcka värme på $0,75\,m$ distans ifrån kamerans centrum.

Värmekameror fungerar genom att mängden infraröd strålning som når sensorn ändrar den elektriska resistansen hos sensorn och därigenom kan en temperatur räknas ut [26]. Då värmekameran som används i projektet har en relativt låg upplösning innebär det att

värmevärdet för en pixel kommer vara medeltemperaturen som kameran uppfattar för den pixeln. Medeltemperturen mäts dock inte alltid i pixelns centrum [27].

Vid testning av värme- och eldidentifiering, se avsnitt 4.3, används även en kamera av modellen Raspberry Pi Camera Board v1.3 (5MP, 1080p) med upplösningen 640x480 [28]. Kameran används som ett komplement till värmekameran för att kunna se om en värmekälla som värmekameran upptäckt är en eld eller bara något varmt föremål.

2.5 Kommunikation

Då prototypen består av tre olika datorsystem, bestående av en laptop, en Raspberry Pi och en flight controller, behövs viss kommunikation mellan dessa. Det här avsnittet presenterar de styrsignaler flight controllern kräver för att styra drönaren samt de olika kommunikationskanalerna som behöver etableras mellan de olika datorsystemen.

2.5.1 Drönarens kontrollsignaler

För att flyga drönaren används fyra olika operationer [29]:

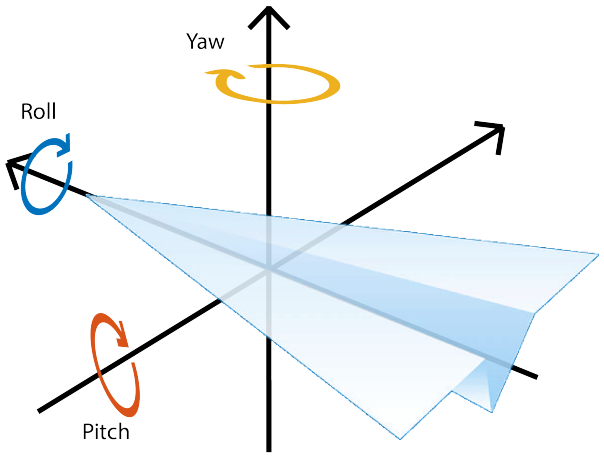
- *Throttle* påverkar lyftkraften genom att ett värde på 0% *throttle* resulterar i noll lyftkraft medan 100% ger maximal lyftkraft.

- *Roll* påverkar lutningen i sidled, vilket gör att drönaren accelererar rakt åt sidan. Ett värde på mindre än 50% *roll* får drönaren att accelerera åt vänster, medan ett värde på mer än 50% accelererar den åt höger.

- *Pitch* påverkar lutningen framåt eller bakåt, vilket gör att drönaren accelererar i någon av de två riktningarna. Om *pitch* är mindre än 50% accelererar drönaren framåt, medan om *pitch* istället är mer än 50% accelererar den bakåt.

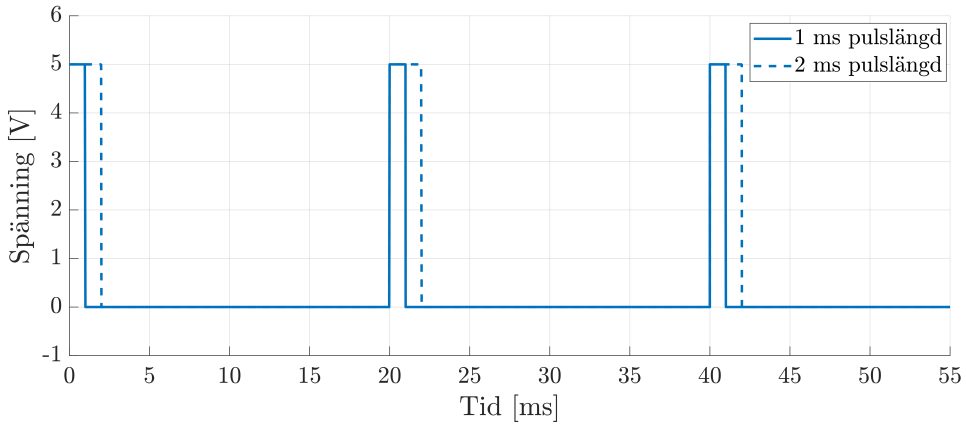
- *Yaw* styr rotationen kring z-axeln. Ett värde på mindre än 50% får drönaren att rotera motsols, medan ett värde på mer än 50% roterar den medsols.

Rörelserna *roll*, *pitch* och *yaw* motsvarar alltså en rotation kring en av kordinataxlarna. I Figur 2.2 framgår vilka styrsignaler som påverkar vilka rotationer.



Figur 2.2: *Drönarens olika rörelseoperationer, där yaw får drönaren att rotera åt höger eller vänster, roll justerar lutningen i sidled och pitch styr lutningen framåt och bakåt.*

Drönaren ska kontrolleras med hjälp av styrsignaler från Raspberry Pi. Dessa styrsignaler behöver levereras parallellt och kontinuerligt till flight controllern för att bibehålla styrning av drönaren. Drönaren kan kontrolleras med en RC-kontroll [30] vilket betyder att drönaren styrs av elektriska signaler med varierande pulslängd, även kallat PWM-signaler (Pulse-Width Modulation). Detta betyder att varje rörelseoperation får varsin pulssignal med frekvensen 50 Hz och där varje pulssignals pulsbredd ligger mellan 1 ms och 2 ms. Pulsbredden motsvarar hur mycket av varje styroperation som ska ske, där 1 ms motsvarar 0% och 2 ms motsvarar 100% av styroperationens maximala värde. Dessa signalers form framgår i Figur 2.3.



Figur 2.3: *Drönarens kontrollsignaler. Den heldragna linjen har en pulsbredd på 1 ms, vilket motsvarar 0% av en operation. Det streckade linjen har i kontrast en pulsbredd på 2ms, vilket motsvarar 100% av en operation.*

Förutom nämnda styrsignaler styrs även servomotorn, som aktiverar släckmekanismen, av en PWM-signal [17]. En mätning av servomotorns funktion visar att en pulslängd på 1 ms motsvarar en grundposition och en pulslängd på 2 ms motsvarar en rotation cirka 90° moturs.

2.5.2 Nätverkskommunikation

För att kommunicera mellan den på drönaren monterade Raspberry Pi:n och det datorsystem där det autonoma styrsystemet exekveras krävs ett kommunikationsprotokoll som fungerar utan kablar och som ger en hög prestanda, då det stundatals behöver skickas mycket data. WiFi uppfyller båda dessa krav. Ett annat alternativet är att använda sig av Bluetooth som också finns inbyggt på Raspberry Pi. Fördelen med WiFi är en teoretiskt högre hastighet för *WiFi 802.11n* [31] än för *Bluetooth 4.1* [32] som den använda Raspberry Pi modellen är utrustad med [33]. Till följd av detta används WiFi i projektet.

För varje sensorgrupp, det vill säga ultraljudssensorer, värmekamera respektive optisk flödessensor, skickas en egen dataström till datorsystemet i samma takt som datan blir tillgänglig på Raspberry Pi:n. För samtlig data används UDP (User Datagram Protocol) som protokoll då datan är av realtidskaraktär och datamängden är relativt liten per mätning.

2.6 Reglersystem

En stabil flyghöjd är viktig för att drönaren ska kunna navigera på ett säkert sätt. Därför utvecklas en höjdreglering där höjden mäts med hjälp av flödessensorns inbyggda ultraljudssensor och regleras digitalt på Raspberry Pi. Höjdregleringen ska på ett effektivt sätt kunna få upp drönaren på rätt höjd, utan risk för icke kontrollerat beteende, och därefter kunna hålla denna höjd utan för stora avvikelser.

För att undvika positionsdrift är även hastigheten i horisontell led viktig att kunna reglera. Positionsdrift är negativt dels eftersom drönaren behöver kunna hållas på plats vi släckning och dels eftersom det autonoma styrsystemet ska ha full kontroll över flygningen under avsökningen. Då drönaren inte använder sig av GPS-positionering används enbart den optiska flödessensorns hastighetsdata till regleringen, som även denna utförs digitalt på Raspberry Pi.

Det finns väldefinierade metoder för reglering av mekaniska system. Dessa metoder innefattar negativ återkoppling samt en regulatorfunktion. En mycket vanlig regulatorfunktion är PID-regulatorn [34] som visas i Ekvation 2.2.

$$F(s) = K_p + \frac{K_i}{s} + \frac{K_d}{1 + T_f s} \tag{2.2}$$

K_p, K_i, K_d och T_f är parametrar som kan bestämmas för önskat systembeteende. Den finns flera metoder för att hitta rimliga värden på dessa parametrar [34]. I detta projekt används dels Routh-Hurwitz stabilitetskriterium som beskrivs i Appendix A.3, samt undersökning av stegsvar och nivåer på genererade styrsignaler för att hitta passande parametrar. Ekvation 2.2 kan delas upp i tre termer:

- P-verkan motsvaras av den första termen i regulatorfunktionen och förstärker det nuvarande reglerfelet. En högre P-verkan ger en snabbare system till priset av större styrsignaler samt minskade stabilitetsmarginaler. [34]

- I-verkan representeras av den andra termen i regulatorfunktionen och integrerar upp reglerfelet över tiden. Detta betyder att processstörningar är bättre kompenserade och kvarvarande fel kan försvinna. Högre I-verkan ger på samma sätt som P-verkan ett snabbare system men med större styrsignaler samt mindre stabilitetsmarginaler. [34]

- D-verkan motsvaras av den tredje termen i regulatorfunktionen och reagerar på skillnader på reglerfelet i tiden. Detta betyder att en större D-verkan kan trycka ner snabba förändringar, vilket medför bättre stabilitetsmarginaler [34], men samtidigt att systemet blir mer känsligt för felaktiga mätningar.

2.7 Artificiell intelligens

För att uppnå autonom flygning krävs ett system som ersätter den mänskliga piloten. Systemet ska under flygning kunna tolka drönarens fysiska omgivning och i realtid översätta detta till beslut om styrning. Systemet måste vara robust och dynamiskt i den mån att alla möjliga kombinationer av omgivningen resulterar i ett relevant beslut. För att uppnå dessa krav krävs ett intelligent system med tolerans för varierande data och möjlighet att optimera efter ett abstrakt mål.

2.7.1 AirSim

Det intelligenta styrsystemet kräver kontinuerliga tester under utveckling. Tillgång till en simulerad miljö är mycket fördelaktigt i denna process då risken för kollisioner eller andra fysiska begränsningar som flygtid inte behöver tas i beaktande. Detta innebär att den artificiella intelligensen kan utvecklas, testas och optimeras i en miljö där försiktighetsåtgärder inte är nödvändiga, vilket säkerhetsställer att styrningen är så komplett som möjligt innan den anpassas till den fysiska drönaren.

AirSim[35] är en fotorealistisk simulator för fordon, se Figur 2.4, utvecklad av Microsoft som en open-source plugin för Unreal Engine [36]. Källkoden är skriven i C++, men simulatorn erbjuder API:er (Application Programming Interface) för programmeringsinteraktion med fordonen genom *remote procedure calls*, en metod där ett datorprogram exekverar en subrutin i ett annat adressutrymme utan att programmeraren behöver koda detaljerna för hur mottagaren ska interagera, vilket innebär att program för fordonsstyrning kan skrivas i godtyckligt programspråk. AirSim simulerar med hjälp av en emulerad flight controller och Unreal Engines fysikmotor ett verklighetstroget beteende hos drönaren. Den har även ett inspelningsläge där data registrerad av drönaren kan sparas ned i textfiler med korta, jämna mellanrum.

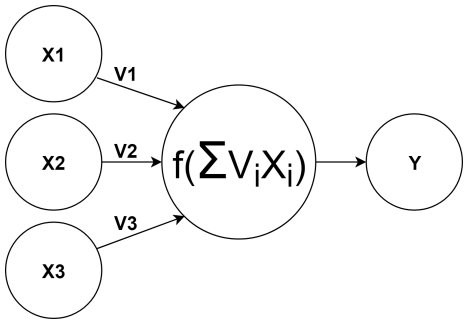
Figur 2.4: *Fotorealistisk simulerad omgivning i AirSim. Från [37] Copyright (c) Microsoft Corporation. All rights reserved. MIT License.*

2.7.2 Neurala nätverk

Artificiella neurala nätverk är inlärningsalgoritmer som imiterar funktionen hos de biologiska neurala nätverk som levande hjärnor utgör. Dessa system ”lär sig” uppgifter progressivt genom mänsklig definition av fördelaktiga och ofördelaktiga beteenden hos nätverket. Nätverk är därför mycket användbara för uppgifter där målet är svårt att definiera men där det är lätt att skilja på bra och dåliga resultat. Genom användning av neurala nätverk sätts ingen övre gräns på vilken komplexitet systemet kan uppnå eller vilken typ av data som går att processera. Med enbart en konfiguration av ultraljudssensorer tillgänglig skulle en kombination av hårdkodad kollisionsundvikning och en simpel avsökningsalgoritm potentiellt kunna uppnå ett beteende som uppfyller delmål tre. Detta sätter dock en gräns på hur nyanserat och utökningsbart systemet skulle bli, då önskade beteendeförändringar hos systemet eller variationer i drönarens miljö skulle kräva en stor manuell insats eftersom varje önskat resultat från varje variation av indatan måste definieras i programkod. Genom användning av neurala nätverk lämnas fler möjligheter öppna och framtida modifikationer till nya miljöer, domäner eller uppgifter behöver inte betyda att systemet måste omstruktureras. Då det i dagsläget dessutom är relativt simpelt att implementera neurala nätverk med hjälp av maskininlärningsramverk på hög nivå förflyttas en stor del av implementationsarbetet från programimplementation till datainsamling vilket är fördelaktigt då AirSim erbjuder stora möjligheter att samla in välstrukturerad och kvalitativ data.

Ett neuralt nätverk består av sammanbundna uträkningsenheter, neuroner, med tillhörande modifierbara värden. Sammanbindningarna har så kallade vikter, vilka avgör signalstyrkan mellan neuroner, se exempel i Figur 2.5. Genom kontinuerlig justering av dessa kan systemet progressivt bli bättre på sin uppgift, det vill säga att en viss insignal ska resultera i en önskad utsignal. Denna justering görs iterativt genom att mata nätverket med indata, värdera resultatet och sedan justera vikterna. Det generella målet med användning av dessa modeller är att de ska kunna generalisera över varierande, men liknande, data och således kunna användas för till exempel beslutstagande i situationer de inte tidigare sett

[38], [39].



Figur 2.5: *Beräkning av signalstyrka till en neuron Y från sammankopplade neuroner X.*

2.7.3 Djupinlärning

Framsteg både inom hårdvara, som till exempel beräkningskraftiga GPU:s (Graphical Processing Units), och mjukvara, däribland verklighetstrogna simulatorer, har lett till en strategi för modellering av neurala nätverk som, olikt traditionell maskininlärning, inte baseras på ”feature engineering”, se Appendix A.4. Strategin bygger istället på att nyttja den potential som omfattande mängder data innebär. I och med möjligheten att genom AirSim snabbt samla in stora mängder skräddarsydd data är denna djupinlärningsmetod ett bra tillvägagångssätt för att hos drönaren uppnå ett avancerat autonomt beteende som dessutom är lätt att justera genom att konfigurera datan. När djupinlärning används justeras vikterna i ett neuralt nätverk för att så mycket som möjligt efterlikna beteendet som representeras av datan. Därmed blir kvaliteten hos datan en nyckelfaktor. Funktionaliteten liknar mänsklig inlärning i den mening att tidigare observationer används för att skapa ett beslutssystem som direkt processerar indata och genererar ett resultat, utan att använda sig av fördefinierade mellansteg [40].

2.7.4 Träningsdata

Neurala nätverk tränade med djupinlärning blir aldrig bättre än datan de har tränats på. De övergripande egenskaper och tendenser som träningsdatan innehar förs över till nätverket under träningsfasen då nätverket iterativt ändras just för att anpassa sig till datan in i detalj. För att datan ska fylla det önskade syftet krävs att flera faktorer överensstämmer med det utsatta målet. Dels bör mängden data spegla komplexiteten av systemet det modellerar och dels måste variationen av datan vara tillräcklig för att översiktligt täcka de situationer som kan uppstå. Nätverkets generaliseringskunskaper täcker sedan varierande detaljer. [41]

2.7.5 Nätverksdesign

Beroende på hur neurala nätverk konstrueras lämpar de sig olika bra för olika uppgifter och ändamål. Uppgiften i kombination med typ och mängd tillgänglig träningsdata är stora

faktorer i beslut om nätverksdesign, men det finns inga definitiva regler om vilka nätverk som är bäst på en uppgift. Detta då det ofta finns situationsspecifika omständigheter som kan påverka. Genom att använda rätt struktur och design för en uppgift kan effektiviteten hos nätverket ökas och de viktigaste egenskaperna från träningsdatan speglas. Således maximeras nätverkets möjlighet att lösa sina uppgifter.

För att kunna modellera de generella egenskaperna som finns i träningsdatan och sedan applicera det i godtycklig miljö krävs det att storleken och komplexiteten hos nätverket sätts till en lagom nivå. Om nätverket har för få justerbara parametrar finns risken att nätverket inte blir anpassningsbart nog för situationen, benämns underanpassning, och således inte speciellt användbart. Motsatsen är förstås att föredra, men om nätverket har allt för många ändringsbara parametrar finns risken att nätverket blir en alldeles för bra modell av den tillgängliga datan och på grund av detta tappar sin möjlighet till att generalisera till liknande situationer. Detta fenomen kallas för överanpassning [42]. Vilken nivå som är den optimala varierar beroende på applikation, om det önskas mycket eller lite generalisering, bredden på appliceringsområdet samt komplexiteten på både uppgiften och träningsdatan. Det krävs empiriska tester för att hitta den nivå som fungerar och om några omständigheter ändras kan även nätverkets effektivitet komma att ändras och dess design behöva omevalueras. [41]

Kapitel 3

Implementation och utveckling

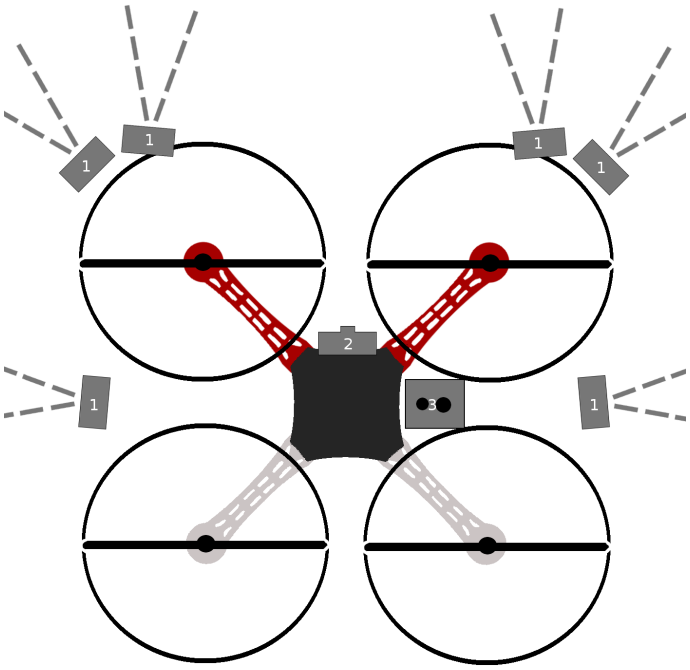
Drönarprototypen konstruerades genom att först implementera och testa de olika delsystemen separat. Därefter monterades och testades systemen tillsammans på drönaren och justeringar gjordes för att integrera systemet. I det här kapitlet redovisas hur de komponenter som nämndes i föregående kapitel använts för att implementera dessa delsystem och därmed realisera prototypen. Källkoden till systemen finns att hitta på projektets git-repo, se Appendix A.5.

3.1 Sensorer

Avsnittet nedan behandlar hur de olika sensorerna, som beskrivs i avsnitt 2.4 i kapitlet om komponenter och system, användes för att kunna bedöma avstånd och drönarens hastighet samt identifiera eld. Dessutom ges en beskrivning av var sensorerna placerades och varför.

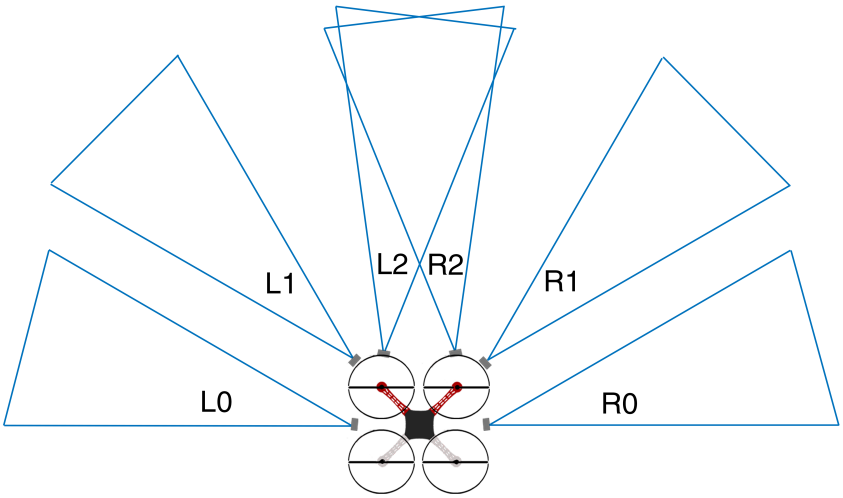
3.1.1 Bedömning av avstånd till föremål

För att det autonoma styrsystemet skulle fungera behövde det avståndsdata till hinder för en halvcirkel i drönarens färdriktning, framförallt rakt fram. Till följd av ultraljudssensorernas mätvinkel, se avsnitt 2.4, kunde inte hela synfältet täckas och därför placerades en sensor åt höger, en åt vänster samt fyra stycken sensorer framåt i olika vinklar, se Figur 3.1.



Figur 3.1: *Översiktsbild över placeringen av alla sensorer samt kameran på drönaren. 1 är ultraljudssensorerna, 2 är värmekameran och 3 är den optiska flödessensorn, riktad nedåt. De streckade linjerna visar respektive sensors synfält.*

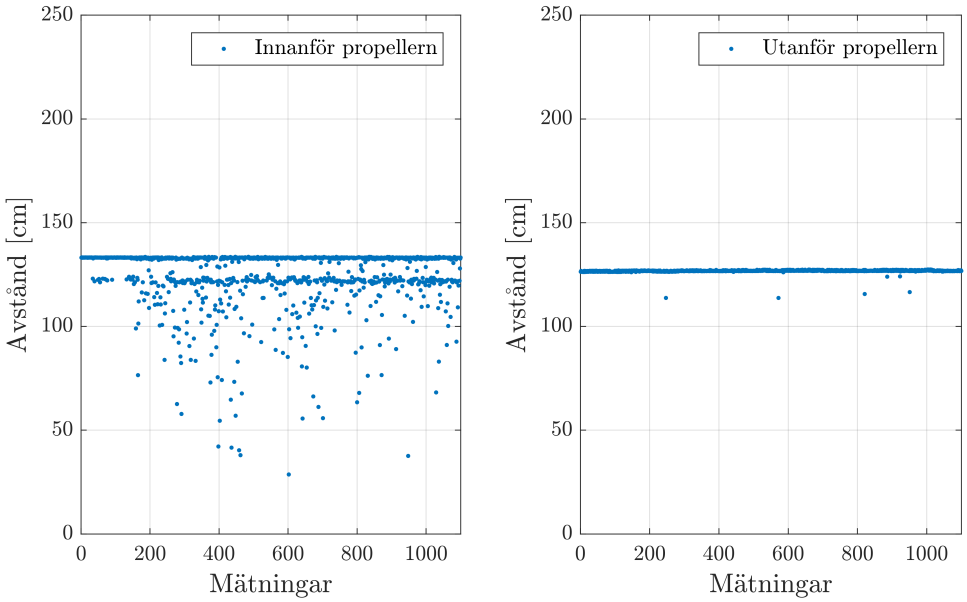
De fyra ultraljudssensorerna framtill på drönaren placerades längst ut på ett par kolfiberstänger så att de hamnade precis utanför propellrarnas luftflöde, se Figur 3.1. För att fästa sensorerna vidareutvecklades en design av Dennisbaldwin [43] anpassad för 3D-printers eftersom den ger stor frihet vid design. De inre, framåt riktade sensorerna vinklades 7° inåt medan de yttre vinklades rakt ut i samma riktning som armen. Sensorerna på sidorna vinklades 15° framåt och alla sensorer fästes med dämpande kuddar för att minska vibrationsstörningar. Synfältet för drönaren kan ses i Figur 3.2 och ett föremål i drönarens storlek syns alltid inom minst $1,5\,m$ avstånd från drönaren.



Figur 3.2: *Bild över ultraljudsensorernas synfält, där drönaren är något förstorad för tydlighet. Sensorerna är namngivna L0, L1, L2, R2, R1, R0 från vänster till höger i bild.*

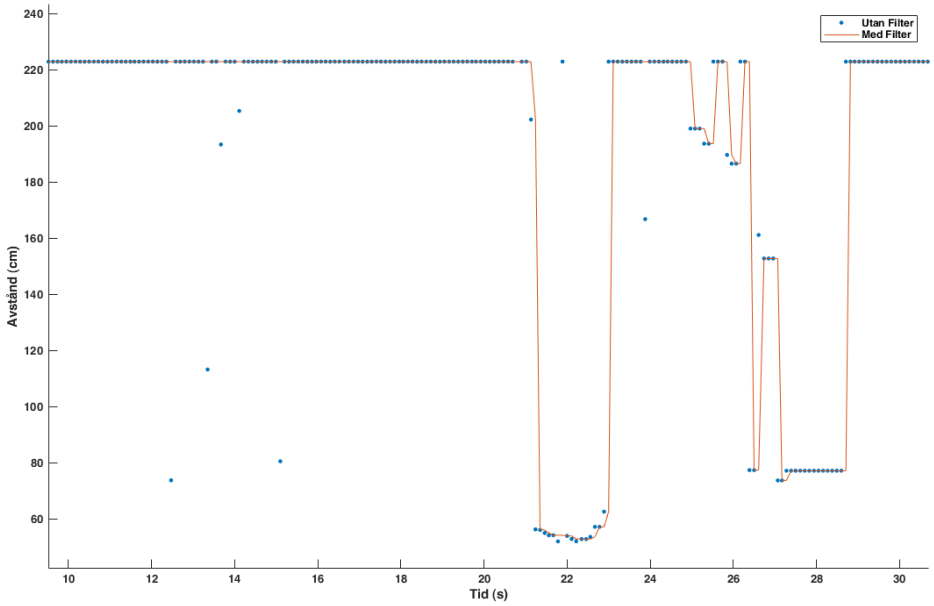
Emellertid var denna placering av sensorer inte den första som gjordes. I ursprungsplaceringen satt sensorerna direkt på ändarna av drönarens armar och därmed även under propellrarna, men test visade att ultraljudssensorerna inte gav tillförlitlig data när de satt där, se Figur 3.3. Anledningen till varför sensorerna ursprungligen fästes på drönarens armar var att störningen från propellrarnas luftflöde uppskattades vara så pass liten att det skulle gå att kompensera för den med hjälp av programkod samt att vikten skulle öka med en ställning liknande den som i slutändan ändå gjordes.

När sensorerna istället placerades längre ut så sågs minskade störningar på grund av propellrarnas luftflöde och därmed bättre mätdata genom att ett färre antal felaktiga värden genererades av sensorerna. De felaktiga värdena kunde lura drönaren att den var för nära ett hinder när den egentligen var långt borta eller tvärt om.

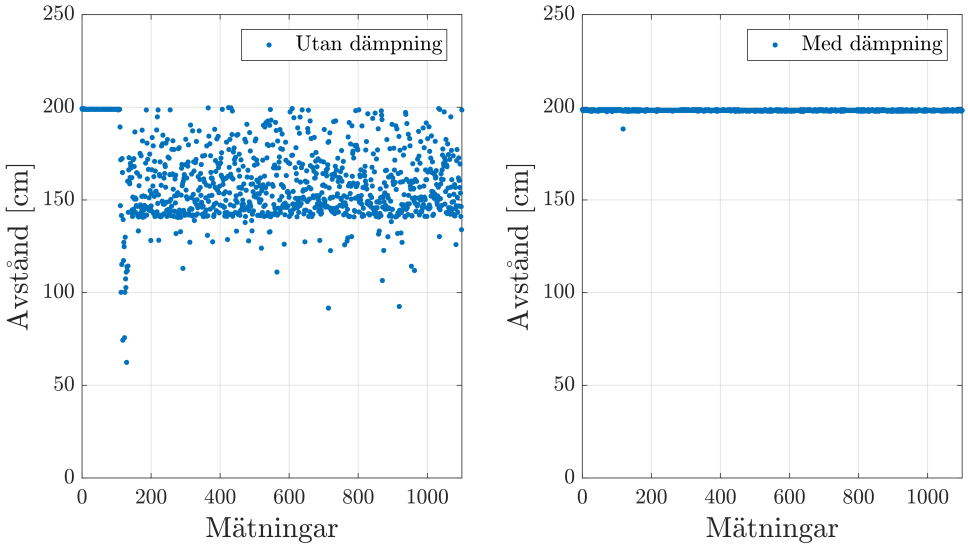


Figur 3.3: *Test med en ultraljudssensor placerad under en av drönarens propellrar respektive test med sensorn placerad utanför propellerns räckvidd. Målet var en vägg på 130 cm avstånd från sensorn. Ultraljudssensorn var inte monterad på drönaren vid testen.*

Vid test av ultraljudsesnorerna monterade på drönaren syntes ytterligare störningar. De berodde i huvudsak på vibrationer i ramen som orsakades av motorerna när de var igång. Genom att placera en bit skumplast mellan ramen och motorerna gick det att minimera dessa störningar, se Figur 3.5. Skumplasten ledde dock att motorerna fick svårare att rotera och fick därmed tas bort. För att ändå minska störningarna infördes ett medianfilter på mätningarna från ultraljudssensorerna. Filtret tar medianen av de tre senaste värdena och hjälper till att motverka spikar i mätningarna, i utbyte mot en fördröjning på en mätning vid faktiska förändringar i avstånd, se Figur 3.4.



Figur 3.4: *Ultraljudsmätningar under flygning med och utan medianfilter. Notera hur enskilda avvikande värden filtreras bort och hur den filtrerade datan ändras något efter att den verkliga datan gör det.*



Figur 3.5: *Mätningar av ultraljud före och efter installation av dämpande skumplast. De första 100 mätningarna utan motorerna på, de nästa 1000 med motorerna igång. Under mätningarna var drönaren fastmonterad i en ställning.*

Då alla sensorerna använder ultraljud fanns det risk för störningar mellan sensorerna om de körs parallellt. Istället kördes sensorerna i serie vilket, tillsammans med vinkeln mellan sensorerna, innebär att risken för interferens minskar. Dock medför detta att varje komplett

avståndsmätning kan ta upp till 6 gånger längre tid än vad en individuell mätning tar. Trots den teoretiska maximala tiden på $18,5$ ms per sensor uppmättes det att det praktiskt kan ta upp till 127 ms för en komplett mätning. Detta beror på att koden för att sätta upp sensorerna tar tid att köra samt att den verkliga maxtiden för en mätning är något högre än den teoretiska.

3.1.2 Bedömning av drönarens hastighet

Med syfte att mäta drönarens hastighet användes en optisk flödessensor som placerades på sidan av drönaren med fri sikt ner mot marken, se Figur 3.1. Det viktiga var att både kameran och den monterade ultraljudssensorn kunde ta mätningar utan störningar.

Då mätserien visade en stor varians i form av brus användes ett lågpass-filter för att få bort högfrekventa komponenter i mätserien. Eftersom drönaren har en begränsad acceleration kan inte hastigheten ändras för mycket mellan varje sampling, vilket betyder att de höga frekvenserna innehåller obetydlig information i form av mätfel och brus. För att dämpa så mycket oönskat brus som möjligt, men samtidigt hålla filtrets komplexitet låg, användes ett butterworth-filter av andra graden med dämpning ett. Den analoga överföringsfunktionen $H(s)$ i Laplace-planet kan beskrivas enligt

$$H(s) = \frac{\omega_0^2}{s^2 + 2\omega_0 s + \omega_0^2}, \tag{3.1}$$

där ω_0 är filtrets gränsfrekvens.

Med hjälp av bilinjär transform togs den diskreta överföringsfunktionen $H(z)$ i Z-planet fram, vilken beskrivs enligt

$$H(z) = \frac{b_2 z^{-2} + b_1 z^{-1} + b_0}{a_2 z^{-2} + a_1 z^{-1} + a_0} \tag{3.2}$$

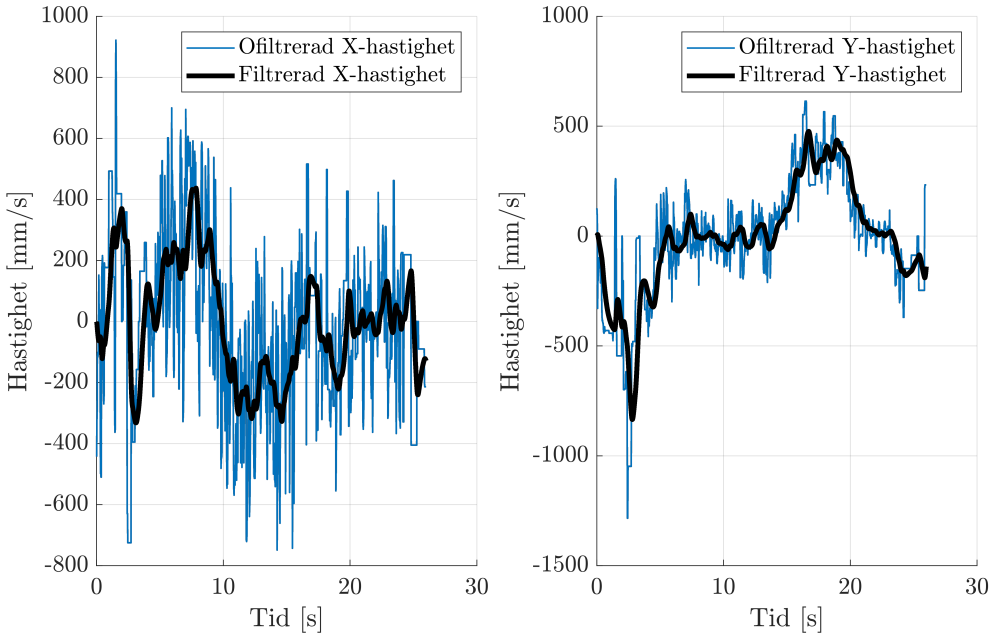
$$\begin{aligned} a0 &= 4 + 4\omega_0 T + (\omega_0 T)^2 \\ a1 &= 2(\omega_0 T)^2 - 8 \\ a2 &= 4 - 4\omega_0 T + (\omega_0 T)^2 \\ b0 &= (\omega_0 T)^2 \\ b1 &= 2(\omega_0 T)^2 \\ b2 &= (\omega_0 T)^2, \end{aligned}$$

där T är samplingsintervallet i sekunder.

Denna överföringsfunktion kunde sedan inverstranformeras till differensekvationen som uttrycks i Ekvation 3.3 nedan.

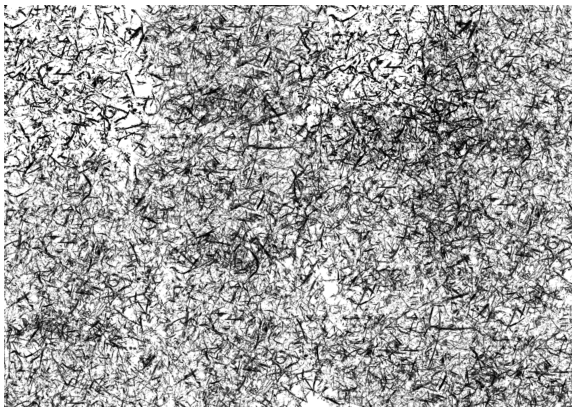
$$y[n] = (b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2])/a_0 \tag{3.3}$$

Resultatet av ett test av flödessensorn går att se i Figur 3.6. Testet innefattade rörelse i både x- och y-led där sensorn bars fram och tillbaka i ett rum.



Figur 3.6: *Verklig och filtrerad hastighetsdata för x- och y-led vid ett test. Den svarta linjen visar på en tydlig förbättring av hastighetsdatans mätvärden efter filtrering.*

Ytterligare en åtgärd togs för att säkerställa bra sensormätningar. Som nämndes i avsnitt 2.4 fungerar flödessensorn med hjälp av en kamera som jämför rörelsen av pixelformationer mellan samplingar. För att få detta att fungera så bra som möjligt behöver kameran kunna uppfatta tydliga mönster att jämföra mellan. Då det inte går att garantera detta för alla ytor där drönaren testades användes ett utskrivet detaljerat mönster i svartvitt, som visas i Figur 3.7, under drönaren för att ge tydliga mönster även vid dåliga ljusförhållanden. Detta garanterade att flödessensorn kunde ge tillförlitlig data, även när underlaget gjorde att kvaliteten på mätningarna blev för dålig.



Figur 3.7: *Det mönster som används som underlag för att förbättra hastighetsbedömningen från flödessensorn vid flygning under ljussvaga förhållanden.*

3.1.3 Detektera värmekälla och identifiera eldkälla

Som nämnts i avsnitt 2.4 behövde en värmekamera användas till värmedetektering och eldidentifiering. Värmekameran fästes framåt på drönaren för att därigenom kunna upptäcka när drönaren närmade sig något varmt, se Figur 3.1. Från början var tanken att även en vanlig kamera, inkopplad till Raspberry Pi:n, skulle behöva användas. Det visade sig dock efter testning, se avsnitt 4.3, att det räckte med värmekameran eftersom den inte bara kunde generera värmedata i ett rutnät, utan också användas som ett koordinatsystem för att hitta varmaste punkten. Temperaturangivelsens placering i rutnätet står med andra ord i direkt relation till var i kamerans synfält som temperaturen uppmättes.

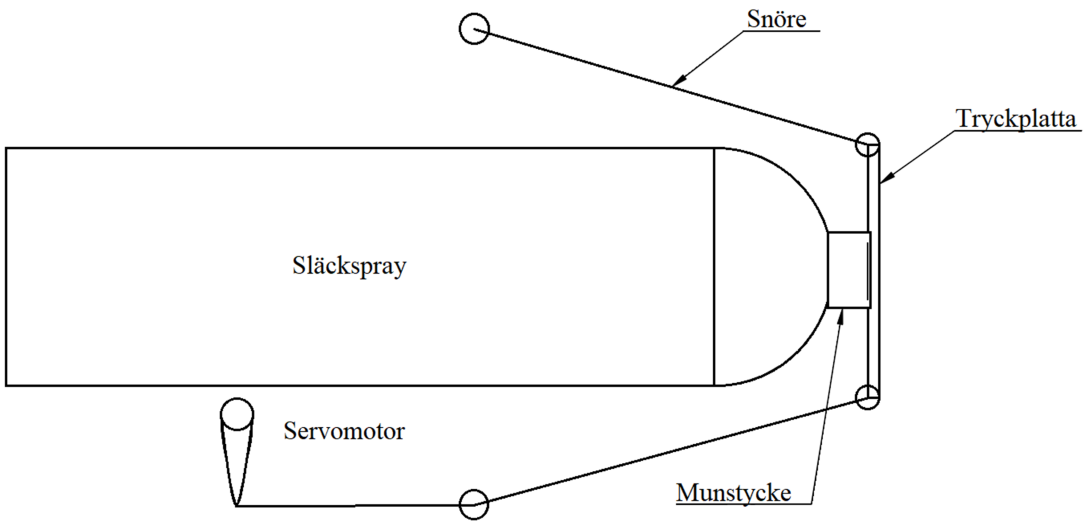
Utöver detta visade testet att temperaturvärdena som skrivs ut i rutnätet kan vara mycket lägre än det verkliga temperaturvärdet, men det går ändå utan problem urskilja var i koordinatsystemet som det finns en varm punkt, om det finns någon. Detta eftersom värdet ändå är högre än temperaturangivelserna runt omkring. Ju närmare värmekällan drönaren rör sig, desto högre blir värdet i den varma punkten i koordinatsystemet och skiljer sig allt mer från de närliggande temperaturvärdena. Anledningen till varför de uppmätta temperaturerna var lägre än de teoretiska berodde på att värmekameran ger medeltemperaturen för en specifik pixel, se avsnitt 2.4 för vidare förklaring.

Något som i början dock antogs vara ett problem var hur värmekameran skulle kunna skilja på en eld och exempelvis ett varmt element eller en människa. Detta var anledningen till varför en kamera troddes behövas. Testning, se avsnitt 4.3, visade emellertid att den uppmätta temperaturen hos en människa (även på nära håll) samt ett varmt element inte gav mycket högre värden än intilliggande, vilket gjorde att en eld gick skilja från andra värmekällor utan att använda sig av bildbehandling.

3.2 Släckmekanism

Släckmekanismen består av tre delar; släckspray, servomotor samt snören som överför servomotorns vridmoment till en kraft på släcksprayens munstycke. Släcksprayen sattes fast med buntband mot undersidan av drönaren. Detta gjordes eftersom sprayens munstycke skulle vara så långt som möjligt från den elektronik som sitter på drönare. En annan anledning till detta är eftersom sprayen har en relativt stor massa som hade medfört ett stort moment på drönaren vid placering utanför centrum. Därtill fanns mycket fritt utrymme på ramens undersida.

För att kunna aktivera släcksprayen behövdes en anordning som kan ta vara på servomotorns vridmoment för att trycka på munstycket. Detta gjordes genom att en platta limmades på munstycket för att få en mindre vinkel mellan snöret och munstycket och på så sätt få en större applicerad kraft. Den ena sidan på plattan fästes med ett snöre till drönarens ena ben. Den andra sidan fästes med ett annat snöre, genom ett av drönarens ben på motsatt sida, till en arm på servomotorn. Detta framgår i Figur 3.8. Kraften som är möjlig att applicera på munstycket beräknas i Appendix A.6 till cirka $4,3\,N$.



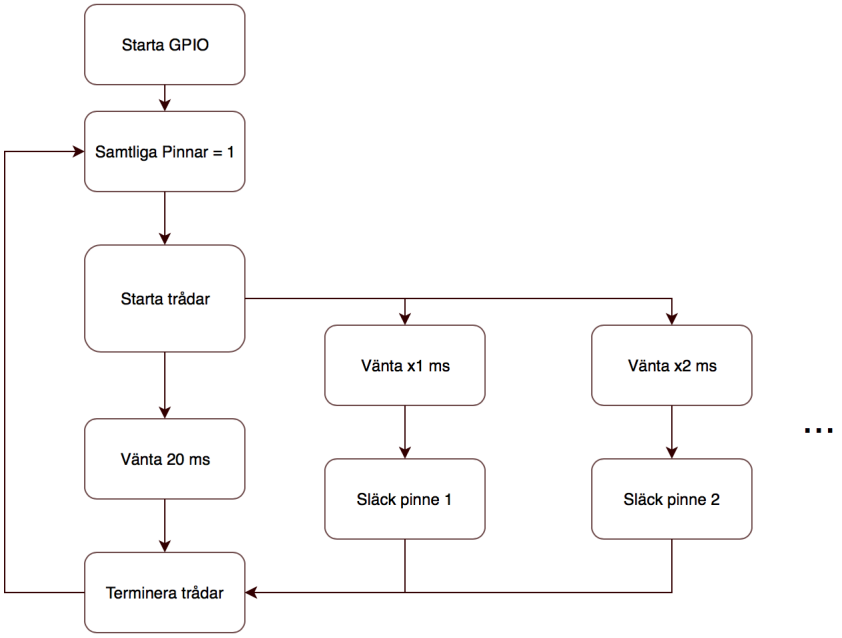
Figur 3.8: *Ritning över släckmekanismen. En servomotor drar i ett snöre så att en tryckplatta applicerar kraft på munstycket och aktiverar därmed släcksprayet.*

3.3 Kommunikation mellan datorsystem

Implementationen av de olika kommunikationskanalerna skedde i C och Python. I följande avsnitt finns detaljer om implementationen av kommunikationen samt de avvägningar som gjordes.

Kommunikationen med drönaren sker genom att en GPIO-pinne på Raspberry Pi:n kopplas till varje input på flight controllern på samma sätt som en RC-mottagare kopplas in [30]. Genom att tända och släcka GPIO-pinnarna kan PWM-signaler fås fram på det sätt som drönaren kräver, vilket framgår i kapitel 2.2. På så sätt kan drönaren styras enbart med hjälp av Raspberry Pi:n.

För att skriva till GPIO-pinnarna på Raspberry Pi:n används ett C-bibliotek som heter pigpio [44]. Ett problem med denna kommunikationstyp är att varje GPIO-pinne behöver vara hög olika länge, vilket kan lösas genom att varje GPIO-pinne får sin egna tråd. Alla pinnarna sätts höga från en huvudtråd, därefter startas alla trådar för varje pinne där en startas med en fördröjning med rätt längd med hjälp av *usleep* [45]. När en fördröjning är slut släcks GPIO-pinnen. Under denna tid är huvudtråden låst i en egen sleep-funktion som motsvarar pulssignalens periodtid. När denna är slut termineras samtliga GPIO-trådar och alla GPIO-pinnar sätts höga igen. Algoritmen framgår i Figur 3.9.



Figur 3.9: *Flödesschema över kommunikationsalgoritmen för drönarens styrsignaler. Med hjälp av separata trådar kunde PWM-signaler med individuella pulslängder skickas parallellt och kontinuerligt till drönarens Flight Controller.*

Vid uppmätning av utsignalerna märktes att pulslängden inte blev konstant. Vid ett test med hjälp av servomotorn märktes att variationen i pulslängderna gjorde att servomotorn inte var helt still, utan ryckte märkbart. Detta beror troligtvis på schemaläggningen hos trådarna i Raspberry Pi:s operativsystem. I stället testades därför att använda pigpio:s inbyggda funktion för servostyrning. Med hjälp av denna funktion kan PWM-signaler startas som sedan skickas parallellt och kontinuerligt till de GPIO-pinnar man väljer [44].

När dessa signaler testades märktes att pulsbredden var mycket mer konstant. Signalerna testades på samma sätt, det vill säga med hjälp av servomotorn och ett oscilloskop. Det gick inte att märka några ryckningar i servomotorn och inga avvikelser syntes på oscilloskopsbilden.

Vidare krävdes kommunikation mellan Raspberry Pi:n och datorsystemet, vilka kommunicerade med varandra över WiFi. Kommunikationen mellan Raspberry Pi och datorsystemet kunde enkelt delas upp i två delar. En del som samlade ihop data på Raspberry Pi:n och skickade den vidare till datorsystemet och en annan del som skickade tillbaka de uträknade styrsignalerna till Raspberry Pi från datorsystemet.

För att åstadkomma kommunikation från Raspberry Pi:n behövdes tre par av trådar, bestående av tre mottagartrådar och tre sändartrådar och där varje par hade en egen socket. Detta ledde till att kommunikationen kunde hållas helt asynkron och den insamlade sensordatan kunde skickas så snart den fanns tillgänglig. Då de olika sensorerna har olika uppdateringsfrekvens var det lämpligt att hålla hanteringen av datan separerad.

Kommunikationen tillbaka till Raspberry Pi gjordes så enkel som möjligt. En UDP socket skickade över 5 bytes av information vid varje uppdatering från det autonoma styrsystemet. Datan togs sedan emot och omvandlades till lämpliga värden för att styra drönarens höjd, *yaw*, *pitch* och *roll*. Den sista byten innehöll kontrollbitar för att aktivera släckmekanismen eller för att få drönaren att landa.

Som en säkerhetsåtgärd så aktiveras en landningsfunktion om Raspberry Pi:n inte får någon data från datorsystemet under för lång tid. Landingsfunktionen håller drönaren stabil i luften och sänker den långsamt ner mot marken.

3.4 System för höjd- och hastighetsreglering

För att kunna säkerställa säker, autonom flygning behövde reglersystem för höjd- och hastighetskontrollering implementeras. Regleringen som ska möjligöra en stabil styrning kan som en förenkling delas upp i en reglering i varje dimension i rummet. En reglering där höjden i z-led regleras och en i x- respektive y-led där hastigheten regleras. I detta avsnitt presenteras först hur höjdregleringen löstes och därefter det gemensamma reglersystemet för hastighetsreglering i både x- och y-led.

3.4.1 Höjdreglering

Drönaren kan modelleras mekaniskt som en massa, fritt svävande i luften. Nettokraften F som påverkar drönaren medför en acceleration som i sin tur medför en hastighet och slutligen en position h som kan mätas med ultraljudsensorn. Denna modell G kan beskrivas enligt

$$G(s) = \frac{1}{ms^2}, \tag{3.4}$$

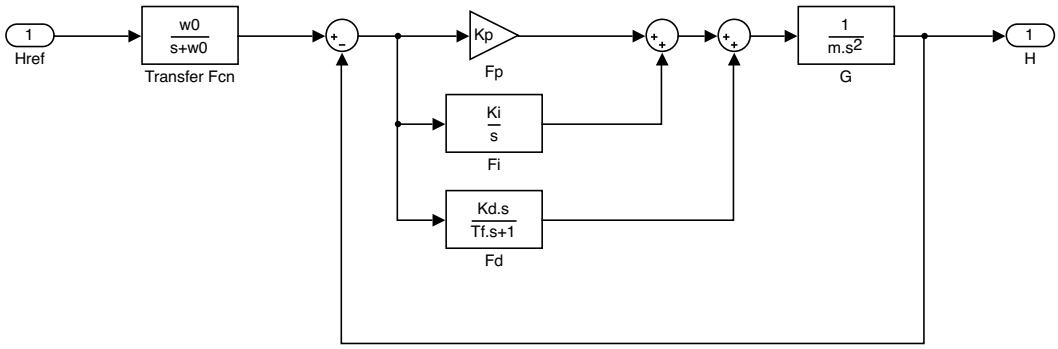
där m är drönarens totala massa. Med hjälp av en regulatorfunktion F_h och en enkel återkoppling kan höjden regleras. Regulatorfunktionen F_h kan antas vara en PID-regulator, se avsnitt 2.6. En stabilitetsundersökning med Routh-Hurwitz stabilitetskriterium (läs Appendix A.3) kunde sedan göras genom att först bilda kretsöverföringen $L(s)$ och på så vis få fram den karaktäristiska ekvationen $1 + L(s) = 0$ för det återkopplade systemet. Detta visas i ekvation 3.5.

$$\begin{gathered} 1 + L(s) = 0 \implies 1 + F(s)G(s) = 0 \implies \ldots \implies \\ mT_f s^4 + ms^3 + (K_d + K_p T_f)s^2 + (K_p + K_i T_f)s + K_i = 0 \end{gathered} \tag{3.5}$$

Routh-Hurwitz stabilitetskriterium ger således att kriterier listade i Ekvation 3.6 måste gälla för att systemet ska vara stabilt. Det kan även utläsas att stora värden på K_d och K_p är önskvärt för ett stabilt beteende.

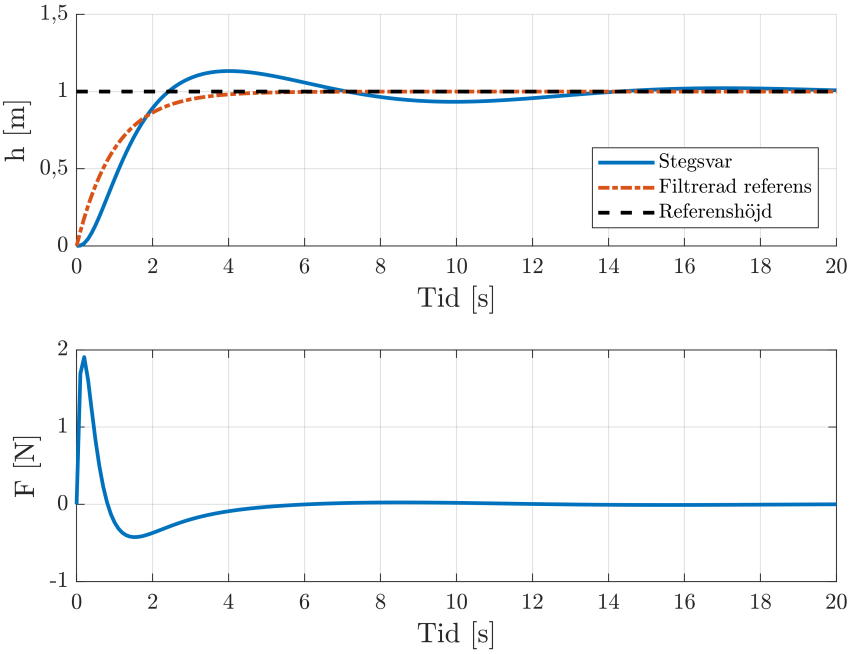
$$
\begin{aligned}
&T_f > 0 \\
&K_d > K_i T_f^2 \\
&K_i > 0 \\
&K_p > \frac{K_i m}{K_d - K_i T_f^2} - K_i T_f
\end{aligned}
\tag{3.6}
$$

För att undvika stora och ryckiga styrsignaler vid ändrade referenssignaler kan ingången lågpass-filtreras. Den slutgiltiga systembeskrivningen visas i Figur 3.10.



Figur 3.10: *Systemdiagram för höjdregleringen. H_{ref} motsvarar referenshöjden och H motsvarar drönarens faktiska höjd.*

Då styrsignalen till drönaren i detta fall är en kraft får den inte överskrida den kraft som drönaren klarar av att leverera. Den resulterande kraften på drönaren beräknas som drönarens lyftkraft minus dess massa gånger tyngdaccellerationen. Drönarens lyftkraft har en värdemängd på $[0, 37]\,N$ (se Appendix A.1). Detta betyder att den resulterande kraften på drönaren har en värdemängd på cirka $[-10, 27]\,N$. I Figur 3.11 visas styrsignalen samt stegsvaret för en rimlig uppsättning av regulatorparametrar.



Figur 3.11: *Den övre grafen visar stegsvaret för höjdreglersystemet. Den undre grafen visar motsvarande styrsignal. Styrsignalens storlek är alltid med god marginal innanför dess värdemängd.*

För att implementera kontrollsystemet på Raspberry Pi:n måste regulatorfunktionen diskretiseras. Detta görs enligt den metod som visas i Appendix A.7. När styrsignalen är uträknad beräknas utifrån den vilken *throttle* som ska ges till drönaren. För att göra detta beräknas först vilken kraft F_z drönaren måste ge för att dess resulterande kraft ska bli lika stor som styrsignalen $u_h(k)$ i Ekvation 3.7.

$$F_z(k) = u_h(k) + mg \tag{3.7}$$

För att hitta rätt *throttle* antas att drönarens styrning är linjär. Detta medför att 0% *throttle* ger en kraft på 0 N, medan 100% *throttle* ger en maximal kraft F_{MAX} och där emellan ökar kraften linjärt. Mängden *throttle* i procent kan i sådant fall beräknas enligt Ekvation 3.8.

$$throttle = 100 \cdot \frac{F_z(k)}{F_{MAX}} \tag{3.8}$$

3.4.2 Hastighetsreglering

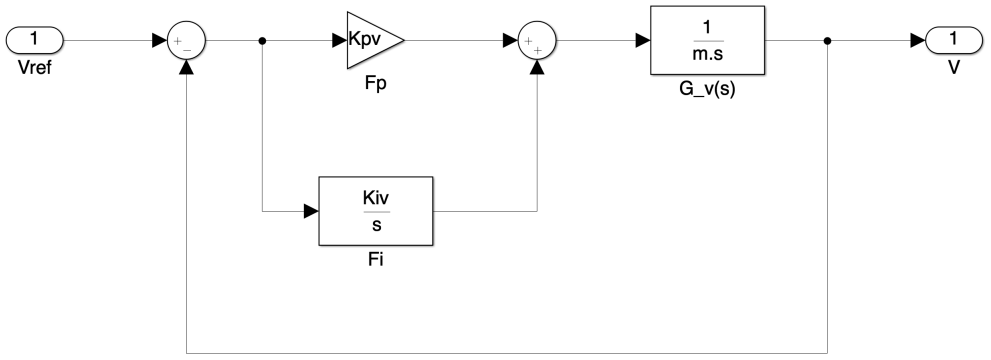
Hastighetsregleringen sker i två dimensioner, men eftersom drönaren är symmetrisk runt x- och y-axeln räckte det att utveckla ett reglersystem och använda detta i båda riktningarna.

På liknande vis som för höjdregleringen modelleras drönaren enligt

$$G_v(s) = \frac{1}{ms}, \tag{3.9}$$

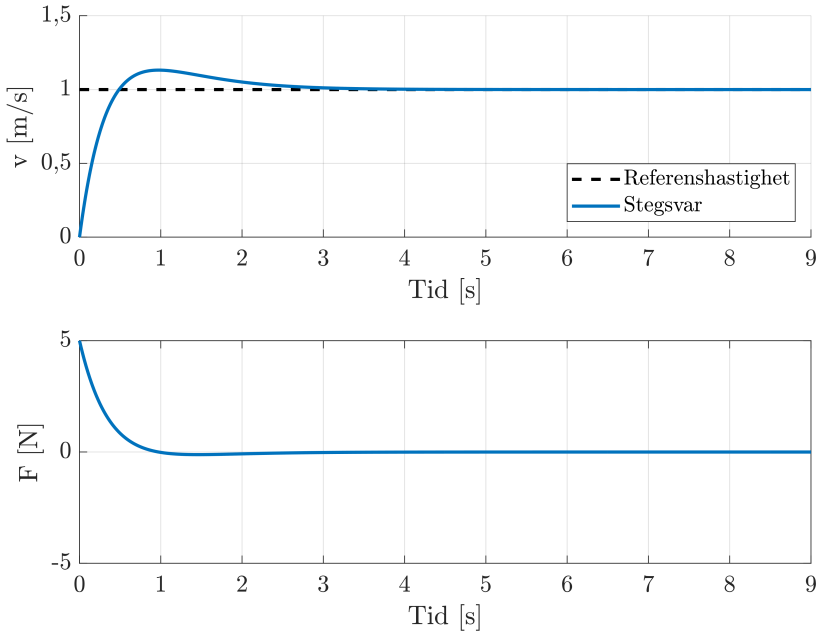
där m är drönarens massa. I detta fall är processen av första ordningen eftersom drönarens acceleration bara behöver integreras en gång till en hastighet. Detta gör även att det inte behövs någon derivata-verkan i regleringen för att det återkopplade systemet ska bli stabilt. En PI-regulator används därför för att lättare ha kontroll över regulatorparametrarna. Regulatorfunktionen framgår i Ekvation 3.10. Det återkopplade systemet framgår i Figur 3.12. En undersökning av Routh-Hurwitz stabilitetskriterium visar att systemet är stabilt så länge K_{iv} och K_{pv} är strikt positiva.

$$F_v(s) = K_{pv} + \frac{K_{iv}}{s} \tag{3.10}$$



Figur 3.12: *Systemdiagram för hastighetsregleringen. V_{ref} är den önskade hastigheten och V är drönarens verkliga hastighet.*

Med lämpliga regulatorparametrar fås stegsvaret samt styrsignalen som visas i Figur 3.13.



Figur 3.13: *I den översta grafen visas det teoretiska stegsvaret för hastighetsreglersystemet. I den undre grafen visas motsvarande styrsignal.*

På samma sätt som för höjdregleringen, förutom att derivata-verkan inte är med, diskretiseras regulatorfunktionen enligt Appendix A.7. Även i detta fall tolkas styrsignalen som en resulterade kraft. Denna kraft måste översättas till en *roll* eller *pitch* vilket motsvarar en önskad vinkel. Låt F_z representera den nuvarande kraften från motorerna i höjdled. Då kan kraften i x-led respektive y-led beräknas enligt

$$\begin{aligned} F_x &= F_z \sin\theta_x \\ F_y &= F_z \sin\theta_y, \end{aligned} \tag{3.11}$$

där θ är lutningen från z-axeln i respektive riktning. Detta kan skrivas om enligt Ekvation 3.12.

$$\begin{aligned} \theta_x &= \arcsin\frac{F_x}{F_z} \\ \theta_y &= \arcsin\frac{F_y}{F_z} \end{aligned} \tag{3.12}$$

Om det antas även i detta fall att styrningen är linjär kan önskad *roll* och *pitch* beräknas enligt Ekvation 3.13, där θ_{MAX} är den maximala tillåtna lutningen på drönaren.

$$
\begin{aligned}
roll &= 100 \cdot \frac{\theta_x}{\theta_{MAX}} \\
pitch &= 100 \cdot \frac{\theta_y}{\theta_{MAX}}
\end{aligned}
\tag{3.13}
$$

3.5 Landningsfunktion

För att säkerställa att drönaren kunde landa säkert implementerades en enkel landningsrutin. Vid förutbestämda tillfällen, som när användaren avbryter flygningen eller Raspberry Pi:n tappar kontakten med laptopen aktiveras den.

Landningsfunktionen fungerar genom att den sätter referenshöjden för höjdregleringen till 0 samtidigt som den sätter *roll*- och *pitch*-värden som får drönaren att stå så still som möjligt i luften. När drönaren väl är nära marken övergår landningsfunktionen till att linjärt sänka *throttle* tills dess att motorerna är avstängda och drönaren står på marken.

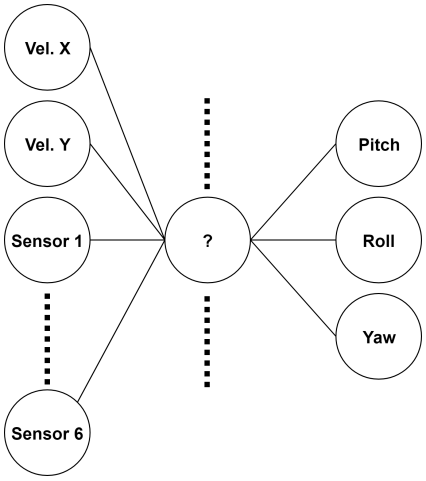
3.6 Neuralt nätverk

Detta avsnitt behandlar processen bakom framtagningen av det neurala nätverket som styr drönaren vid flygning. Programkod för databehandling och modellering skrevs i det funktionella programmeringsspråket Python, då Python hade störst stöd för de i dagsläget mest populära ramverken för databehandling och maskininlärning.

3.6.1 Nätverksstruktur

För att nätverket skulle styra drönaren kontinuerligt behövde drönarens situation vid varje tidpunkt observeras och ges ett motsvarande önskat beteende. Input-noderna i nätverkets första lager består därför av värdena på datan som drönaren genererar, det vill säga hastigheter i meter per sekund samt uppmätta avstånd från sensorerna. De uppmätta avstånden är omskalade så att maxavståndet representeras av värdet 1. Output-noderna i det sista lagret representerar de styrsignaler som ska skickas till drönaren i form av *pitch*, *roll* och *yaw*. Även dessa är omskalade, men till ett spann mellan -1 och 1. Denna mall för nätverket illustreras i Figur 3.14. Då den fysiska drönaren skulle flyga på en jämn höjd togs inte *throttle*-signalen med i beräkningen.

Med anledning av tillgången till simulator var det en god idé att på detta sätt konfigurera nätverket så att det tränas att, givet ett set av inputvärden, förutspår relevanta outputvärden baserat på den insamlade datan från önskvärda flygningsbeteenden i simulatorn. I ett väl tränat nätverk liknar alltså den förutspådda outputen i en viss situation så mycket som möjligt de faktiska styrsignalerna i motsvarade situation i träningsdatan.



Figur 3.14: *Nätverkets yttre lager där det vänstra ledet utgör inputneuroner och det högra outputneuroner. Dessa visar formen på datan som flödar genom nätverket. Eventuella mellanlager där uträkningar sker har utelämnats.*

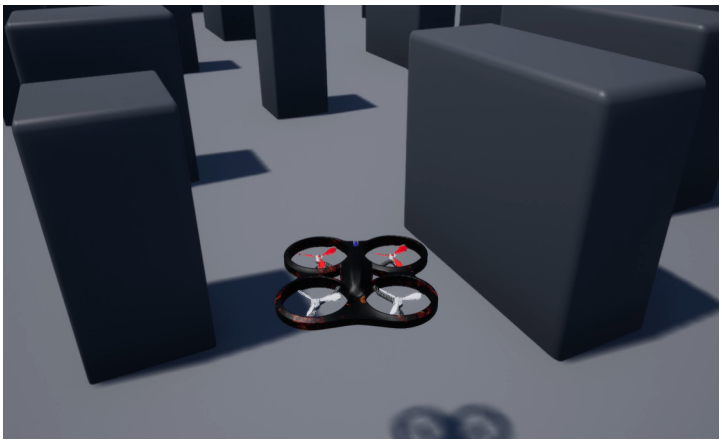
3.6.2 Anpassning av simulerad miljö

Nätverket behövde vara välanpassat till dess ändamål att användas på den fysiska drönaren, och det var därför av intresse att träningsdatan som hämtades från den simulerade drönaren i så stor utsträckning som möjligt överensstämde med den väntade realtidsdatan från den fysiska drönaren. Följaktligen justerades den simulerade drönarens egenskaper i AirSim att noga efterlikna de hos den fysiska drönaren, vilket inkluderar parametrar som chassimodell, propellerstorlek och total vikt samt mätningsfrekvens, räckvidd och förvrängningsgrad av mätdata hos sensorerna. Omgivningen där drönaren skulle flyga konfigurerades även att bestå av ett kvadratiskt rum med hinder enligt avgränsningarna i kapitel 1.3. Rummet illustreras i Figur 3.15.

Figur 3.15: *Överblick av det simulerade flygområdet. Området innehåller flertalet vertikala hinder och avgränsas av väggar. För storleksreferens, se drönaren i mitten av bildens övre del.*

För att kunna använda AirSim för de önskade ändamålen var programmets källkod tvungen att modifieras. Programmets API utökades för att tillåta hämtning och tilldelning av styrsignaler, koden för dataloggning vid inspelning anpassades till att logga hastighet, sensorvärden och styrsignaler, och stöd för att positionera, rotera och fästa multipla sensorer på drönaren implementerades.

Det lättaste sättet att uppnå ett önskvärt flygningsbeteende för nätverket att efterlikna var att styra drönaren i simulatorn manuellt. AirSim erbjuder möjligheten att tolka inputen från en Xbox-kontroll[46] som signaler från en verklig radiosändare för realtidsstyrning av drönare. Detta innefattade två separata styrspakar med stegfri rörelse i två dimensioner, vilket gjorde att relationen mellan mänsklig input och resulterande flygning i hög grad liknade verkligheten. Ett exempel på hur den manuella flygningen kunde se ut illustreras i Figur 3.16.

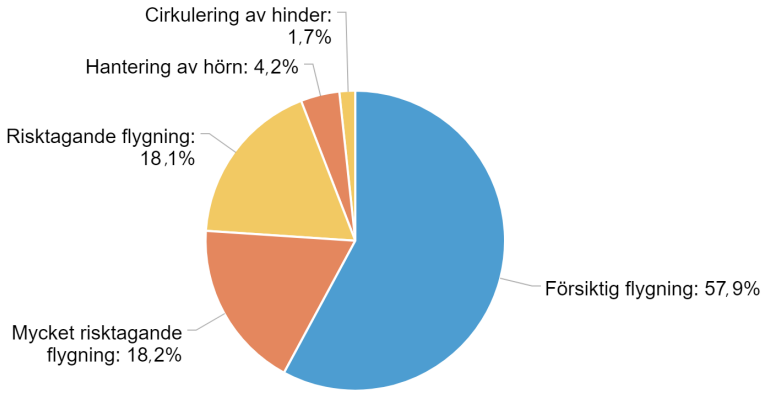


Figur 3.16: *Pågående flygning i den simulerade miljön.*

3.6.3 Insamling av träningsdata

När träningsdatan samlades in var det mycket viktigt att vara medveten om hur drönaren styrdes. Eftersom nätverket inte tar medvetna beslut på samma sätt som en mänsklig förare krävdes förberedelser och inspelning för alla typer av situationer som drönaren kunde tänkas hamna i. Målet var att nätverket skulle efterlikna den inspelade flygningen i så stor grad som möjligt, men det var också oundvikligt att det vid något tillfälle skulle misslyckas med detta och försätta drönaren i en osäker situation, i vilken det skulle krävas att nätverket hade observerat liknande data tidigare så att styrsignaler relevanta till att ta sig ur situationen skulle genereras. Det krävdes också att den totala datan skulle vara fördelad över de olika situationerna på ett sådant sätt att nätverket skulle lära sig att så ofta som möjligt ge ett lugnt och säkert beteende. Samtidigt behövde datan ha en så pass hög frekvens av oönskade situationer att dessa influerar nätverkets träning så att dessa situationer tas i beaktande.

Träningsdatan spelades in i linje med ovanstående resonemang under sammanlagt tre timmar och resulterade i 72 600 datapunkter, varav en exempelföljd bestående av fem datapunkter visas i Tabell 3.1. Majoriteten av den inspelade datan bestod av försiktig flygning eftersom det är beteendet som önskades hos drönaren, men även risktagande samt mycket risktagande flygning där drönaren flögs i högre hastigheter och närmare hinder spelades in. En liten del av datan fokuserade också på specialsituationer som hantering av hörn där drönaren måste vända helt om samt cirkulering av hinder för att drönaren skulle röra sig åt sidan om den råkade flyga väldigt nära ett hinder. Den exakta fördelningen av flygbeteenden illustreras i Figur 3.17. Flygningen var präglad av att hålla ett så konsekvent beteende som möjligt, vilket innebar att flyga på samma sätt i situationer som liknade varandra, i syfte att göra det tydligt vid nätverksträningen vilka beteenden som var önskade vid de olika situationerna. Drönaren styrdes heller aldrig med hastighet bakåt, eftersom det inte fanns sensorer i den riktningen som kunde upptäcka eventuella hinder.



Figur 3.17: *Fördelning av flygningsbeteende i den inspelade datan. Denna uppdelning av träningsdataset gav nätverket det önskade försiktiga beteendet efter träning.*
Notera andelen "Risktagande" och "Mycket risktagande" flygning som krävdes för att ge nätverket möjligheten att vid behov ge starka styrsignaler.

Tabell 3.1: *En exempelföljd av datapunkter från flygdatan med intervall på 50 ms. Följden beskriver en situation där drönaren rör sig framåt åt höger och är nära ett hinder på höger sida, vilket hanteras genom att ge starka styrsignaler i motsatt riktning.*

H. Framåt	H. Sidled	Pitch	Roll	Yaw	L2	L1	L0	R0	R1	R2
0,128	0,161	-0,42	-1	-0,25	2,38	2,38	1,12	0,63	1,33	2,38
0,118	0,145	-0,42	-1	-0,25	2,34	2,34	1,12	0,61	1,17	2,34
0,112	0,133	-0,42	-1	-0,25	2,34	2,34	1,13	0,61	1,10	2,34
0,106	0,121	-0,42	-1	-0,25	2,30	2,30	1,12	0,58	1,01	2,30
0,100	0,110	-0,42	-1	-0,25	2,43	2,43	1,20	0,59	0,96	2,43

3.6.4 Databehandling

Då ett enkelt sätt att hantera dessa stora mängder data var önskvärt formatterades med hjälp av h5py-ramverket[47] den insamlade datan för lagring i .h5-filer. Under denna formattering gavs varje datapunkt en unik tagg varpå kategorisering och uppdelning av all data i separata dataset skedde. Detta resulterade i ett set med träningsdata och ett set med valideringsdata, vilket behövdes senare vid träning av nätverket. Dessutom medförde det att datan sparades ner i en och samma fil, vilket ökade hastigheten med vilken datapunkter kunde laddas in för bearbetning av GPU:n.

I samband med koden för taggning och uppdelning av datapunkter implementerades kod för att sålla ut en del av datapunkterna, i syfte att uppnå en jämnare balans mellan olika typer av datapunkter i dataseten. 95% av datapunkterna där alla sensorer mätte maxavstånd och 70% av datapunkterna där alla styrsignaler var 0 slängdes. Anledningarna till detta var att styrsignalerna som skickas när alla sensorer mäter maxavstånd inte var särskilt relevanta eftersom drönaren då är omedveten om alla detaljer i sin omgivning samt eftersom dataseten innehöll så pass många datapunkter där styrsignalerna var 0 att nätverket annars skulle ha blivit benäget att generera väldigt svaga styrsignaler.

Slutligen, efter processering av datan, implementerades ett system för spegling av datapunkterna där varje datapunkt kopierades och gavs en ny tagg. Vidare bytte värdet på datapunktens sensorer som satt mitt emot varandra plats och värdena på datapunktens styrsignaler i sidled samt hastighet i sidled inverterades. Detta möjliggjorde en dubblering av antalet datapunkter. Tabell 3.2 beskriver en datapunkt där drönaren är på väg mot ett hinder på sin främre högra sida och väjer bakåt åt vänster samt datapunktens speglade motsvarighet där hindret istället är på vänster sida och drönaren väjer bakåt åt höger. Det totala antalet datapunkter efter genomförd databehandling var 65 680.

Tabell 3.2: *Exempel på en datapunkt och dess speglade motsvarighet.*

H. Framåt	H. Sidled	Pitch	Roll	Yaw	L2	L1	L0	R0	R1	R2
0,128	0,161	-0,42	-1	-0,25	2,38	2,38	1,12	0,63	1,33	2,38
0,128	-0,161	-0,42	1	0,25	2,38	1,33	0,63	1,12	2,38	2,38

3.6.5 Träning av nätverk

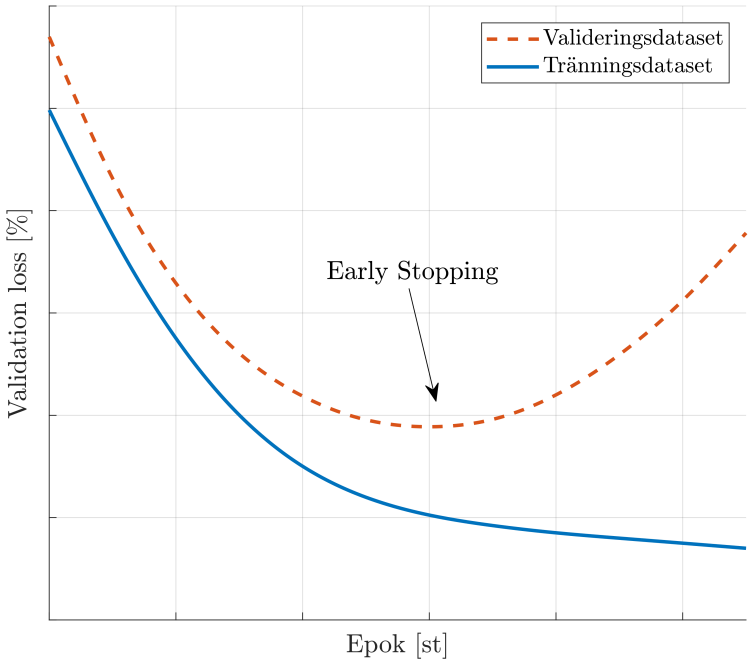
För att den indata som gavs i möjligaste mån skulle generera den motsvarande outputen hos träningsdatan behövde nätverkets vikter iterativt anpassas med hjälp av olika valideringsmetoder. För att uppnå detta användes djupinlärnings-ramverket Keras[48] som är ett högnivå-ramverk med flexibelt och lättåtkomligt API och möjliggör skapandet av djupa nätverk med stora möjligheter till modifikation utan att behöva skriva mycket programkod. Denna fördel följer av att Keras fungerar som ett gränssnitt för att kunna definiera de viktigaste egenskaperna, såsom nätverksstruktur, optimeringsalgoritm och träningslängd, utan att behöva specificera alla kringliggande parametrar i detalj. Istället tas dessa detaljer hand om av andra maskininlärnings-ramverk, i detta fallet TensorFlow[49], för att göra uträkningar och optimering.

Vid starten av träningen specificerades nätverkets djup och lagerbredd, dataseten för träning och validering, en optimeringsmetod samt ett slutkrav. Träningen var en iterativ process som delades upp i så kallade epoker, där varje epok bestod av att datorns processor laddade in alla datapunkter från träningsdatasetet till GPU:n för bearbetning och viktanpassning. När datapunkternas inputs matades in i nätverkets nuvarande konfiguration evaluerades de genererade styrsignalerna och beroende på hur de mätte sig med dess korrekta motsvarighet ändrades vikterna i nätverket i syfte att få de genererade signalerna närmare de korrekta.

Denna viktanpassning gjordes med optimeringsmetoden Nadam[50], en stokastisk sjunkgradientsalgoritm som optimerar en funktion genom att finna dess lokala minimum. Algoritmen användes då den generellt sett fungerar väl i brusiga miljöer, alltså situationer där uppmätt data ofta skiljer sig från verkligheten, vilket bedömdes vara relevant då den insamlade datan från den fysiska drönaren förväntades vara mycket brusigare än i simulatorn.

Varje epok avslutades med en evaluering där nätverket testades mot evalueringsdatasetet, vilket innebar att de genererade styrsignalerna, givet inputs från detta dataset som nätverket inte hade tränats på, jämfördes med deras korrekta motsvarigheter. De utgjorde därför ett mått på hur väl nätverket kunde generera styrsignaler i en ny miljö. Detta mått gavs i formen *validation loss* som motsvarar den aggregerade skiljegraden mellan nätverkets output och den korrekta outputen i evalueringsdatan.

Efter varje epok utvärderades hur väl anpassat nätverket var till dataseten för att avgöra om slutkravet hade uppnåtts och träningen därmed skulle avslutas. Slutkravet definierades som att träningen hade nått en punkt där nätverkets *validation loss* på träningsdatasetet fortsatte att sjunka medan dess *validation loss* på evalueringsdatasetet hade stannat av och börjat stiga. Detta indikerar överanpassning, alltså att nätverket blir så pass anpassat till träningsdatan att dess förmåga att agera i en ny miljö försämras. Denna metod för att avsluta träning kallas ofta *early stopping* [51], vars princip illustreras i Figur 3.18. En annan vanlig metod för att undvika överanpassning, kallad *dropout*, se Appendix A.8, användes även på alla inre lager i nätverket.

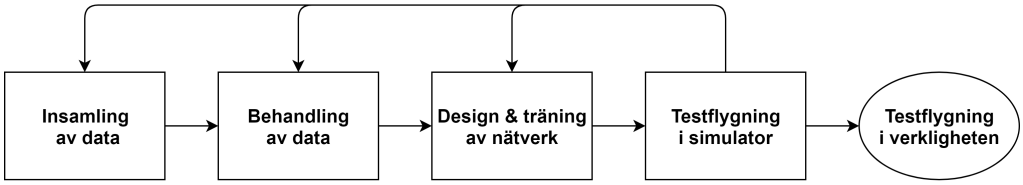


Figur 3.18: *Early stopping vid nätverksträning. Ju lägre validation loss, desto mer anpassat är nätverket till datasetet. Notera att early stopping sker då nätverket endast förbättras mot det dataset den tränas mot och inte valideringssetet som används till att mäta generaliseringsförmågan i liknande dataset. Detta för att undvika överanpassning till träningsdatan.*

Att nå en optimal nätverkskonfiguration och ett välfördelat dataset var en omständig process då det var mycket ovisst hur ett visst nätverk presterar innan empiriska tester hade utförts. De första konfigurationerna som togs fram var därför i hög grad slumpmässigt bestämda och användes för att testa vad som behövde ändras för att nå bättre resultat. Evalueringen av flygprestandan gjordes genom manuella tester i simulatorn baserat på subjektiva mått så som beteende med hänsyn till förmåga att undvika hinder, fart, risktagande och likhet till den inspelade flygningen.

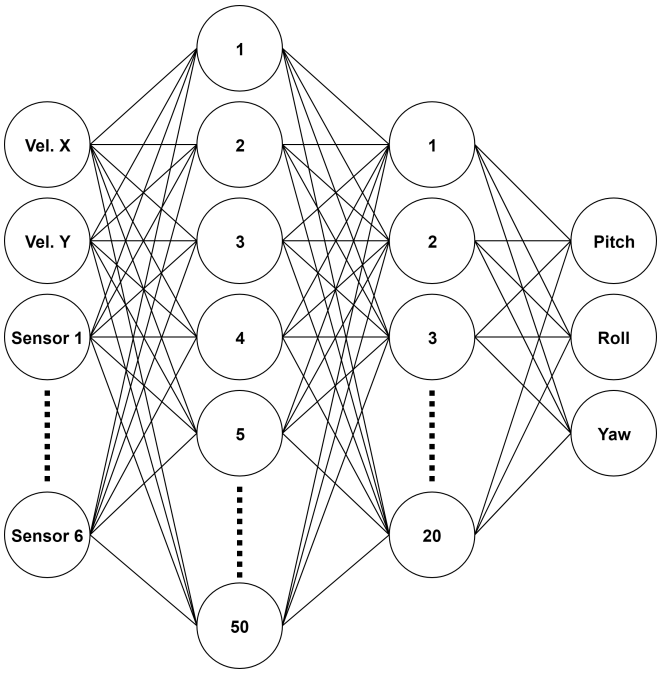
Baserat på denna evaluering gjordes alltså bedömningar om vilka brister som observerats och vilka ändringar som följaktligen behövde göras för att minska dessa oönskade beteenden. Exempel på detta var fall då drönaren fastnat eller gjort fel flertalet gånger i liknande situationer, vilket kunde bero på att träningsdatan var bristande när det kom till att beskriva just dessa situationer. Detta krävde att ny, relevant data spelades in och adderades till datasetet. Träningsdatan kunde också som tidigare nämnts ha varit fördelat på ett sådant sätt att vissa beteendetendenser varit för starka och överrepresenterade delar behövde därför tas bort.

Nätverkets struktur ändrades också kontinuerligt då nätverkets optimala storlek till stor del berodde på den tillgängliga datamängden. Då mängden data eller dess komplexitet ökade behövdes generellt sett ett större nätverk med fler konfigurerbara parametrar för att undvika ökad underanpassning av nätverket. På samma sätt krävdes en minskning av parametrar då datamängden eller komplexiteten minskade.



Figur 3.19: *Implementationsprocessen för det neurala nätverket. Bilden beskriver en iterativ process i simuleringsmiljö, där kvaliteten av nätverkets resultat värderas efter testflygning och ändringar sedan görs i tidigare steg av processen. Endast när flygningen bedöms säker flyttas tester till verkligheten och den fysiska drönaren.*

Träningprocessen var alltså mycket iterativ och baserad på empiriska tester. Träningsdata, databehandlingsmetod samt nätverkets djup och bredd evaluerades i kombination med varandra och påverkade varandra kontinuerligt vilket innebar att de olika stegen i Figur 3.19 ovan behandlades flera gånger. Nätverkets slutgiltiga arkitektur kom att bestå av två inre lager med 50 respektive 20 noder och beskrivs i Figur 3.20.



Figur 3.20: *Det slutgiltiga nätverkets fullständiga arkitektur. Till skillnad från Figur 3.14 visas här även mellanlager där uträkningar sker.*

3.6.6 Anpassning av nätverk till fysisk drönare

Amplituden hos styrsignalerna som nätverket genererade behövde skalas om eftersom de som högst nådde 1. Pulserna som i slutändan styrde rotorhastigheterna hos den fysiska drönaren behövde resultera i en acceleration som vid värde 1 var tillräckligt stark för att drönaren skulle kunna undvika annalkande hinder vid hög hastighet, men inte så pass stark att drönaren fick ett ryckigt och okontrollerat beteende. Det var även viktigt

att svaga styrsignaler fortfarande resulterade i en acceleration som kunde föra drönaren framåt, men som inte var så pass stark att drönarens förmåga att navigera finkänsligt äventyrades. Den mest lämpade omskalningen togs fram genom upprepade flygtester där omskalningsfaktorerna finjusterades efter varje test tills egenskaperna ovan uppnåddes.

Som följd av risken att den fysiska drönaren, på grund av till exempel mätningsfel eller styrningsfel, skulle uppnå hastigheter så höga att nätverket inte längre kunde kontrollera situationen implementerades även ett felsäkringssystem. I felsäkringssystemet observerades hastigheten kontinuerligt och genererade styrsignalen 1 i hastighetens motsatta riktning när för höga värden uppmättes. Systemet reagerade även om drönaren skulle ha en hastighet bakåt, vilket innebar en situation som nätverket inte var tränat att hantera, genom att generera styrsignal 0,5 framåt. Om en eller flera styrsignaler genererades av felsäkringssystemet överskuggades dessa styrsignalerna från nätverket.

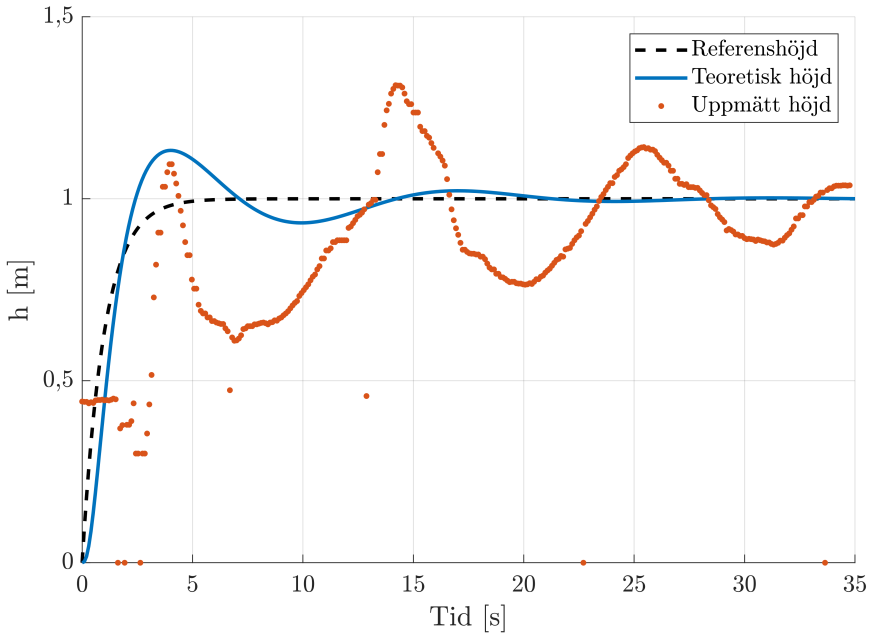
Kapitel 4

Systemtest och resultat

Den resulterande prototypen i detta projekt blev en drönare som kan flyga autonomt inom ett avgränsat område och där implementationen av identifiering och släckning av en eldkälla är påbörjad. I det här kapitlet redovisas resultatet för hur de olika systemen på drönaren fungerade vid verkliga testflygningar och tester. En utförligare diskussion av vad som är utvecklat på drönaren och vad som är kvar att utveckla ges i avsnitt 5.1.

4.1 Verkligt stegsvar hos systemet för höjdreglering

För att kunna säkerställa en stabil flygning utfördes ett test av höjdregleringens förmåga att få upp och hålla sin höjd. Till följd av detta testades systemets verkliga stegsvar, vars resultat visas i Figur 4.1. Systemets stigtid motsvarade ungefär den teoretiska stigtiden. Oscillationer i den verkliga signalen var däremot mycket mer påtagliga i jämförelse med i teorin. Detta kan bero på de förenklingar som gjordes vid modellering av systemet. Figuren antyder dock att oscillationerna minskar i amplitud med tiden. Detta tillsammans med oscillationernas höga periodtid, på cirka 10 sekunder, gör att systemets funktion var tillräcklig för att nå dess syfte.



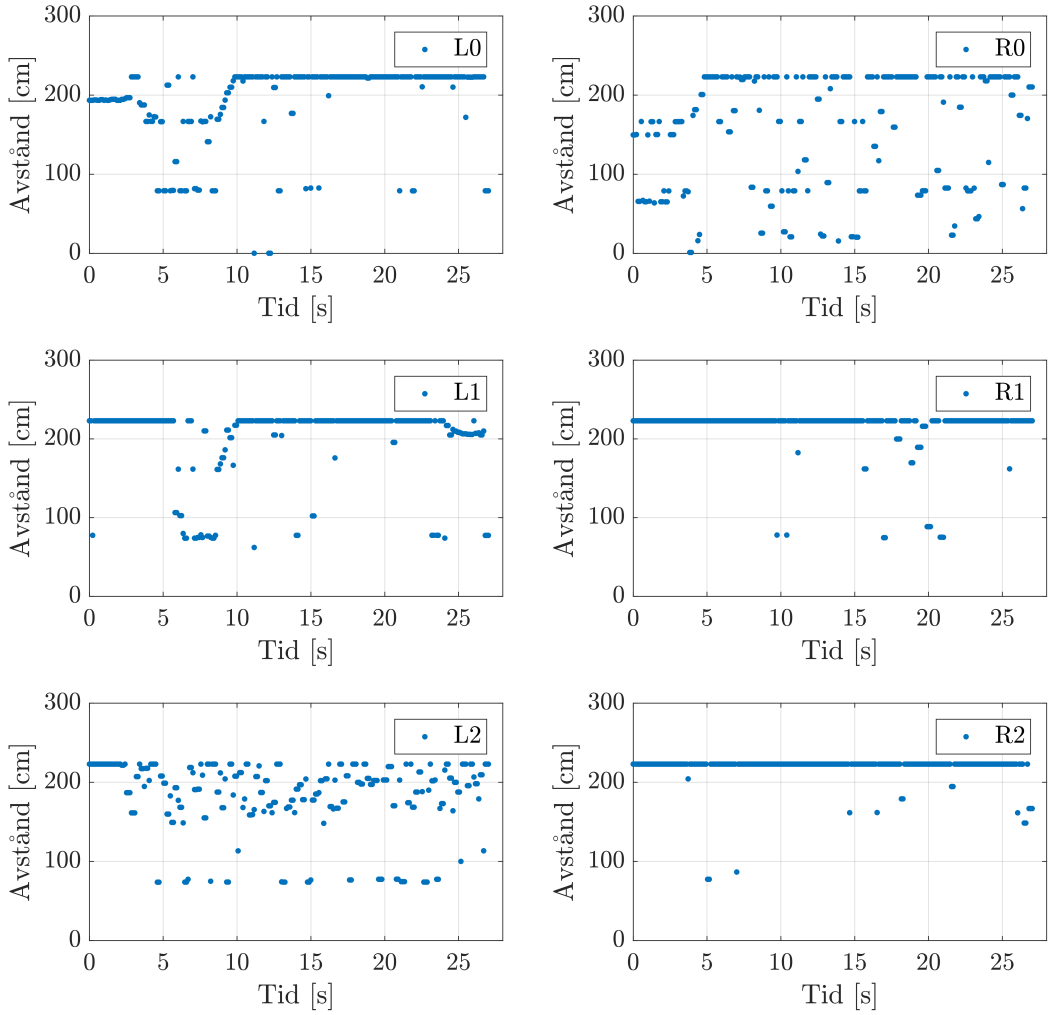
Figur 4.1: *Data från flygning vid test av höjdregleringen. Grafen visar att drönarens höjd stabiliserades på liknande sätt som den teoretiska signalen, fast med större översläng.*

Batteriets laddningsgrad visade sig ha stor inverkan på insvängningstiden hos systemet. Detta beror på att den maximala lyftkraften som drönaren kan åstadkomma är proportionell mot batteriets spänning, vilken sjunker vid minskad laddningsgrad. Då den maximala lyftkraften sjunker minskar även den applicerade kraften vid samtliga styrsignaler. Detta betyder i sin tur att drönaren inte har tillräcklig kraft att lägga sig på rätt höjd med hjälp av dess direkta förstärkning, utan kraften behöver integreras upp med systemets integralfaktor.

4.2 Sensormätningars kvalitet

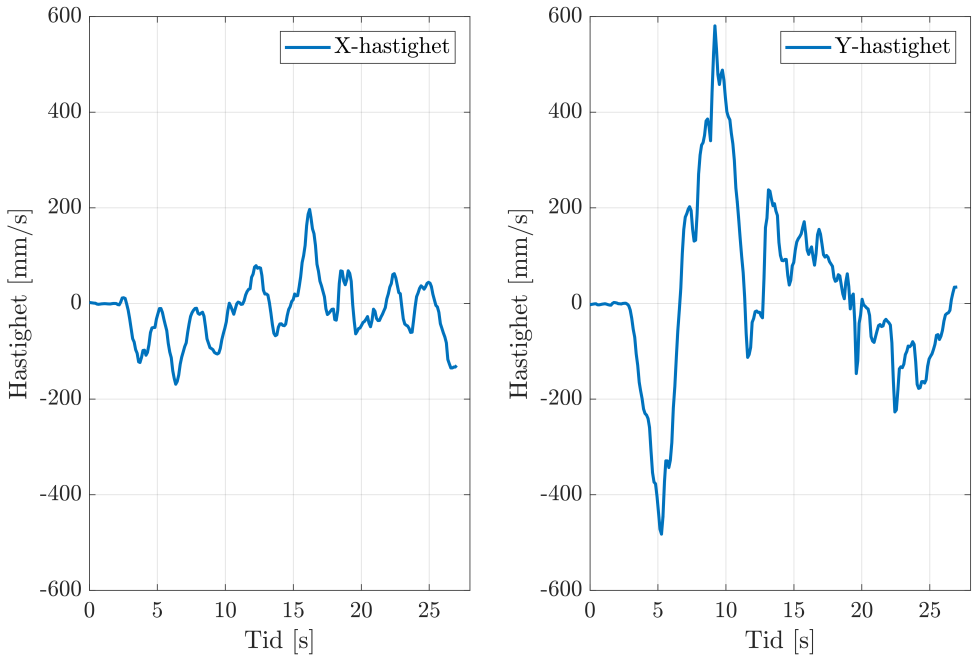
Som nämndes i avsnitt 3.1.1 kunde det uppstå osäkerheter vid mätningarna från ultraljudssensorerna, både från luftflödet från propellrarna och från vibrationer. Tester med drönaren fastmonterad i marken gav goda mätvärden efter att sensorplaceringen var utanför propellrarnas räckvidd och vibrationerna dämpades, se Figur 3.5.

Emellertid fungerade inte alla sensorer lika bra när drönaren var i luften under testflygningarna, se Figur 4.2. Anledningen till detta är svår att analysera då drönaren i luften befinner sig i nära nog konstant rörelse med många olika möjliga faktorer som kan störa mätningarna. Sannolikt är dock att vibrationerna från motorerna fick en annan påverkan då drönaren var i luften jämfört med då den var fastmonterad i marken samt att, som nämndes i avsnitt 2.4, mätningarna fungerar sämre mot mål som har en vinkel mot sensorn. Trots mer än önskvärt brus fungerade sensorerna i den mån att de varnade när hinder kom för nära.



Figur 4.2: *Filtrerad data från ultraljudssensorerna under en flygning. Sensorerna är ordnade från vänster till höger i drönarens flygriktning, där L alltså står för left och R för right. Synfältet för respektive sensor illustreras i Figur 3.2. Notera speciellt hur sensor L2 och R0 i figuren ger mycket brusig data.*

Mätningarna från den optiska flödessensorn fungerade i praktiken bra så länge drönaren flög över det utskrivna mönstret som nämndes i avsnitt 3.1.2. Mätningarna var nog exakta för att drönaren skulle kunna veta åt vilket håll den åkte samt kunna åka i ungefär rätt hastighet. Däremot fanns det inte tillgång till något verktyg för att verifiera att den uppmätta hastigheten stämde överens med den verkliga hastigheten, vilket gjorde kvalitetstestning av den optiska flödessensorn svår att genomföra. Hastighetsdatan för en flygning går att se i Figur 4.3. Storleksordningen och riktningen på hastigheten stämde på ett ungefär; när hastighetsdatan var hög åt ett håll gick det att se att den verkliga hastigheten hög åt samma håll. Även om hastighetsmätningarna ej kunnat verifieras har de fungerat till så vida att det neurala nätverket har fungerat med dem.



Figur 4.3: *Filtrerad hastighetsdata från den optiska flödessensorn under en flygning. Notera att hastighetsdatan trots filtreringen är brusig och inte perfekt matchar verkligheten.*

4.3 Test av värme- och eldidentifiering

Först placerades en tänd marschall i rummet. Marschallen var omsluten halvvägs runt lågan med aluminiumfolie för att hålla kvar värme kring lågan och därmed öka sannolikheten för att elden upptäcks av värmekameran. Avståndet från lågan som värmekameran kunde upptäcka elden ökade från $1\,m$ till $1,5\,m$ med denna lösning.

Därefter fick värmekameran mäta medeltemperaturen vid lågan samt runtomkring i olika vinklar och från avstånden $0,5$, 1 respektive $1,5\,m$. Samtidigt som temperaturen mättes tog en kamera bilder på lågan och den nära omgivningen. Testdatan som samlades in redovisas i Appendix A.9.

Vidare upprepades testet mot en människa och ett utrymme utan varma föremål, varpå högsta temperaturen och koordinaterna för dess pixel togs fram för varje testfall. Efter det jämfördes koordinaterna som värmekameran kommit fram till, med koordinaterna kameran tagit fram. Kamerans koordinatsystem skapades genom att kamerans synfält delats upp i ett 8x8 stort rutnät.

Från testet kan ett flertal slutsatser kring implementation av eldidentifiering dras. För det första kan slutsatsen att kameran inte behövdes dras, efter att en jämförelse mellan koordinaterna värmekameran tog fram och de kameran kommit fram till med hjälp av bilder gjorts. Detta eftersom jämförelsen visar att värmekameran på egen hand kan ge en en relativt exakt position för eldens varmaste punkt. Kameran är alltså överflödig och

monterades därför aldrig på drönaren.

Att enbart använda en värmekamera i syftet att identifiera eld är dessutom fördelaktigt. En fördel är att det ger ett lättare sätt att kunna känna igen en eldkälla jämfört med att använda sig av bildbaserad eldigenkänning. Då drönaren rör på sig hade bildbaserad eldigenkänning ställt höga krav på kommunikationshastigheten över nätverket och på någon form av mekanisk bildstabilisering. I kontrast ger värmekameran simplare värden som är lättare att analysera snabbt, men samtidigt har nog precision på de avstånd som är relevanta.

Vidare visade en jämförelse mellan temperaturvärden för eld, en människa och ett varmt element att värmekameran utan hjälp från en kamera och bilder kan skilja dem åt. Detta eftersom värmekameran genererade temperaturvärden som först och främst inte var så stora, men som även var ungefär desamma som intilliggande och närliggande värden för allt som inte var en eld. För en eld var det med andra ord endast något enstaka värde som avvek från övriga. Värdet avvek dessutom relativt mycket från närliggande värden, under förutsättning att värmekameran befann sig ganska nära eldkällan samt jämfört med hur mycket temperaturvärdena skiljde sig åt för testen med element och människa. Exempelvis visade värmekameran på 40–50 centimeters avstånd från eldkällan en temperatur på över 80°C.

En beskrivning av vad som behöver implementeras för att uppnå delmålet om eldidentifiering ges i avsnitt 5.3.3.

4.4 Prestanda hos det neurala nätverket

För att testa hur väl nätverket klarade av att styra drönaren i simulatorn tilläts drönaren stiga till en höjd på en meter, varpå nätverksstyrningen sattes igång och flygningsbeteendet observerades. Drönaren flög tills kollision uppstod för att få fram ett mått på hur länge nätverket klarade att hålla drönaren flygande.

I det slutgiltiga testet tog det fem minuter och fjorton sekunder innan drönaren kolliderade. Detta skedde dock inte på grund av att nätverket misslyckades med att generera relevanta styrsignaler, utan på grund av att drönarens bakre del kolliderade med ett hinder vid hastighet i sidled, något som kunde ske eftersom inga sensorer var riktade bakåt och det därför inte fanns något sätt att observera det annalkande hindret. Majoriteten av de andra testen slutade också med kollisioner av samma anledning, vilket innebär att det inte finns någon indikation på att drönaren inte hade kunnat flyga ännu längre i simulatorn.

Det observerade beteendet hos drönaren var i enlighet med beskrivningen av delmålet ”Autonom avsökning” i avsnitt 1.2. Nätverket genererar kontinuerligt styrsignaler som för drönaren framåt och håller den ständigt utforskande i området, så länge sensorerna mäter en fri väg framåt. Vid observering av hinder genererades styrsignaler som styr drönaren på en bana som inte resulterar i kollision. Vid kontinuerliga hinder i sidled, som vid flygning parallellt med väggar höll drönaren ett jämnt och stadigt avstånd till hindret. I de tillfällen då vägen framåt var fri, men hinder observerades av sidledssensorerna styrde nätverket drönaren med lägre hastighet och med jämnt avstånd till båda sidor. Drönaren

hade generellt sätt ett lugnt beteende och flög i låga hastigheter, vilket resulterade i att även undvikning av hinder skedde med mjuka långsamma rörelser i de flesta fall. Det observerades fall då drönaren undvek hinder genom mer drastiska rörelser då den till exempel flugit med sidledsvelocitet mot ett hinder på ett sätt så att hindret först var synligt för sensorerna när avståndet var litet. Drönaren lyckades dock i en majoritet av dessa fall undvika kollision. Undantagen var till största delen de fall då hindret aldrig, eller först när avståndet var ytterst kort, var synligt för sensorerna.

4.5 Flygtester

Vid flygtester aktiverades det neurala nätverket då drönaren nått en viss höjd för att sedan flyga autonomt tills en landningssekvens initierats, varpå nätverket deaktiverades och drönaren sjönk mot marken för att landa. Testen framfördes i en miljö där vertikala hinder uppgjorda av kartong placerats inom ett rektangulärt område som avgränsades med väggar av trä.

Verkligheten introducerade flertalet faktorer som medförde att nätverket inte kunnat ge lika konsistenta resultat vid flygningar som de i simulatorn. Vid de tillfällen då mätvärden och nätverkskommunikation fungerade pålitligt observerades ett beteende som liknar det vid flygning i simulator. Drönaren rörde sig i de fallen försiktigt runt i området och höll goda avstånd till hinder och väggar. Testerna var dock som sagt inkonsistenta och en högre kollisionsfrekvens, jämfört med i simulatorn, observerades. Kollisionerna föranleddes av flera faktorer och vid analyser av loggad indata vid kollisionssituationer observerades flertalet gånger felaktiga mätvärden som i vissa fall kunde attribueras till felmätningar från ultraljudssensorerna och i andra fall till paketförluster i den trådlösa kommunikationen mellan Raspberry Pi och dator. Trots en viss tolerans till brusig data var nätverket inte robust nog att överse detta och gav i dessa fallen oönskade styrsignaler som i vissa fall följdes av en kollision. Drönaren observerades ha ett mer konsistent och pålitligt beteende i närheten av de mer ultraljudsreflekterande träväggarna än kring hindrena av kartong och undvek generellt väggarna i en högre grad.

Kapitel 5

Diskussion

I rapportens tidigare delar har det beskrivits hur drönaren och dess styrsystem konstruerades, från komponenter, genom implementation och till de tester som gjorts på prototypen samt de resultat projektet fått som följd.

I det här kapitlet analyseras och diskuteras hur väl resultatet överensstämmer med det syfte och de delmål som introducerades i avsnitt 1.1 respektive 1.2. Det diskuteras även vilka andra lösningar som kunde gjorts under projektet och hur man kan vidareutveckla prototypen. Slutligen presenteras olika reflektioner kring hur projektet kan påverka samhället ur ett etiskt perspektiv.

5.1 Uppföljning av delmål

Nedan presenteras hur väl de fem delmål som definierades i avsnitt 1.2 har uppfyllts samt en specifikation av vad som saknas för att rapportens syfte skulle kunna sägas ha uppfyllts helt.

Den mekaniska konstruktion som utvecklats för att hålla samtliga sensorer och komponenter på plats har till stor mån uppfyllt sitt syfte och har bidragit till den kvalitet som har åstadkommits för datainsamlingen. Vad gäller den strömförsörjning som har utvecklats fungerar den på ett önskvärt sett och har inte inneburit några problem under testningsfasen av produkten.

Trots bruset som syntes i mätningarna från ultraljudet så fungerar mätningarna för relativt säker flygning vid lägre hastigheter. Detta till stor del tack vare det arbete som gjordes med positioneringen av sensorerna för att minimera störningarna från luftflöde och ramvibrationer samt den signalbehandling som utfördes på den insamlade datan. Bedömningen för höjden fungerar däremot bra och var tillförlitlig i den stora majoriteten av fallen, med få felaktiga mätningar.

Hastighetsmätningarna fungerar också godkänt vid flygning i lägre hastigheter ovanför det

utskrivna mönstret efter att resultatet filtrerats. Trots att mätningarna fungerar godkänt så är uppdateringsfrekvensen av mätvärdena på laptopen, framförallt för ultraljudsdatan undermålig. Detta leder till att det neurala nätverket inte får uppdateringar i den takt det behöver för att styra drönaren på ett säkert sätt. Anledningen till detta beror antingen på förluster av datapaket då datan skickas mellan Raspberry Pi:n till laptopen via WiFi eller på buggar i koden för mätningen av ultraljudsdatan som leder till att mätningarna tar längre tid än förväntat. I praktiken leder detta till att flera mätningar i rad hade exakt samma värden, vilket i sin tur leder till felaktiga styrsignaler, se avsnitt 4.5.

Vidare fungerar den höjdregling som finns implementerad tillräckligt bra för att möjliggöra en säker avsökning trots en längre insvägningstid än önskat. En hastighetsreglering är implementerad, men har inte testats, vilket gör att det neurala nätverket alltid aktivt måste styra drönaren. Detta betyder att en stilla flygning, det vill säga utan rörelse framåt, bakåt eller åt sidorna, inte kan säkerställas vid projektets slut.

Nätverket styr drönaren framåt och undviker hinder på vägen, vilket innebär att ingen organiserad avsökning av området sker alltså utan drönaren flyger omkring endast som en funktion av dess omgivning. Detta är inte är en optimal avsökningsmetod eftersom det finns risk att det tar mycket lång tid innan drönaren har sökt av hela området, men den är trots allt god nog för att styrsystemet ska uppfylla de kriterier som beskrivs i delmålet ”Autonom styrning”eftersom det neurala nätverket är viktat till att alltid fortsätta framåt, vilket leder till en kontinuerlig progression i området. Nätverket har validerats till hög grad i simulator och till viss grad i verkligheten. Det fungerar mycket bra i simulatorn där flygtiden är lång nog för att möjliggöra en avsökning av området. I verkligheten fungerar också nätverket godkänt så länge som mätningarna från ultraljudssensorerna och den optiska flödessenorn är tillförlitliga och levereras kontinuerligt. Tyvärr ledde problemen med sensorerna till att nätverket inte fungerade optimalt i verkligheten.

Fortsättningsvis är utvecklandet av en algoritm för att upptäcka värme och identifiera eld påbörjad. Ett sätt för drönaren att känna av värme har tagits fram genom att fästa en värmekamera på den. Med hjälp av temperaturdatan värmekameran genererar kan varmaste temperaturvärdet hittas och det går även att med hjälp av högsta värdet och närliggande avgöra om det är en eld samt hur nära elden drönaren i så fall befinner sig. Kod för detta finns dock inte implementerad.

Därtill är systemet för att släcka en eldkälla på god väg att lösas, men tidsbristen gjorde att testning uteblev. Den nuvarande konstruktionen av släckmekanismen har dålig kvalitet och behöver lösas på ett bättre sätt. Ett annat hinder vid släckningen är att inte heller systemet för hastighetsreglering är testat, vilket betyder att en stilla flygning inte är fullt åstadkommet. Utan en stilla flygning är det svårt att sikta på eldkällan vid aktivering av släcksprayen.

Med tanke på att de delmål som rör eldbekämpning inte har fulländats och således ej heller testats går det inte att dra slutsatser om hur väl de hade fungerat, varken i testmiljöer eller verkliga situationer. Med detta sagt hade en lyckad testning i bästa fall kunnat påvisa att det finns stöd för vidare forskning på ämnet och mer extensiva projekt hade behövts för att validera konceptet av eldbekämpning med autonom drönare i verkligheten.

5.2 Annorlunda begynnelsevillkor

En stor del av det konstruktionsarbete som gått in i projekt har varit riktat mot att implementera funktioner eller byta delar som vid starten av projektet antogs vara funktionella och tillräckliga för syftet. Om antagandet att vi hade en flygfärdig drönare som endast behövde utökas, snarare än modifieras hade stämt med verkligheten, hade vägen till testning och sammankoppling av systemet varit kortare. Detta hade i sin tur tillåtit en process där delmålen kunnat uppfyllas på basnivå betydligt tidigare och projektets fokus därför handlat om att förfina och perfektera delmålen. Detta hade medfört att alla delmål, inklusive eldbekämpning, hade kunnat sättas i test, vilket hade lett till mer nyanserade resultat.

Vidare var startsträckan för att börja samla in data och träna samt köra nätverket i AirSim mycket lång då AirSim fortfarande var i utvecklingsfas vid projektets start. Detta innebar framförallt att dokumentationen var bristfällig och i många fall frånvarande, men också att flera delar av programmet hade buggar eller var implementerade på ett sådant sätt att utökning av koden var komplicerad. En stor del av arbetet med att ta fram det neurala nätverket bestod av att väldigt grundligt sätta sig in i programmets flöde för att ta reda på vilken del av koden som gör vad och sedan utöka den, men också att lösa många olika buggar. Denna del av arbetet resulterade i en accepterad pull request till AirSims git-repo, vilket markerar att processen definitivt var omfattande.

Om det vid projektets start hade funnits tillgång till en simulator där datainspelning och drönarstyrning hade kunnat ske direkt, hade en mycket större portion av den disponerade tiden för implementation av artificiell intelligens kunnat användas till att raffinera och färdigställa nätverksstrukturen. Mer resurser hade då kunnat läggas på andra delar av projektet.

5.3 Vidareutveckling av projektet

Produkten projektet resulterade i är långt ifrån färdigutvecklad. Detta dels eftersom alla delmål inte hann uppfyllas och dels då det finns möjlighet till förbättring av precision och stabilitet.

I det här avsnittet diskuteras vad ett uppföljningsprojekt kan göra annorlunda och hur man kan vidareutveckla projektet i framtiden.

5.3.1 Byte av hårdvara

Just nu är datorkraften i prototypen uppdelad i två delar med Raspberry PI:n på drönaren och det neurala nätverket på en laptop. Om det går att installera en lika lätt, men mer beräkningskraftig dator på drönaren så kan man helt komma runt kommunikationen mellan Raspberry Pi:n och laptopen. Därmed kan fördröjningen mellan att sensordatan läses och att en handling sker minska och risken att man tappar uppkopplingen mellan de olika

delarna av systemet elimineras. En kraftigare dator hade troligen också inneburit att den kräver mer ström, vilket ställer ytterligare krav på batterier och strömförsörjningen som också måste tas i beaktande.

Även om den optiska flödessensorn fungerade godkänt så har den ett flertal begränsningar. För det första behöver den vara på rätt höjd ovanför marken för den fokallängden den är inställd på. För det andra behöver den en tydligt texturerad yta och bra ljus och för det tredje måste den ha höjddata för att fungera. Detta leder till att vilka miljöer sensorn fungerar i blir något begränsade. Om en ny prototyp ska utvecklas som kan fungera i olika miljöer krävs en sensor som kan hantera de miljöerna. Utrustning som kan testas här inkluderar GPS, accelerometrar och system som bestämmer drönarens position utifrån ankarpunkter i området. Dessa, tillsammans med eller istället för den optiska flödessensorn, kan ge bättre mätningar av hastigheten i fler miljöer och därmed göra drönaren mer flexibel.

Som det diskuterades i avsnitt 3.1.1 är ultraljudssensorerna känsliga för störningar, både från vibrationer och från luftflödet från propellrarna. Dessutom har de en begränsad räckvidd på ungefär 3 meter. Två möjliga alternativ för att förbättra detta är att antingen använda sig av ett annat mätinstrument som till exempel LIDAR istället för ultraljud eller försöka hantera upptäckandet av hinder via kameror istället.

En populär modell av LIDAR som passar till Raspberry Pi är LIDAR Lite v3, som har en räckvidd på 40 meter [52] istället för ultraljudens 3 meter. Då ljus inte påverkas lika mycket som ljud av strömmande luft är det även rimligt att anta att luftflödet från propellrarna skulle vara ett mindre problem än i nuläget. Kamerabaserad hinderundvikning kräver en del beräkningskraft och smarta lösningar för att upptäcka var hindren är. Detta är dock ett område som är mycket intressant just nu och det finns rapporter på hur det går att göra och därmed kan det också vara ett intressant område att titta närmare på.

5.3.2 Förbättring av höjdreglering

Som nämns i avsnitt 4.1 finns en del oscillationer i reglersystemets stegsvar, samt en inkonsekvens i systemets insvängningstid, beroende på batteriets laddningsgrad. Detta är problem som till stor grad hade kunnat förbättras vid en vidareutveckling. Oscillationer kan främst elimineras genom en noggrannare modellbestämning, alternativt genom manuella parameterjusteringar vid upprepade tester. Variansen hos insvängningstiden fås lättast bort genom mätning av batteriets spänning samt korrigering av drönarens styrsignalstyrka i realtid.

En stor begränsning hos det nuvarande systemet är den höjdbegränsning som finns inbyggd i den optiska flödessensorns ultraljud. För det första kan den inte mäta lägre värden än 30 cm, vilket ger dålig kontroll av drönaren väldigt nära marken. Därtill kan den inte mäta värden över tre meter, vilket gör höjdreglering över två meter riskabel och över tre meter omöjlig. Vid kortare flygning samt inomhusflygning duger en höjdreglering i detta intervall, men vid eventuellt utomhusbruk och om drönaren ska färdas längre sträckor kan en högre referenshöjd vara önskvärd. En lösning hade kunnat vara att använda LIDAR, vilket nämns i föregående avsnitt. En annan lösning skulle kunna vara använda sig av

barometermätningar vid högre höjder, det vill säga över 2 till 3 meter och ultraljud vid lägre höjder.

5.3.3 Implementation av eldidentifiering

Slutsatsen av hela värme- och eldidentifieringstestet, se avsnitt 4.3, att det relativt enkelt går att vidareutveckla projektet och implementera eldidentifiering hos drönaren. För att få drönaren att kunna identifiera eld behöver en algoritm för att analysera datan från värmekameran tas fram. Algoritmen måste leta upp högsta temperaturvärdet i 8x8 rutnätet, bestående av temperaturdata, och utifrån det kunna avgöra ungefär hur långt bort från värmekällan drönaren är. Vidare måste algoritmen utifrån närliggande värden och hur högt temperaturvärdet är kunna avgöra om en eld har hittats.

Efter att en eld har identifierats måste drönaren på ett säkert sätt kunna närma sig elden och lyckas sväva på rätt höjd och i rätt vinkel i förhållande till eldkällan, vilket kan bestämmas utifrån rutnätet genom att förflytta drönaren så att det varmaste, uppmätta temperaturvärdet befinner sig i mitten av det.

5.3.4 Revidering av släckmekanism

Även om det finns en släckmekanism implementerad på prototypen är den i låg grad en effektiv brandsläckare. För det första är det mekaniska systemet grovt undermåligt. Släckanordningen är inte stabilt monterad och aktiveringen är beroende av att en kraft appliceras på ett munstycke, vilket inte heller sitter fast på ett tillförlitligt sätt. För det andra har ingen studie gjorts i huruvida det valda släckmedlet är att föredra vid passande bränder. För att produkten ska kunna vara en effektiv brandsläckare behöver mycket arbete läggas på att mekanismen är tillförlitlig och effektiv samt att det är lätt att träffa den eld som är identifierad.

Eftersom det i detta projekt inte lades någon tid på att faktiskt försöka träffa en eld med släckmedlet behöver även detta utvecklas. Ett kontrollsystem behöver tas fram som aktivt kan hålla drönaren på rätt höjd och avstånd från eldkällan för att kunna släcka den, utan att drönaren förstörs av den potentiella värmen från elden.

5.3.5 Förbättring av artificiell intelligens

Den vidareutveckling som skulle möjliggöra för störst förbättring av den artificiella intelligens hade varit en utökning av vilken information som finns tillgänglig för det neurala nätverket som indata. Med den nuvarande konfigurationen av ultraljudssensorer har det funnits en tydlig gräns för vad intelligensen kan åstadkomma, då den enda information som funnits tillgänglig är ett fåtal avståndsmätningar i generella riktningar. Avståndsmätningarna har som bäst tillåtit nätverket att kunna fungera som ett system för att undvika kollisioner. Det har inte funnits möjlighet till att lära nätverket mer avancerade navigeringsmanövrar eller ge det möjlighet att skilja på snarlika hinder då den

tillgängliga datan varit ganska abstrakt, men framförallt väldigt gles.

Genom att skapa en preferens hos nätverket att styra drönaren framåt har systemet ovan dock förlängts till en form av utforskning samtidigt som det undviker hinder. En mer fullständig och precis bild av samma dimensioner, avstånd och riktning, genom till exempel användning av en LIDAR, se avsnitt 5.3.1, eller kameror i stereouppsättning, skulle emellertid höja nivån av vad som är möjligt att åstadkomma. Det skulle kräva mer beräkningskraft, men inte nödvändigtvis öka komplexiteten vid implementationen eller kräva stora förändringar i dess process då det endast innebär att en ny optimal nätverksstruktur skulle behöva hittas. Nätverksstrukturen skulle i så fall skapas genom att modifiera den implementation som redan finns tillgänglig. En mindre drastisk utökning på samma spår som också hade medfört förbättringar hade varit om drönaren varit utrustad med sensorer av den typ som redan finns på drönaren, men ha fler utav dem och i fler vinklar. Framförallt hade sensorer bakåt behövts då avsaknaden av det har varit den största orsaken till kollisioner i simulatormiljö. Detta har motverkats genom att inte låta drönaren flyga bakåt med hjälp av mjukvaruspärrar, men det är tydligt att möjlighet att flyga helt fritt i planet hade varit fördelaktigt.

Om nätverket hade haft tillgång till drönarens relativa position i rummet, vilket hade varit fallet i en uppskalad version av detta projekt där drönaren hade flugit i stora områden och GPS-positionering kunnat användas, skulle det öppna för möjligheten att algoritmiskt planera flygrutter för att effektivisera avsökningen av utrymmet efter eld. Detta hade lett till ett mer komplett uppfyllande av delmålet ”Autonom avsökningdå avsökningen hade blivit mycket mer raffinerad än att enbart röra sig förbi hinder stegvis och se vad som finns bakom. Detta skulle dock utöka komplexiteten av systemet och kräva extra komponenter för att lagra data om miljön samt komma ihåg var drönaren tidigare befunnit sig.

Projektet avslutades innan nätverket hann gå igenom en hel iteration av testning i verkligheten och planerade förbättringar hann således inte fullbordas. En andra fas i nätverkets utveckling hade planerats där så kallad *reinforcement learning*, se Appendix A.10, skulle användas i simulatorn. Det nätverk som tagits fram genom djupinlärning och som var baserat på manuellt inspelad data skulle då fungera som en bas som kunde förfinas. Förfiningen skulle ha skett genom en process där drönaren hade fått flyga runt utan mänsklig observation under lång tid och algoritmiskt justera sina vikter baserat på fördefinierade negativa och positiva resultat. Exempel på sådan feedback kan vara att ge negativ feedback vid kollision och positiv feedback vid lyckad avsökning av en stor andel av utrymmet.

5.4 Samhälleliga och etiska aspekter

Drönare är lättillgängliga och inte särskilt komplicerade att komma igång med och styra, även för privatpersoner. Detta innebär att det numera finns en tillgång till luftrummet som tidigare enbart användes i professionella syften. Det har lett till politiska diskussioner med bland annat nya lagar som följd [7]. Den största aspekten rör störningar av privatliv, där kamerautrustade drönare kan ta bilder och filmer som inkräktar på människors privatliv. Störningar kan även uppstå i form av ljud, vilket kan få negativa konsekvenser på naturlivet och dess lokala populationer [53].

Problemen ovan finns redan med manuellt kontrollerade drönare, dock finns det en risk att det blir lättare att inkräkta på andras privatliv med en autonom drönare som självmant kan förfölja en person och ta bilder. Hårdvaru- och mjukvarubuggar medför även en ökad risk för människor och natur om man låter drönaren ta beslut om mål och flygväg själv utan en människa som kan avgöra när flygningen blir för farlig.

Fördelarna med utgången av ett projekt som detta väger dock upp nackdelarna. Det finns flera samhälleliga fördelar som en uppskalad version av projektet skulle kunna medföra. Att kunna lokalisera samt släcka en brand på så kort tid som möjligt är kritiskt innan elden hinner sprida sig. Ett obemannat fordon som dessutom kan utföra detta på egen hand äventyrar heller inga ytterligare liv utöver de som eventuellt hotas av branden i sig. En uppskalning av projektet med drönaren som utvecklats genom detta projekt som bas skulle alltså kunna rädda liv.

Vidare är den metod som togs fram för den artificiella intelligensen i detta projekt relativt allmän och skulle därför efter omarbetning kunna användas inom flera andra områden. Sådana områden kan till exempel inkludera att leta efter försvunna människor i terräng och till sjöss.

Litteraturförteckning

[1] Volvo Car Corporation, "Introducing drive me: Shaping the future of autonomous driving,"2018. [Online]. Tillgänglig: https://www.volvocars.com/intl/buy/explore/intellisafe/autonomous-driving/drive-me, hämtad: 2018-04-18.

[2] Amazon, "Amazon Prime Air,"2018. [Online]. Tillgänglig: https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011, hämtad: 2018-02-09.

[3] S. Babcock, "BGE is testing autonomous drone inspection of power lines,"2018. [Online]. Tillgänglig: https://technical.ly/baltimore/2018/02/19/bge-testing-autonomous-drone-inspection-power-lines/, hämtad: 2018-05-09.

[4] FliteTest, Producent, "Fire Fighting Drone | FliteTest", *Youtube*, 2015. [Video]. Tillgänglig: https://www.youtube.com/watch?v=glmBvkQhKk4, hämtad: 2018-04-17.

[5] M. Margaritoff, "Drones in Firefighting: How, Where and When They're Used,"2017. [Online]. Tillgänglig: http://www.thedrive.com/aerial/16770/drones-in-firefighting-how-where-and-when-theyre-used, hämtad: 2018-05-09.

[6] D. Quick, "Autonomous air tankers could soon joining the fire-fighting fray," *New Atlas*, jan. 2018. [Online]. Tillgänglig: https://newatlas.com/autonomous-air-tanker-firefighting/53029/, hämtad: 2018-02-09.

[7] Transportstyrelsen, "Nya regler för drönare gäller från den 1 februari 2018,"u.d. [Online]. Tillgänglig: https://www.transportstyrelsen.se/sv/luftfart/Luftfartyg-och-luftvardighet/Obemannade-luftfartyg-UAS/nya-regler-for-dronare/, hämtad: 2018-02-09.

[8] A. Segales, R. Gregor, J. Rodas, D. Gregor, S. Toledo, "Implementation of a low cost UAV for photogrammetry measurement applications,"i 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 2016, ss. 926–932. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/7502609/, hämtad 2018-04-01.

[9] MT2212 750KV, u.d. [Online]. Tillgänglig: https://www.rcflight.se/visaprodukt.aspx?id=1983&p=t-motor-mt2212-750-kv, hämtad: 2018-05-04.

[10] C. Chéron, A. Dennis, V. Semerjyan, Y. Chen, "A multifunctional HIL testbed for multirotor VTOL UAV actuator,"i *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, QingDao, Kina, 2010, ss. 44–48. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/5552032/, hämtad: 2018-04-01.

[11] P. Pounds, R. Mahony, "Design principles of large quadrotors for practical applications,"i 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, ss. 3265–3270. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/5152390/, hämtad 2018-04-01.

[12] Butikssäljare på RCFlight, privat kommunikation, feb. 2018.

[13] Gens ace, "Gens ace 3300mAh 14.8V 25C 4S1P Lipo Battery Pack,"u.d. [Online]. Tillgänglig: http://www.gensace.de/gens-ace-3300mah-14-8v-25c-4s1p-lipo-battery-pack.html, hämtad: 2018-02-03.

[14] E300: *System*: DJI, 2017, [Online]. Tillgänglig: https://www.dji.com/e300/spec, hämtad: 2018-03-04.

[15] *FlameWheel 450 User Manual V2.2*: DJI, u.d. [Online]. Tillgänglig: dl.djicdn.com/downloads/flamewheel/en/F450_User_Manual_v2.2_en.pdf, hämtad: 2018-04-18.

[16] Housegard, "FirePal släckspray – Kitchen,"2016. [Online]. Tillgänglig: http://www.housegard.com/products/firepal-slackspray-kitchen-ny/, hämtad: 2018-05-03.

[17] Hobby King, "HXT900 Micro Servo 1.6kg / 0.12sec / 9g,"2018. [Online]. Tillgänglig: https://hobbyking.com/en_us/hxt900-micro-servo-1-6kg-0-12sec-9g.html?___store=en_us, hämtad: 2018-02-09.

[18] *Distance Measuring Sensor Unit: GP2Y0A710K0F*, Japan: Sharp, 2006. [Online]. Tillgänglig: https://cdn-shop.adafruit.com/datasheets/gp2y0a710k.pdf, hämtad: 2018-05-03.

[19] P. Malheiros, J. Gonçalves, P. Costa, "Towards a more Accurate Infrared Distance Sensor Model,"i International Symposium on Computational Intelligence for Engineering Systems, Porto, Portugal, 2009. [Online]. Tillgänglig: http://www.gecad.isep.ipp.pt/iscies09/Papers/19November/iscies09_sharp_model.pdf, hämtad: 2018-05-10.

[20] *PING))) Ultrasonic Distance Sensor (#28015)*, USA: Parallax, 2013. [Online].

Tillgänglig: https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf, hämtad:
2018-02-09.

[21] *Ultrasonic Ranging Module HC - SR04*: ElecFreaks, u.d. [Online]. Tillgänglig: http://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf, hämtad:
2018-02-11.

[22] Okänd, "ljudhastighet,"i *Nationalencyklopedin*. [Online]. Tillgänglig: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/ljudhastighet, hämtad: 2018-03-28.

[23] Pixhawk, "PX4FLOW Smart Camera,"u.d. [Online]. Tillgänglig: https://pixhawk.org/modules/px4flow, hämtad: 2018-02-09.

[24] D. Honegger, L. Meier, P. Tanskanen and M. Pollefeys, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications,"i 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Tyskland, 2013, ss. 1736–1741. [Online].
Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6630805, hämtad: 2018-04-09.

[25] *Adafruit AMG8833 8x8 Thermal Camera Sensor*: Justin Cooper, 2018. [Online]. Tillgänglig: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-amg8833-8x8-thermal-camera-sensor.pdf, hämtad: 2018-02-17.

[26] R. Gade and T. B. Moeslund, "Thermal cameras and applications: a survey," Machine Vision and Applications, vol. 25, nr. 1, ss. 245–262, 2014.

[27] *Built-in Sensors 2015-2016: Infrared Array Sensor Grid-EYE (AMG88)*, Kadoma, Japan: Parasonic, 2015. [Online].
Tillgänglig: https://industrial.panasonic.com/content/data/CP/PDF/built-in_sensor_catalog_e.pdf, s. 39, hämtad: 2018-04-04.

[28] Pi Supply, "Raspberry Pi Camera Board v1.3 (5MP, 1080p),"2018. [Online]. Tillgänglig:
https://www.pi-supply.com/product/raspberry-pi-camera-board-v1-3-5mp-1080p/, hämtad: 2018-02-09.

[29] ArduPilot Dev Team, "Stabilize Mode,"2016. [Online]. Tillgänglig: http://ardupilot.org/copter/docs/stabilize-mode.html, hämtad: 2018-02-09.

[30] ArduPilot Dev Team, "Archived: Connecting the Radio Receiver (APM2)," 2016. [Online]. Tillgänglig:
http://ardupilot.org/copter/docs/common-connecting-the-radio-receiver-apm2.html, hämtad: 2018-02-09.

[31] I. Poole, "IEEE 802.11n Standard — Wi-Fi WLAN — Radio-Electronics.com,"u.d.

[Online]. Tillgänglig:
http://www.radio-electronics.com/info/wireless/wi-fi/ieee-802-11n.php, hämtad: 2018-05-04.

[32] M. Mukhtar, "Bluetooth 4 Vs. Bluetooth 5: A feature comparison," *iTechtics*, jan. 2017. [Online]. Tillgänglig:
https://www.itechtics.com/bluetooth-4-vs-bluetooth-5-feature-comparison/, hämtad: 2018-05-04.

[33] Raspberry Pi Foundation, "Raspberry PI 3 Model B,"u.d. [Online]. Tillgänglig:
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, hämtad: 2018-02-09.

[34] B. Lennartson, *Reglerteknikens grunder*, 4 uppl., Lund, Sverige: Studentlitteratur, 2011.

[35] S. Shah, D. Dey, C. Lovett, A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,"uppsats, Cornell University, Ithaca, NY, USA, 2017. [Online]. Tillgänglig: https://arxiv.org/abs/1705.05065, hämtad: 2018-02-19.

[36] Epic Games, Inc., "What is Unreal Engine 4,"2018. [Online]. Tillgänglig:
https://www.unrealengine.com/en-US/what-is-unreal-engine-4, hämtad: 2018-05-03.

[37] S. Shah, "KiteDemo2,"2017. [Elektronisk bild]. Tillgänglig:
https://github.com/Microsoft/AirSim/blob/master/docs/images/screenshots/KiteDemo2.png, hämtad: 2018-05-04.

[38] J. A. Hertz, A. S. Krogh, R. G. Palmer, *Introduction To The Theory Of Neural Computation*, Santa Fe, CA, USA: Addison-Wesley Publishing Company, 1990.

[39] S. S. Haykin. *Neural networks: and Learning Machines*. 3 uppl. Upper Saddle River, NJ, USA: Pearson Education, 2009.

[40] Y. LeCun, Y. Bengio, G Hinton, "Deep Learning," *Nature*, vol. 521, nr. 14539, ss. 436–444, maj 2015. doi:10.1038/nature14539, [Online]. Tillgänglig:
https://www.nature.com/articles/nature14539, hämtad: 2018-05-02.

[41] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Book in preparation for MIT Press, 2016. [Online]. Tillgänglig: http://www.deeplearningbook.org, hämtad: 2018-06-01.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, ss. 1929–1958, 2014. [Online]. Tillgänglig:
http://jmlr.org/papers/v15/srivastava14a.html, hämtad: 2018-05-01.

[43] Dennisbaldwin, DJI F450 F550 Flamewheel Long Leg Extensions - 5mm Thick, https://www.thingiverse.com/thing:94537, Ändrad, CC BY, https://creativecommons.org/licenses/by/3.0/.

[44] Pigpio, "pigpio C Interface,"2018. [Online]. Tillgänglig: http://abyz.me.uk/rpi/pigpio/cif.html, hämtad: 2018-02-15.

[45] The Open Group, "The Single UNIX ® Specification, Version 2," 1997. [Online]. Tillgänglig: http://pubs.opengroup.org/onlinepubs/7908799/xsh/usleep.html, hämtad: 2018-02-15.

[46] S. Shah, "AirSim - Remote Control," Github Repository, Microsoft, Redmond WA, USA, 2018. [Online]. Tillgänglig: https://github.com/Microsoft/AirSim/blob/master/docs/remote_control.md, hämtad: 2018-04-30.

[47] A. Collette, "HDF5 for Python,"u.d. [Online]. Tillgänglig: http://www.h5py.org/, hämtad: 2018-04-30.

[48] F. Chollet, San Francisco, CA, USA "Keras: The Python Deep Learning library,"2018. [Online]. Tillgänglig: https://keras.io/, hämtad: 2018-05-01.

[49] Google Inc., Mountain View, CA, "TensorFlow: An open source machine learning framework for everyone,"u.d. [Online]. Tillgänglig: https://www.tensorflow.org/, hämtad: 2018-05-01.

[50] T. Dozat, "Incorporating Nesterov Momentum into *Adam*," Projekt, Stanford University, Stanford, CA, USA, 2015. [Online]. Tillgänglig: http://cs229.stanford.edu/proj2015/054_report.pdf, hämtad: 2018-05-01.

[51] Skymind, "Early Stopping,"2017. [Online]. Tillgänglig: https://deeplearning4j.org/earlystopping, hämtad: 2018-05-02.

[52] *Lidar Lite v3 Operation Manual and Technical Specifications*, New Taipei City, Taiwan: Garmin, 2016. [Online]. Tillgänglig: http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf, hämtad: 2018-04-13.

[53] Natursidan, "Drönare – nya regler och så påverkar de naturen,"2017. [Online]. Tillgänglig: http://www.natursidan.se/nyheter/trenden-med-dronare-regler-och-paverkan-pa-naturen/, hämtad: 2018-02-09.

[54] *L78: Positive voltage regulator ICs*: ST, 2016. [Online]. Tillgänglig: http://www.st.com/resource/en/datasheet/cd00000444.pdf, hämtad: 2018-02-15.

[55] Institutionen för Energi och miljö, *Elteknik*, Göteborg, Sverige: teknologtryck Chalmers, u.d.

[56] *LM2676 SIMPLE SWITCHER® High Efficiency 3-A Step-Down Voltage Regulator*, Dallas, TX, USA: Texas Instruments, 2016. [Online]. Tillgänglig: http://www.ti.com/lit/ds/symlink/lm2676.pdf, hämtad: 2018-02-15.

[57] Okänd, "E-series of preferred numbers,"i *Wikipedia*. [Online]. Tillgänglig:

https://en.wikipedia.org/wiki/E-series_of_preferred_numbers, hämtad: 2018-02-15.

[58] Institutionen för elektroteknik, Avdelningen för system- och reglerteknik. ”Labb PM Reglerteknik E/Z/TM, Inlämningsuppgift 3,” opublicerat.

Bilaga A

Appendix

A.1 Lyftkraft och Flygtid

Valet av komponenter har stor betydelse för vilken lyftkraft som drönaren kommer få samt hur lång dess flygtid blir. Dessa står även i relation till varandra. Större motorer och propellrar drar ofta mer ström och är också tyngre, vilket minskar flygtiden. Samtidigt ger batterier med större kapacitet längre flygtid, men ökad vikt. Dessutom påverkas motorns lyftkraft av batterispänningen, vilken också påverkar vikten hos batteriet. Eftersom alla dessa aspekter har inverkan vid val av komponenter är det viktigt att göra en uppskattning av behoven och prioritera mellan flygtid och lyftkraft därefter.

Vid projektets start uppskattades drönarens slutvikt till 2041 g, se Tabell A.1. För att följa den allmänna rekommendationen där lyftkraften är dubbelt så stor som totalvikten behöver varje motor kunna lyfta cirka 1000 g vid full gas. Detta var dock inte möjligt med den ram som fanns eftersom tillräckligt starka motorer inte kunde monteras på den. För att hantera detta behövdes istället en kompromiss mellan rekommendationen, flygtiden och lyftkraften göras där lyftkraften prioriterades över flygtiden. Därför valdes istället ett batteri med lägre kapacitet på 3300 mAh och en, enligt rekommendationen, lägre lyftkraft på 3760 g samt ett förhållande mellan lyftkraft och totalvikt på $1,79$ istället för 2.

Tabell A.1: *Uppmätta vikter för komponenterna i projektet.*

Komponent	Vikt [g]
Ram och varvtalsregulatorer	623
Motorer	220
Batteri	366
Raspberry Pi	42
Ultraljudssensorer	54
FlowSensor	19
Värmekamera	19
Servomotor	12
Flight controller	31
Kraft-PCB	25
Släckspray	500
Sladdar	30
Buffert	150
Totalvikt	**2041**

Flygtiden är direkt proportionell mot hur stor ström som motorerna drar under flygning, och därigenom proportionell mot flygningens karakteristik. För att kunna uppskatta flygtiden och säkra att det fanns tillräcklig tid för att kunna göra meningsfulla tester, gjordes en uppskattning av flygtidens max- och minvärden enligt Ekvation A.1 där $C_{kapacitet}$ är kapaciteten hos batteriet mätt i amperetimmar. Flygtidens maxvärde beräknas utifrån urladdningsströmmens maxvärde medan minvärdet använder den urladdningsström som krävs för att drönaren precis ska lätta från marken. Uppskattningen antar att motorströmmen är approximativt linjär i det relevanta intervallet på mellan 50% och 100% gas, något som stöds av databladet. De uppskattade värdena för flygtidens max- respektive minvärde är $12,78\,min$ respektive $5,11\,min$, vilket ger ett medelvärde på $8,9\,min$.

$$T_{flygtid} = \frac{C_{kapacitet}}{I_{motor} + I_{komponenter}} \tag{A.1}$$

A.2 Strömförsörjning

För att Raspberry Pi:n, kameror och sensorer ska fungera behöver de kopplas till en spänningskälla. Som Tabell A.2 visar behöver samtliga en spänning på $5\,V$. Den totala strömförbrukningen har ett maxvärde som kan ligga över $1500\,mA$. Drönarens batteri levererar en spänning på $14,8\,V$ och klarar att leverera en kontinuerlig ström på $82,5\,A$, se avsnitt 2.1. För att flyga krävs en ström på max $40\,A$, vilket betyder att batteriet klarar att leverera den ström som behövs för att driva Raspberry Pi:n, kamerorna och sensorerna.

Tabell A.2: *Strömbelasningar hos olika komponenter.*

Komponent	Spänning [V]	Ström [mA] (min)	Ström [mA] (max)
Raspberry Pi	5	700	1000 [33]
Servomotor	5	0	300
Flow Sensor	5	115	115 [23]
Ultraljudsensor	5	$3 \cdot 30$	$3 \cdot 35$ [20]
IR-Kamera	5	~ 5	~ 5 [25]
Totalt	**5**	**910**	**1525**

För att få en stabil spänning på $5V$ krävs en spänningsregulator. För att reglera spänningen kan man använda en linjär spänningsregulator. Ett exempel på en sådan är komponenten LM7805 [54]. Nackdelen med den linjära spänningsregulatorn är att spänningsskillnaden mellan batteri och utgång bidrar till en effektförlust i form av värme. Effektförlusten P_F beräknas enligt Ekvation A.2. Komponenten själv klarar en värmeutveckling P_D som beräknas enligt Ekvation A.3 till $2,1\,W$. T_{jmax} är komponentens maximala temperatur och θ_{JA} motsvarar den termiska resistansen mellan komponent och omgivning. Omgivningstemperaturen T_A antas vara $20°C$. En värmeutveckling på $15,0$ W skulle därför medföra att komponenten behöver en kylfläns. Verkningsgraden hos komponenten i denna arbetspunkt beräknas enligt Ekvation A.4 till $33,8\%$.

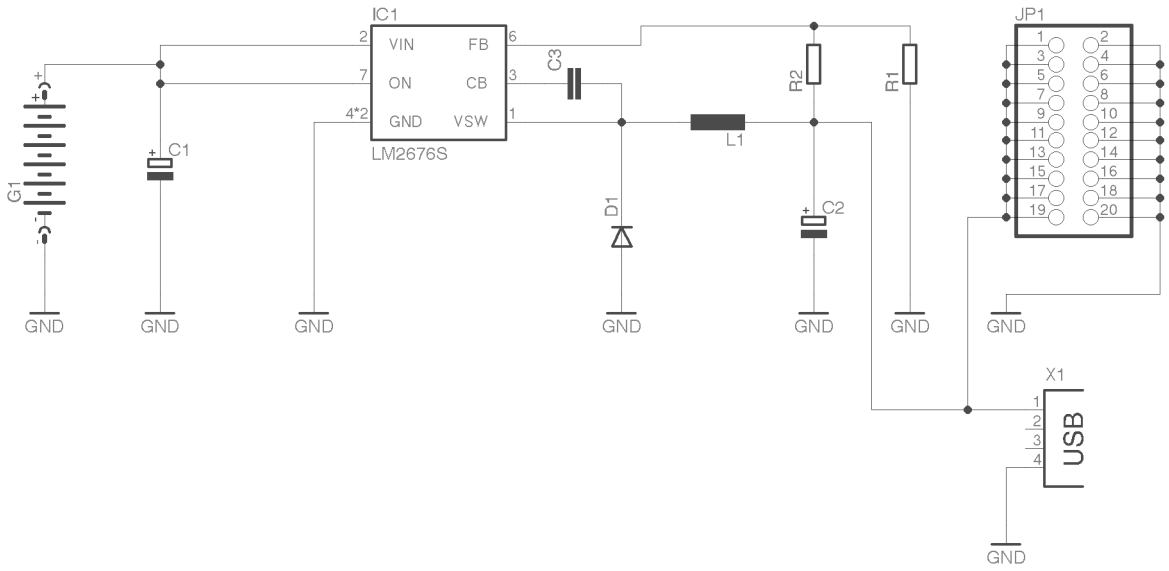
$$P_F = (V_{in} - V_u t) * I_{Last} = (14.8V - 5V) * 1.5\,A \approx 15\,W \tag{A.2}$$

$$P_D = \frac{T_{jmax} - T_A}{\theta_{JA}} = \frac{125°C - 20°C}{50\frac{C}{W}} \approx 2,1\,W \tag{A.3}$$

$$\eta = \frac{P_{Ut}}{P_{In}} = \frac{V_{Ut} \cdot I_{Last}}{V_{In} \cdot I_{Last}} = \frac{V_{Ut}}{V_{In}} = \frac{14,8\,V}{5\,V} \approx 33,8\% \tag{A.4}$$

Eftersom kylflänsar både tar plats och väger mycket jämfört med komponenten samt att den linjära regulatorn har låg verkningsgrad väljs istället en LS-omriktare, även kallad buck-converter. LS-omriktare fungerar genom att utspänningen bestäms av om en transistor leder eller inte. När transistorn leder, är utspänningen den samma som inspänningen och när transistorn inte leder är utspänningen noll. AC-komponenten av denna utsignal filtreras sedan bort med hjälp av en spole och en kondensator och kvar finns en DC-spänning som är medelvärdet av utsignalen. DC-spänningens nivå bestäms av en PWM-signal som styr transistorn och är proportionell mot denna signals duty-cycle [55]. Denna duty-cycle kan bestämmas genom att mäta och därefter reglera utspänningen till önskad nivå.

Ett liknande kontrollsystem finns implementerat i kretsen LM2676 [56]. Komponenten har en variabel utspänning som styrs av två externa återkopplingsresistorer. Med hjälp av förslagsapplikationen i dess datablad kan ett kretsschema skapas som presenteras i Figur A.1.



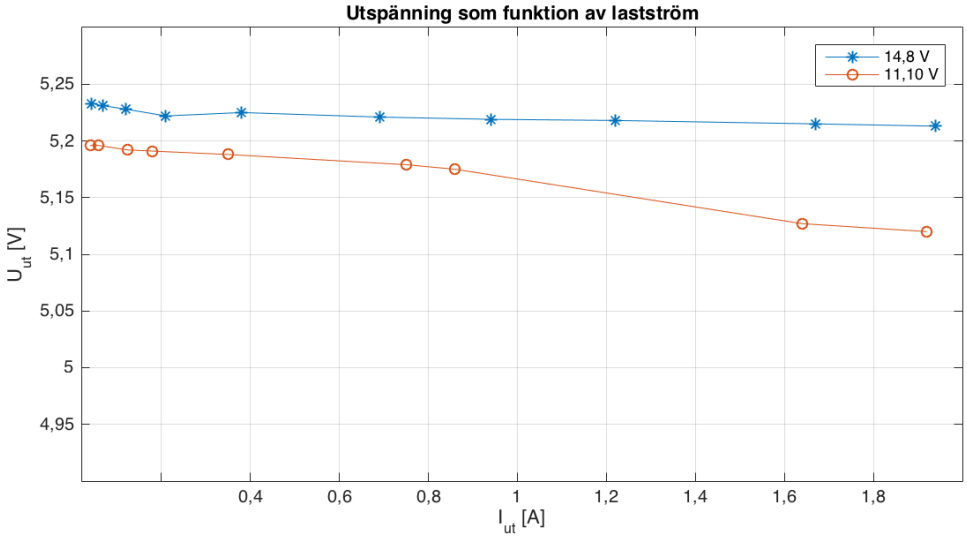
Figur A.1: *Kretsschema för strömförsörjningen.*

I figur A.1 motsvarar G1 drönarens batteri. C_1 och C_2 är elektrolytkondensatorer som används för att filtrera bort rippel i ut- respektive inspänningen. C_3 är en bootstrap-kondensator [56]. Induktansens värde är 33 μH. Dioden är till för att leda ström från spolen genom lasten när transistorn ej leder. J_{PI} är två stiftlister som passar för koppling till olika sensorer och servomotorn. X_1 är en USB-A kontakt som används för att koppla in Raspberry Pi:n. Resistorerna R_1 och R_2 används för att bestämma vilken utspänning som ska regleras. R_1 rekommenderas att vara 1 $k\Omega$ [56]. R_2 väljs sedan enligt Ekvation A.5, där V_{FB} är $1,21\,V$ [56]. Närmsta värde i E12 serien är $3,3\,kOhm$ [57]. Den väntade utspänningen med denna resistor beräknas till $5,2\,V$ i Ekvation A.6.

$$R_2 = R_1 \cdot (\frac{V_{Ut}}{V_{FB}} - 1) = 1k\Omega \cdot (\frac{5V}{1,21V} - 1) \approx 3,13k\Omega [56] \tag{A.5}$$

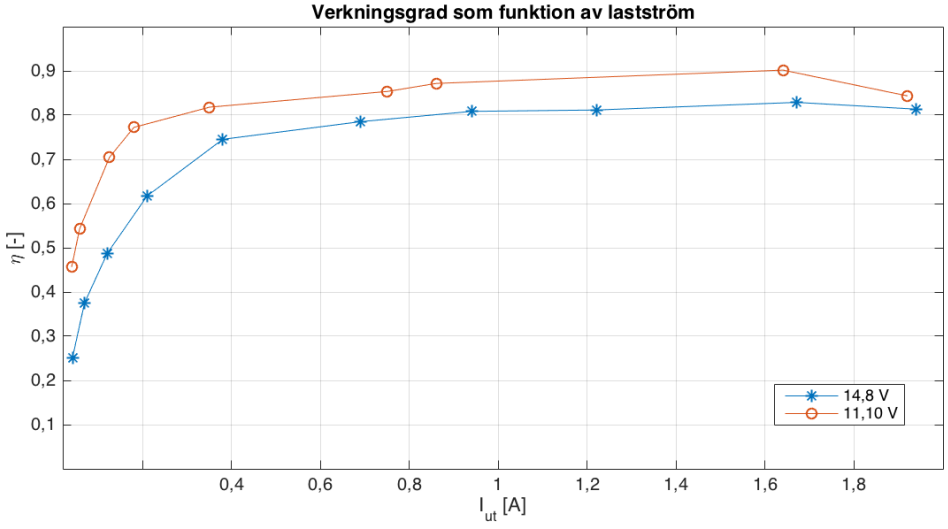
$$V_{Ut} = V_{FB} \cdot (1 + \frac{R_2}{R_1}) = 1,21 \cdot (1 + \frac{3.3k\Omega}{1k\Omega}) \approx 5,20V [56] \tag{A.6}$$

Den viktigaste egenskapen hos systemet för strömförsörjning är att utspänningen är stabil på $5\,V$ även vid maxlast. Därför genomfördes en spänningsmätning där lasten varierades, för två olika batterispänningar. Resultatet visas i Figur A.2. Även om spänningen sjunker något vid ökad lastström, håller den sig med marginal över $5\,V$. En anledning till den sjunkande spänningen skulle kunna vara ökade spänningsfall i kablar eller ledningar på kretskortet.



Figur A.2: *Utspänningens värde beroende på lastströmmen vid två olika batterispänningar.*

En mätning av verkningsgraden som visas i Figur A.3 visar att systemet fungerar mer effektivt vid höga laster. Detta beror förmodligen på att den effekt som krävs för att driva LM2676 blir mer och mer försumbar vid ökad last. Vid maxlast är verkningsgraden drygt 80%, detta kan jämföras med den linjära spänningsregulatorn som har en verkningsgrad på ca 30%.



Figur A.3: *Verkningsgradens värde beroende på lastströmmen vid två olika batterispänningar.*

A.3 Routh-Hurwitz stabilitetskriterium

Routh-Hurwitz stabilitetskriterie används för att visa att den karaktäristiska ekvationen för ett linjärt, tidsinvariant system har negativa rötter [34]. Den kan alltså användas för att se om ett system är stabilt eller inte.

Om ett systems karaktäristiska ekvation är ett polynom $P(s)$ på formen

$$P(s) = a_0 s^n + a_1 s^{n-1} + a_2 s^{n-2} + a_3 s^{n-3} + ... = 0,$$

måste samtliga koefficienter a_i vara strikt positiva. Även alla element i den första kolumnen i följande tabell, se Tabell A.4, måste vara strikt positiva [34].

s^n	a_0	a_2	a_4	$a_6 \ldots$	$c_0 = \frac{a_1 a_2 - a_3 a_0}{a_1}$
s^{n-1}	a_1	a_3	a_5	$a_7 \ldots$	$c_1 = \frac{a_1 a_4 - a_5 a_0}{a_1}$
s^{n-2}	c_0	c_1	$c_2 \ldots$		$d_0 = \frac{c_0 a_3 - c_1 a_1}{c_0}$
s^{n-3}	d_0	d_1	$d_2 \ldots$		.
.	.				.
.	.				.
s^0					

Figur A.4: *Tabell över de koefficienter som används för att ta fram stabilitetskriterier* [34].

A.4 Feature Engineering

Feature Engineering är en process inom maskininlärning där domänkunskap om datan används för att definiera *features*, det vill säga gemensamma egenskaper för all data som ska analyseras. Dessa egenskaper görs sedan kända för inlärningsalgoritmen och används under utförandet av uppgiften. Ett exempel är transkription av tal, där olika *features* av en uttalad mening kan vara sammanhängande ljud, som sedan delas in i fonem, följt av kombineras till ord och som därefter tillsammans formar transkriptionen. Ett annat exempel är bildidentifiering av en katt, där *features* kan vara päls, svans och morrhår.

A.5 Källkod

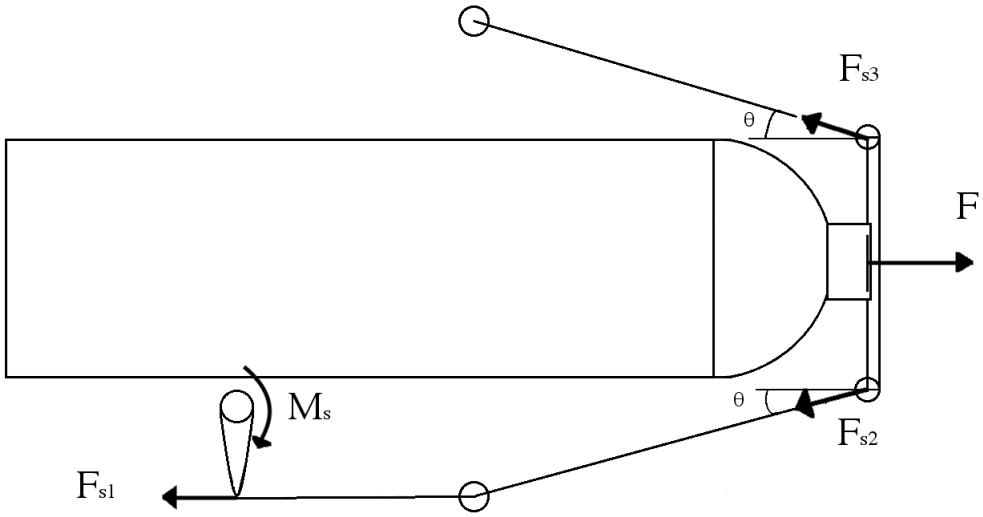
Den kompletta källkoden för att köra drönaren och laptopen finns att hämta på https://bitbucket.org/iliefski/firedrone/src. Filerna som börjar med *comp_* används på laptopen tillsammans med *tensorflow* [49] och *keras* [48]. Laptopens program startas genom

att köra filen *comp_starter.py* med Python 3 på en dator med windows som operativsystem.

Resterande kod används på Raspberry Pi:n och startas genom att köra filen *rpi_receive.py* i Python 3. Ett antal bibliotek från Adafruit har inkluderats under MIT-licence för att köra värmekameran. Testfiler för de individuella sensorsystemen finns också inkluderade.

A.6 Kraftanalys släckanordning

För att ta reda på vilket kraft som kan appliceras på munstycket friläggs munstycket tillsammans med den fastlimmade plattan. Antag att krafter i z-led inte påverkar konstruktionen. På så vis kan följande tvådimensionella kraftschema ritas.



Figur A.5: *Kraftdiagram för släckanordning.*

Först beräknas med vilket kraft F_{s1} servomotorn drar i snöret i Ekvation A.7. M_s är det moment som servomotorn kan ge och a är snörets hävarm till motors rotationsaxel.

$$M_s = F_{s1} \times a \implies |F_s| = \frac{M_s}{a}, |F_s| = |F_{s1}| \tag{A.7}$$

Antag vidare att snörets böjning mot drönarens ben är friktionsfritt. På så vis har F_{s2} samma amplitud som F_{s1}. Nedan visas samtliga krafter som påverkar plattan.

$$\begin{aligned} F_{s2} &= |F_s|(\sin\theta\hat{x} - \cos\theta\hat{y}) \\ F_{s3} &= |F_{s3}|(-\sin\theta\hat{x} - \cos\theta\hat{y}) \\ F &= |F|\hat{y} \end{aligned} \tag{A.8}$$

Enligt Newtons andra lag sätts summan av alla krafter till noll. Detta leder till att F kan räknas fram enligt Ekvation A.9. Servomotorn har ett vridmoment på $1,6\,kgcm$ [17] och hävarmen a är cirka $0,7\,cm$. Vinkeln θ mäts till 20°.

$$\begin{aligned} F + F_{s2} + F_{s3} = \vec{0} &\iff \\ |F| - |F_s|\cos\theta - |F_{s3}|\cos\theta = 0 & \\ |F_s|\sin\theta - |F_{s3}|\sin\theta = 0 &\iff \\ |F_s| = |F_{s3}| & \\ |F| = 2|F_s|\cos\theta = 2\frac{M_s}{a}\cos\theta = 2\cdot\frac{1.6kgcm}{1cm}\cos 20^\circ \approx 4,3N & \end{aligned} \tag{A.9}$$

A.7 Diskretisering av regulatorfunktion

Då reglersystem ska implementeras på Raspberry Pi måste regulatorfunktionen diskretiseras. Detta görs genom att skapa en differensekvation som approximerar differentialekvationen. Styrsignalen kan beskrivas som summan av de tre olika regulatordelarna P, I och D. P beräknas på samma sätt som i det tidskontinuerliga fallet som $P = K_p e(k)$ [58]. Integraldelen kan i det kontinuerliga fallet uttryckas enligt Ekvation A.10.

$$I(t) = K_i \int_0^t e(\tau)d\tau \tag{A.10}$$

Euler framåt-metoden kan nu användas för att approximera denna integral [58]. Resultatet för detta visas i Ekvation A.11.

$$I(k+1) = I(k) + K_i h e(k)[58] \tag{A.11}$$

Derivatadelen kan beskrivas enligt

$$T_f\frac{dD(t)}{dt} + D(t) = K_d\frac{de(t)}{dt}[58]. \tag{A.12}$$

Med hjälp av Euler bakåt-metoden approximeras denna differentialekvation och ger resultatet

$$D(k) = \frac{T_f}{T_f + h}D(k-1) + \frac{K_d}{T_f + h}(e(k) - e(k-1))[58]. \tag{A.13}$$

A.8 Dropout

Dropout är en metod för att undvika överanpassning av neurala nätverk tränade med djupinlärning och innebär att neuroner och deras vikter slumpmässigt ignoreras under träningsiterationer. Detta minskar beroende mellan neuroner och gör att varje neuron får utrymme samt blir tvunget att anpassas att skicka mer och mer korrekta signaler genom nätverket.[42]

A.9 Testdata från testet av eld- och värmeidentifiering

Tabell A.3 nedan visar ett urval av data från testet av eld- och värmeidentifiering, i form av all data kring de högsta, uppmätta temperaturerna i varje testfall.

Tabell A.3: *Insamlad data från testet av värmeidentifiering.*

Avstånd	Kolumn, rad	Temperatur	Var elden är
1	2,5	34,25	(245,366)
1	3,5	34<	(286,324)
1	3,5	31>	(361,329)
1	3,4	33,5>	(355,264)
1	3,4	28,75<	(278,248)
1	o1,4	27,25	(155,214)
1	1,6	37	(132,462)
1	2,6	35,5	(249,438)
1	4,6	29,5	(372,479) (elden utanför bild)
1	3,6	31,25	(355,444)
1.5	-	ogiltig data	-
1,5	-	ogiltig data	-
1,5	-	ogiltig data	-
1,5	-	-	ser inte elden
1,5	4,5	27,75	(434,294)
1,5	3,6	28,25	(352,415)
1,5	3,6	31	(434,356)
1,5	3,5	31	(307,315)
1,5	3,5	28,5	(346,252)
1,5	4,4	25,75	(413,212)
0,5	0,5	42,5	(30,407)
0,5	3,5	44,25	(356,381)
0,5	5,5	39	(467,4)
0,5	4,5	44,5	(394,317)
0,5	2,5	36	(268,321)
0,5	1,4	42,75	(145,274)
0,5	1,4	33,75	(159,244)
0,5	3,4	32,5	(323,232)
0,5	5,5	33,5	(463,342)
0,5	4,7	48,25	(372,utanför)

Avståndsvärdena i ovanstående tabell anges i meter, kolumnnummer och radnummer räknas från noll, temperaturen anges i grader Celcius och var elden är anges i pixlar. ”o”i tabellen står för otydlig data, ”<” och ”>” betyder att ungefär samma värde även finns direkt till vänster respektive höger i rutnätet med värmedata som värmekameran genererat.

A.10 Reinforcement Learning

Reinforcement learning är baserat på nätverksträning där resultat snarare än handling utgör grunden för ändringar i nätverket. Det finns alltså ingen optimering baserat på den utdata som nätverket ger utan endast resultatet som utdatan resulterar i. Detta medför även att träning ofta sker direkt i nätverkets domänmiljö till skillnad från till exempel deep learning där träning sker på ett fördefinierat dataset och i en separat process. Detta gör att nätverket själv till stor del tar reda på vilka faktorer som leder till de resultat som belönas.

En avvägning som görs under processen av reinforcement learning är hur mycket vikt man ska lägga på att utnyttja det som nätverket tidigare lärt sig fungerar för att få belöning och hur mycket nya saker, dvs utforskning, det ska ägna sig åt.

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2018