KTH Mechatronics Advanced Course
MF2059, HT 2017
FINAL REPORT

Firefighting Assistance Using a UAV

PYROTHAN

ANTHONY CLERC, DANIELA ATTALLA
DAN NGUYEN, GUILLAUME MEURISSE
ISAC TÖRNBERG, JACOB EKESUND
MYLES SYVERUD, ROBIN MORENO RINDING
SOFIA NAVARRO HEREDIA

Cybercom Group
ERIK BERGDAHL, FREDRIK EDLUND
JITIN THOMAS, GABRIEL ANDERSSON-SANTIAGO

December 23, 2017

ABSTRACT

Unmanned Aerial Vehicle for Fire-fighting Applications

Over the past decade, the sector for Unmanned Aerial Vehicles (UAV) has been rapidly growing. The advancements in the technology along with an increase in supply have provided society with affordable yet impressively capable devices for a variety of applications. If the trend carries on, the drone market will continue to find new applications. One of the most promising applications of drones is to assist rescue forces in saving wounded people during an emergency since they can explore places deemed too dangerous for human-beings.

In the firefighting world, there are numerous scenarios where a UAV may provide a service not previously feasible. Based on the inputs of various subject matter experts, this report explores one possible solution of how a UAV could benefit a firefighting team in critical moments. This includes a *state of the art* analysis to describe the current UAV solutions being used today as well as a full analysis of a prototype that was designed and tested for such a purpose.

The report demonstrates how the prototype, although not sufficient for use in action, was developed enough to achieve a few critical milestones necessary for such a drone including: integrating large amounts of sensor data to sense the interior of a common building, displaying the data in a user-friendly format, and lastly providing basic obstacle avoidance to aid a pilot. All of these features were carried out on a prototype custom designed and fabricated for this purpose.

Acknowledgements

The team would like to acknowledge the following people for giving us valuable feedback and supervision during the project:

Cybercom Group:

- Erik Bergdahl
- Fredrik Edlund
- Jitin Thomas
- Gabriel Andersson-Santiago

Royal Institute of Technology (KTH):

- Elias Flening
- Björn Möller
- Staffan Qvarnström
- Vicky Derbyshire
- Sofia Cassel

We would also like to acknowledge the following people for giving us their insights on difficulties that fire fighters are currently facing:

Swedish Civil Contingencies Agency (MSB):

- Lasse Nelson
- Stefan Haggö

University of Lund:

- Stefan Svensson

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Description
UAV	Unmanned Aeriel Vehicle
CPU	Central Processing Unit
RAM	Random Area Memory
ECU	Electronic Control Unit
ESC	Electronic speed controller
IMU	Inertial Measurement Unit
GPS	Global Position System
IR	Infrared
PWM	Pulse Width Modulation
PPM	Pulse Position Modulation
PCM	Pulse Code
SBUS	Serial Bus
DSMX	Digital System Multiplexer
RC	Radio Control
FPV	First Person View
FOV	Field of View
HD	High Definition
GRASP	General Robotics, Automation, Sensing and Perception
MAVlink	Micro Air Vehicle Link
OS	Operating System
RTOS	Real-Time Operating System
MCU	Microcontroller Unit
MMU	Memory Management Unit
SWOT	Strengths, Weaknesses, Opportunities, and Threats
LIDAR	Light Imaging, Detection, And Ranging
MSB	Myndigheten för samhällsskydd och beredskap (Swedish Civil Contingencies Agency)
HUD	Heads Up Display
GCS	Ground Control Station
UART	Universal asynchronous receiver-transmitter
GUI	Graphical User Interface
PCB	Printed Circuit Board
I^2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
CSI	Camera Serial Interface
BEC	Battery Eliminator Circuit
FLIR	Forward-Looking Infrared (Thermal Imaging Company, FLIR systems)
SITL	Software In The Loop

Chapter 1

Introduction

1.1 Background

Firefighting crews respond to a variety of calls that can include medical situations, fires, chemical spills, traffic accidents, and search and rescue scenarios. In order to maximize their success, teams have to respond in relatively short time frames and on limited information. Consequently, preparedness and the rapid gathering of information are essential to handle what is commonly a combination of the aforementioned problem areas in the limited time given to them.

Due to the life-saving nature this service provides society, any improvements that can be made to their methods would be an obvious benefit to all. The tools and technology available to these teams are not only essential for their work, but are also continuously evolving due to product demand in other areas such as consumer products.

One product that has recently begun to penetrate both commercial and governmental services are unmanned aerial vehicles (UAVs). Due the relatively low cost and large amount of remotely acquired data these units gather, their application in a typical firefighting scenario is an obvious match.

Sponsoring Organization Motivation The IT and embedded system consulting company Cybercom Group AB saw the opportunity for drones in this field and decided to further investigate the feasibility. A pre-study was conducted to confirm there was a need for a drone as an assistant for firefighters and their perspective on the role of the drone.

1.2 Project Description

The drone in this report was mainly developed to fit firefighters' needs for a safe way of acquiring information as first-responders to a typical fire situation. As explained before, there is a will for a quick reconnaissance before entering blindly into a building on fire to avoid all kinds of human loss or injuries. Therefore, the drone must be easy to transport in a firefighter-truck and setting it up must be quick. Hence, it shall be ready to take-off in less than 1 minute ideally. Indeed, during such critical situation, every minute counts in order to save lives. As the drone will operate in a harsh environment, it must be robust and capable of dealing with unexpected events. The drone should be easy to pilot and should be equipped with a form of intelligence to assist the pilot during his/her mission. Last but not least, the drone must include a vision system so that the user can see directly through the drone's eyes. As the environment might be smoky and fuzzy, the drone shall still be able to provide some valuable information to the firefighters in that case, hence the use of a thermal camera to spot hot zones.

1.3 Requirements, Delimitations & Risk Assessment

1.3.1 Requirements

The requirements and features defined by the company are that the drone shall:

- Withstand loss of one electrical motor
- Cope with wind speed up to $10\ m/s$
- Have a turnaround time of maximum 5 minutes with replacement of vital equipment
- Have a turnaround time of maximum 2 minutes with replacement of batteries
- Withstand bumping into objects
- Produce a steady video stream from cameras
- Be modular and ”hot swappable”

In addition to designing for these requirements, the end-users. namely the firefighting community were asked their opinions in order to refine the company requirements and possibly generate additional features that may have been missed. This additional feedback resulted in several more requirements that the drone shall:

- Have a flight time of 15 minutes
- Handle ambient temperature of 100°C for 5 minutes
- Be rapidly deployed, sub 1 minute.
- Have obstacle detection
- Include pilot assistance to navigate

1.3.2 Delimitations

Ideally, all the requirements would have been achieved and built upon. However due to the time and budget constraints, not all could be achieved.

The prototype’s aim is to help a team in an environment that is extraordinarily complex and harsh. The complexity of the unknown environment is a key aspect of the problem. For this project the expected working environment was dramatically simplified. For instance, there may be falling objects or/and strong wind currents that were not accounted for.

Another major simplification was the size of the drone. Initially, the drone had to be able to fly into a building, but at its current size, navigating through a door frame would be near impossible.

Fire resistance, another key aspect of the product was only lightly touched on. The basic prototype’s foam shell was tested for insulating properties, but a full analysis of the drone in a high temperature environment was not performed.

Lastly, to make product truly hot-swappable, extensive mechanical design would have to be done. Since this project was bound to the prototype phase, this requirement was set to a low priority.

1.3.3 Risk assessment

From early on, there were many high-risk challenges that the team had to overcome. To start, none of the team members had experience with drones prior to the project. This meant that the first weeks were dedicated to understanding the fundamentals of drones and the tools used to design and operate them.

The full list of requirements was also a tall order that the the team recognized as being probably unobtainable to complete, but an exciting challenge.

Another major area of risk was the time and financial budget for the project. To achieve the performance characteristics that a final product would ideally have, a much larger budget for components would be required. For example, components the team purchased were largely under the ”commercial grade” class, whereas for high temperature scenarios, components in the ”military grade” are recommended.

Lastly, truly testing the prototype in a real emergency situation was impossible due to a lack of available resources. The team understands that this step is critical to perform the true validation needed, but due to the early stages of the product, this was not achievable within the given time.

1.4 Readers Guide / Report Disposition

The first chapter gives an introduction about the project, focusing on why the development and construction of a firefighting drone is necessary and who is the sponsor of the project. Next the project and its features are described, as well as the requirements defined by the company, the delimitations and a risk assessment based on the time, budget, knowledge and technology available.

The second chapter depicts the literature review and the State Of The Art (SOTA) analysis carried out to get a rough idea about the available firefighting drones on the market. In addition, a SOTA about obstacle avoidance technologies and how to implement them is explained, and finally a section about drones in high temperature scenarios aims to research how to handle the temperature requirements.

Chapter number three is the core of the project. Here the methods and the implementation are explained. This part starts with a wide description about the team's organization and the project management. Within the next section, the reader will have an idea concerning the system architecture and software in general (operating system, communication, computers, interfaces, protocols, etc.). The following section describes which sensors were purchased and why, how they were placed in the frame, and the program code and sensor fusion algorithm carried out to assist the pilot in harsh environments. Section 3.4 gives the details of the obstacle avoidance implementation regarding the sensor fusion, and 3.5 explains the Graphical User Interface (GUI) the drone and the pilot use to visually communicate. Right after, the method used to control the flight in terms of hardware (Pixhawk, motors, ESC, etc.) and firmware and calibration (PX4, Ardupilot, QGroundControl...) is depicted. The next section contains the Pixhawk-ROS communication, and the simulation tools are explained in the next passage. Section 3.9 illustrates the overall design of the drone and how modularity, hot-swap design and heat resistance have been conceived. Finally, the electronic architecture and its working is described.

The fourth chapter clarifies how the requirements have been verified and tested. This section begins with a technical explanation about flight performance and sensors calibration and verification. Next, the obstacle avoidance verification test is exposed. A thermal resistance section and the modularity tests close this chapter.

Chapter five gives an overview of the project results and explicates if the final product fulfill the requirements in terms of flight performance, obstacle avoidance, thermal considerations and modularity.

The last two chapters of the Technical Review contain the discussions and conclusions of the project, and the future work coming students can continue with respectively.

Chapter 2

State of the Art Analysis

This chapter summarizes the literature review and the State Of The Art (SOTA) the group has performed in order to get a deep knowledge concerning current firefighter drones on the market and their applications, up to date obstacle avoidance algorithms, and drone's performance in high temperature scenarios.

2.1 Firefighter Drones on the Market

There are various groups currently using UAVs for firefighting applications. This section outlines different existing solutions in the field today. The investigation showed that most drones available today are intended for outdoor environments and are basically commercial drones with some added devices (IR camera, live video stream, etc.) to make them helpful when firefighting. Swedish Emergency Services currently use or are planning to use drones for outdoor environments.

Considering the project requirements, the drone should be able to fly in large indoor environments like warehouses, which are more difficult to navigate than an outdoor context as there might be a considerable amount of obstacles to avoid. Different sensors were researched and added to the drone to incorporate a semi-autonomous obstacle avoidance feature.

FAROS (Fireproof Aerial Robot System)
FAROS (Figure 2.1) is a fire-resistant UAV developed by KAIST (Korea Advanced Institute of Science and Technology) to help keep everyone safe in situations like high-rise fires [1]. It is intended to explore inside buildings before firefighters enter to minimize risks by giving information of the fire situation. Although it is designed to fly using a quadrotor system, it can change to spider's crawling on walls when things heat up or an obstacle is noticed. FAROS can detect fires, search smoky building interiors and transfer real-time data from a fire to the ground station.

FAROS can autonomously fly down hallways, guided by a 2D laser scanner, an altimeter and an IMU containing accelerometers and gyroscopes which help it to estimate its location in the building and navigate independently. A thermal imaging camera with a dedicated image-processing system spots humans inside the building and allows the drone to find fire-ignition points.

Figure 2.1: FAROS drone [1]

DJI Phantom 3 Professional

Companies like FLIR Systems and Workswell do not produce their own brands of drones, but they offer complete imaging solutions for firefighters relying instead on those already available on the market, such as DJI's Phantom [2]. The DJI Phanton 3 Professional (Figure 2.2) is a drone with adequate features for firefighting. With those features as GPS-assisted hover, vision positioning system, automatic flight logs, intelligent battery and an unmatched propulsion, the drone processes information and completes complex calculations in real time. The flying machine also comes with a return to home command and a camera with a point of interest function that allows the operator to select a location or object to focus on.

Figure 2.2: DJI Phantom 3 Professional drone [2]

New York City's Fire Department

The New York City's firefighting arsenal already includes drones (Figure 2.3). It captures both standard video and infrared images, delivering high-definition images in real time to commanders [3]. The Fire Department realized the benefits of a drone after a gas explosion in which eight people died. An amateur drone operator sent out his device and captured images that gave a view of the scene.

Figure 2.3: New York City's firefighter drone [3]

Swedish Emergency Services' current use of UAVs
UAVs have just recently become considered as a useful tool within Swedish emergency services. Several emergency services are currently using or are planning to use UAVs in the near future to, for example, find forest fires, give a better overview of a situation, determine limitation-lines by analyzing the fire, to reduce risks for personnel and documentation [4]. The emergency service in Västervik has had a quadrocopter in use since 2014, seen in Figure 2.4. The UAV uses a video camera and an IR-camera, stabilized on a gimbal, that produces a live video stream on a screen monitored by the pilot [7].

Figure 2.4: Quadrocopter used by the emergency service in Västervik [4]

Pitchup AB, based in Mölndal Sweden, is the provider of UAVs to the emergency services in Västervik, Stenungsund, Syd and Källvik. The company offers its self-made UAV, the Explorian, developed to use DJI modules and payload. The Explorian is the UAV currently used by the emergency service in Västervik and Stenungsund. Pitchup also offers the DJI phantom 4, Matrice 200 and 210 as well as the Swedish-made Xrange 3G Industrial UAV[8].

2.2 Obstacle Avoidance SOTA

There is a considerable number of papers related to obstacle avoidance implementation regarding distance measurement of diverse sensors. In [9], obstacle detection and collision avoidance for a UAV with low-cost sensors is studied. Ultrasonic and infrared range finders data (same as in Pyrothan, see section 3.3) are gathered together with optical flow sensors and inertial measurements, and afterwards inputed to the situation awareness

module, whose task is to determine the minimum distances available and the velocities. Beforehand, a weighted filter combines the sensors' values with regards to their reliability. Next, the minimum distances available and the velocities are used to determine the situation assessment, which distinguishes within three areas - free area, near area, danger area - depending on the object's distance. This information is used for the distance control to keep a minimal distance from an obstacle.

Another paper [10] from similar authors presents a simpler approach for obstacle detection and collision avoidance using only ultrasonic sensors and simple data fusion. In this case, redundant sensors data is fused with IMU information and processed into the obstacle detection module. The collision avoidance module distinguishes aswell within the same three different areas depending on the measured distance and activates a PID controller to prevent further proximity to the object. The main problem of this implementation is that the redundant ultrasonic sensors disturb each other.

The obstacle avoidance used in this project is based on [11]. This paper describes a simple 3D decision taker algorithm which overwrites the operator control commands when the drone is closer than a specified distance to the obstacle. The value of the corresponding linear and angular velocity of one or more of the axis is then set to zero depending on the state of the drone in order to avoid that the drone can continue flying towards an obstacle.

2.3 Drones in High Temperature Scenarios

In addition to developing the software, hardware concerns were investigated. After reviewing UAVs in similar environments, it was obvious that the majority of the existing solutions avoided the heat issue entirely by distancing themselves from the danger. A few specialized drones exist (see section 2.1) that are meant to sustain these temperatures.

The research focused on the obvious questions, "what temperatures can we expect to see in an inside fire?, "what temperatures can average/common components used in commercial drones withstand?" and "can we deflect the heat if needed?"

The temperatures the UAV will sense depend, not surprisingly, on the proximity to, type, and size of the fire. The environment such as the size of the room also will affect this value. Additionally, the proximity is dependent on the sensor data the end-user requires to provide value.

For the second question, the team found the majority of components that were tentatively selected were of commercial grade [12], designed only to be able to withstand up to approximately 80°C.

As for deflecting heat the team found a few consumer products that would be appropriate, but analysis beyond noting their availability was not done. As a starting point, the team settled on a simple insulation layer to shield the components.

Overall, the concept of heat rejection will be essential to this product's success, but since the scope of the project focused on the mechatronic aspects of the development, this was not a high priority to implement.

Chapter 3

Methods & Implementation

This chapter describes the methods, tools and techniques the group has used to solve the problems which came up during the project. It includes organizational relevant information and an overview of the system architecture, as well as more detailed data about the implementation of software, hardware and design decisions.

3.1 Organization

This section will discuss how the organization within the team was set up and the motivation behind it. This concludes with a reflection upon the choices that were made and what could be improved in the future for the next team that will pursue the development of the project.

From the very beginning, in order to work efficiently, the team was split into several sub-teams, with the intent that each team would specialize and be considered an *expert* in the field it was given. Therefore, one sub-team was in charge of the mechanical conception (CAD design), another one of the circuitry design, another of the software development and the last one was responsible of making sure that all the components developed separately would integrate seamlessly. Even though the teaching team suggested to regularly change positions in order to gather as much knowledge as possible in every field (remember that the final goal of the project — at least from a teaching standpoint — was to acquire skills and not especially in meeting all the requirements), it was realistically non-feasible because most tasks required a non-negligible amount of prerequisites before being able to be tackled. A relevant example is the software development that necessitates a good understanding of ROS and other development tools such as C++ or Python, the two programming languages that were used.

However, no one was entitled to work only within the boundaries of the role that (s)he was assigned to and several team members explored and worked on other fields, either motivated by their own curiosity or by a need to solve another problem in order to continue the drone's development and meet the deadlines.

Weekly meetings were organized where each team member could briefly explain what he or she had done the previous week and what he or she was going to work on the upcoming week. Problems that arose during the prior week were discussed with all team members and good solutions often resulted of this brainstorming activity. Our managerial method was based on **Scrum** whose we had to adapt slightly as we had weekly meetings instead of daily ones. The scrum master was considered, implicitly, to be our coach and he ensured that all the meetings went smoothly and that everyone could speak equally. During each meeting, someone was in charge of taking notes to summarize what was said so that Cybercom could remotely follow the evolution of the development. At the end of the meeting, a *Trello* dashboard was updated and each member was assigned some tasks to perform. Using *Trello* is a very good way to keep track of all activities being performed by each member for small teams, if everyone 'plays the game' and regularly updates what's being done. The team kept track of most relevant links and information on the *Trello* dashboard as well so that anyone could, at a gander, find what (s)he was looking for.

For the hardware development, the team decided to use *Autodesk Fusion 360* because students can currently get a free license for three years. Moreover, with its server based online development and VSC (version control), everyone can easily access the assembly being worked with, modify it, create new parts, explore old solutions tested before or whatever else (s)he wants to accomplish. It comes with the downside of requiring an internet

connection each and every time someone wants to initiate and access changes. This can be compared with other CAD software such as Solidworks or SolidEdge that do not come by default with a version control (additional expenses are required to get it) and would have complicated the workflow within the team members.

As several members of the group were designing PCBs it was up to them to decide in which software they were to design the circuit boards. Two software packages were suggested and also used for the design: *Circuit Maker* and *Autodesk Eagle*. A positive aspect about Eagle is that it cooperates well with Autodesk Fusion 360, since both CAD tools are manufactured by Autodesk, which makes it easy for one to design the circuit boards in 3D. This is of interest if one wants to take into account the different circuits in their product design. Circuit Maker was also used and could export PCBs to CAD files that were integrated into Fusion 360 seamlessly. Both solutions worked well.

For the software development, the team started using Gitlab (git repository similar in functionalities to Github but with the added advantage of keeping projects private) straightaway to save its code. It is a common tool that is being utilized nowadays in most software based companies. It permits to share code amongst all the developers and to work in parallel on the same piece of software. Without delving too much on the subject, the principle is to use several development branches where each idea can be explored independently of the others. Once an idea is successfully implemented (meaning thorough tests have been performed to assess it's behaving as it's supposed to and doesn't have too apparent flaws), the branch is merged onto the 'trunk' (commonly called 'master branch' in the jargon) and the development of new idea can be pursued afterwards by re-branching from the master branch.

The ROS framework was adopted for software development. It allows to develop individual programs (called nodes) that communicate with each others using topics and services. It's convenient because each node is made to deal with only one feature (for instance, reading one sensor's measurements or streaming the camera feed), therefore remaining simple and easily debuggable. Moreover, the work can be split as each developer can work on a single node and the developers just have to agree on the communication protocol, that is which messages need to be sent between which node. Within ROS, the nodes are always part of a bigger entity called a package which accomplishes a high-end functionality. As an example, a package could be in charge of the mapping functionality and several nodes communicating with each other and perhaps sending messages to the 'outer' world would be used to accomplish it. Defining decent size packages performing one given task is key in having a nice, well organized software architecture.

In order to integrate the ROS workflow seamlessly with git, we decided to create one individual branch per package being developed. Prior to that, we just had to define the messages and services that would be used for the nodes inter-communication and each developer could afterwards works on its own sub-task completely independently. Once the package was demonstrated to work, it was merged to the master branch and could start interacting with the other ROS nodes already present.

Throughout the process, we used the Gitlab repository as a way to access the code on various hardware (desktop, laptop, raspberry-pi,etc..) which proves useful in one hand as it's an easy mean to share the code. However, on the other hand, it led to a massive amount of useless commits, messing the cleanliness of the code history and making it difficult to follow the changes and improvements that were performed by each individual. Indeed, for convenience sake, the code was often written on a desktop or laptop and then pushed online so it could be retrieved and tested on the raspberry-pi. There are numerous reasons why the aforementioned workflow were adopted. First out of the gate, the pi could most of the time be accessed only through ssh (hence no keyboard, mouse and screen were plugged in), meaning that only terminal code editors could be used such as *Vim* or *nano* to write the code. They are quite unpractical as no mouse actions can be performed and the advanced features that are easily doable in a common IDE (such as search and rename) are heavily hidden and can only be done with a substantial amount of knowledge in those editors. Therefore, writing the code on a traditional hardware environment was convenient and time-saving. Unfortunately, most of the code couldn't be tested on a conventional laptop as it was interacting with sensors or cameras that were only connected on the physical pins of the raspberry-pi, explaining why we had to do this incessant back and forth workflow.

Creating several branches for each package proved to be a robust idea that improved dramatically the efficiency as there were less clashes between the work performed by everyone. However, we only agreed lately on defining all ROS messages and services in one package instead of defining them separately in each package as it was done before. This choice led to a massive simplification of the process as everyone could easily modify the content of a message without having to know where the descriptive file is stored. Moreover, when writing the code, there is no more need to look for the header file containing the message definition as all the files are now saved in the same directory. Note that this workflow is ideal as long as you don't want to upload your package online for others to use as this would require to upload the package containing the messages definition as well (which can be cumbersome).

It's worth mentioning that we decided to concentrate all the "to dos" (tasks) in Trello instead of split them between Trello for the hardware development and GitLab for the software part because we feel it's easier to find all pieces of information in one place instead of scattering them amongst several platforms.

Lastly, we wholeheartedly recommend assigning a specific role to each team member so (s)he can really feel entitled to the project, which is a great way to boost everyone's motivation and sense of accomplishment as well.

3.2 System architecture

The drone is composed of several sensors, actuators and computational units (see figure 3.13 and 3.14), all of them must interact in order to provide full functionality. As the project evolved, it became clear that one Raspberry Pi was not enough to handle the sensor reading, obstacle avoidance, video streaming and communication with the Pixhawk. The main issue were that the Raspberry Pi did not have enough RAM. This could either be solved by buying a new more powerful micro-controller or add a second Raspberry Pi. The choice to add a second Raspberry Pi was made because it was determined to be both cheaper and easier to implement. Robot Operating System (ROS) is used as operating system on the Raspberry Pi, whereas PixHawk is running with the open-source flight-control firmware Ardupilot. This section discussed how the distinction has been made between the different functionalities.

The first Raspberry Pi (SensorsPi) is managing the reading of all the sensors and also includes the sensor data fusion. In more detail, the different sonar sensors are allocated to different nodes but use instances of the same program. The radar has also its own node and program. In the other hand the three IRs are sharing the same node since they are connected to a single ADC before to transfer the measures. All the sensors are publishing the measured distances and a separated node is gathering the different values before to fusion the information and define the zones' level (see section 3.3). The previous information is then sent to the user interface as well as a main node (decision taker) which task is to combine the values from the sensors and the pilot to implement an obstacle avoidance before to send the flight command to the pixhawk.

On the other hand, the second Raspberry Pi (CamerasPi) is taking care of the communication, obstacle avoidance and cameras. It receives flight control inputs from the pilot through an Xbox controller connected to the ground station computer and the information from the sensors. It uses these data to implement an obstacle avoidance (see section 3.4) before to transfer the final flight commands to the Pixhawk by the use of MAVROS, an extendable MAVLink communication node. It is also receiving crucial flight information including battery status, current drawn, Pixhawks CPU temperature and compass orientation, that are then sent to the ground station and are displayed on the GUI to assist the pilot. The image processing for both cameras are also handled on this Raspberry Pi and are streamed to the ground station computer.

Finally, the ground computer is responsible to manage the GUI that runs on a "QT rosnode" and to transfer the flight command from the joystick. The user interface is used for informing the pilot of the different measures carried out by the sensors and the Pixhawk, to display the camera feeds as well as command the flight mode

(manual/autonomous), as explained in detailed in section 3.5.

In order to simplify the development in parallel of the different functionalities to attain high flexibility several ROS nodes are created. This facilitates the repartition of the computational load on different computers. Every of the so called nodes are responsible for carrying out a specific task and are run on corresponding Raspberry Pis, as aforementioned (See figure 3.1). The entire system is managed by a single core (ROScore) running on the CamerasPi that can be considered in a nutshell as a node in charge of loading the parameters, registering the nodes, topics and services. The communication between the computer is handled through wifi via SSH. The communication is possible by sharing a single roscore between different clusters.

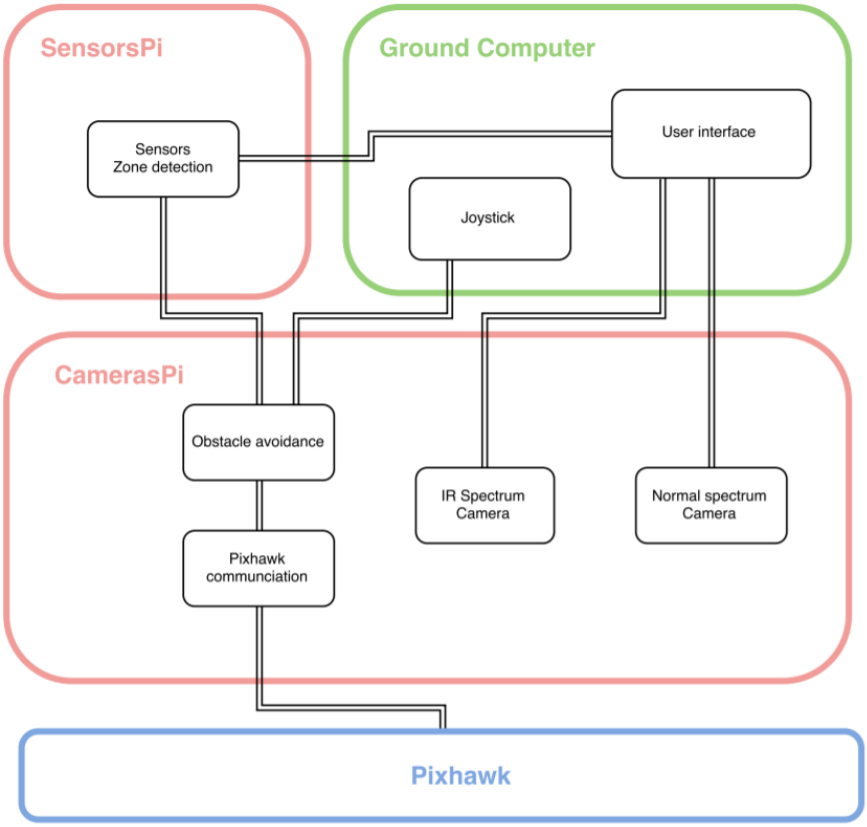


Figure 3.1: Schematic of the design architecture

3.3 Sensors

The drone's main mission is to give feedback about a catastrophe happening in a large indoor area (Warehouse, Tunnel, etc). As the flying environment might be hot, smoky, fuzzy and/or flurry, in short harsh, the drone's navigation must be assisted by an autonomous obstacle detection and avoidance system. In order to create such functionality, the drone is equipped with different sensors, combined to provide an overview of the surroundings. These pieces of information are then processed to guide the drone by overwriting the control commands sent by the pilot. The following sections present the components selection, algorithms and techniques employed in the development.

3.3.1 Sensors selection

The sensor selection is shaped by the constraints applied to the drone, as a result of the evolution in an harsh environment including high temperature, smoke, darkness, etc. Moreover, even if the pilotage is left to human control, the progression in a close area without any direct visibility is too difficult. Therefore, the drone must be able to detect obstacles around it accurately to avoid them or at least give enough feedback for obstacle awareness.

Different drones on the market [13], [2] are using stereoscopic cameras to obtain a 3D image of the surroundings. However, this is not applicable for dark or smoky environments. The choices are then directed to more simplistic detectors that rely on different technologies more robust for harsh environments as described afterwards.

Infrared sensors
Infrared (IR or laser) are accurate single point sensors efficient for measuring long range with a high update rate. Nevertheless, measurements are influenced by the refractive index of the air (changing with temperature) and reflective surfaces, which can include inaccuracy for the scope of the project.
Three long range IRs (SHARP GP2Y0A710K0F whose range is $100 - 550\ cm$) have been chosen to equip the drone. They are placed at 0°, 90° and -90° (see Figure 3.2 below). However, these IR sensors cannot handle high temperatures and are highly influenced by the environment. More expensive IR sensors, such as FLIR-MLR100 [14] should be selected to cope with extreme conditions and some heat resistant glass can be added on the frame to handle high temperatures.

Ultrasonic sensors
Ultrasonic transducers (sonar) provide measurements in a cone of vision around 35° with a good accuracy. However, the update rate is lower than infrared sensors, the measurements are altered by gradient of temperature and they can be disturbed by each other.
For the project, a first MB1000 LV-MaxSonar-EZ0 sonar was purchased. This sonar does not support the I^2C communication used in the project, therefore five new sonars of the type SRF02 were acquired. They are placed at 45°, -45°, 90°, -90°, 180°(see Figure 3.2). The maximum distance range of the ultrasonic sensors specified is around $6\ m$ and the accuracy is below $10\ cm$. These sonars cannot handle high temperatures neither. The option considered is a sonar for outdoor environment from MaxBotic [15] that handle better though conditions.

Radar sensors
Radars work by using radio waves to determine distance, which allow to provide measures in all kind of environments, the accuracy of such sensors is around $20\ cm$ and the measure is carried out in an elliptical cone.
µSharp Patch Collision Avoidance Radar was preferred within the radars available in the market. It provides high resolution in the range of 0.5 to $120\ m$ and its horizontal and vertical fields of view are 50° and 30° respectively. The operational temperature range varies between -55° and +85°. For this project, the radar is located at 0° facing the front. In order to use it in high temperature conditions, the radar must be protected with a case, dielectric foams provide good protection without having signal loss [16].

3.3.2 Sensors positioning

Figure 3.2 shows how the sensors are placed within the drone. The radar has the highest range of all the sensors, hence it has been positioned facing the front of the drone considering that it is moving forward. The aim of the IR sensors is to add robustness and redundancy, therefore they have been placed to support the most sensitive areas (i.e. three of the zones the obstacle avoidance has been implemented: front, right and left). Their high update rates give them an advantage over ultrasonic. The sonars endeavor to analyze the maximum non-covered yet domain without disturbing each other. The different numbers refer to the different areas the detection zones have been divided into. Each area has four possible states: green or ”far”, yellow or ”near”, red or ”danger”

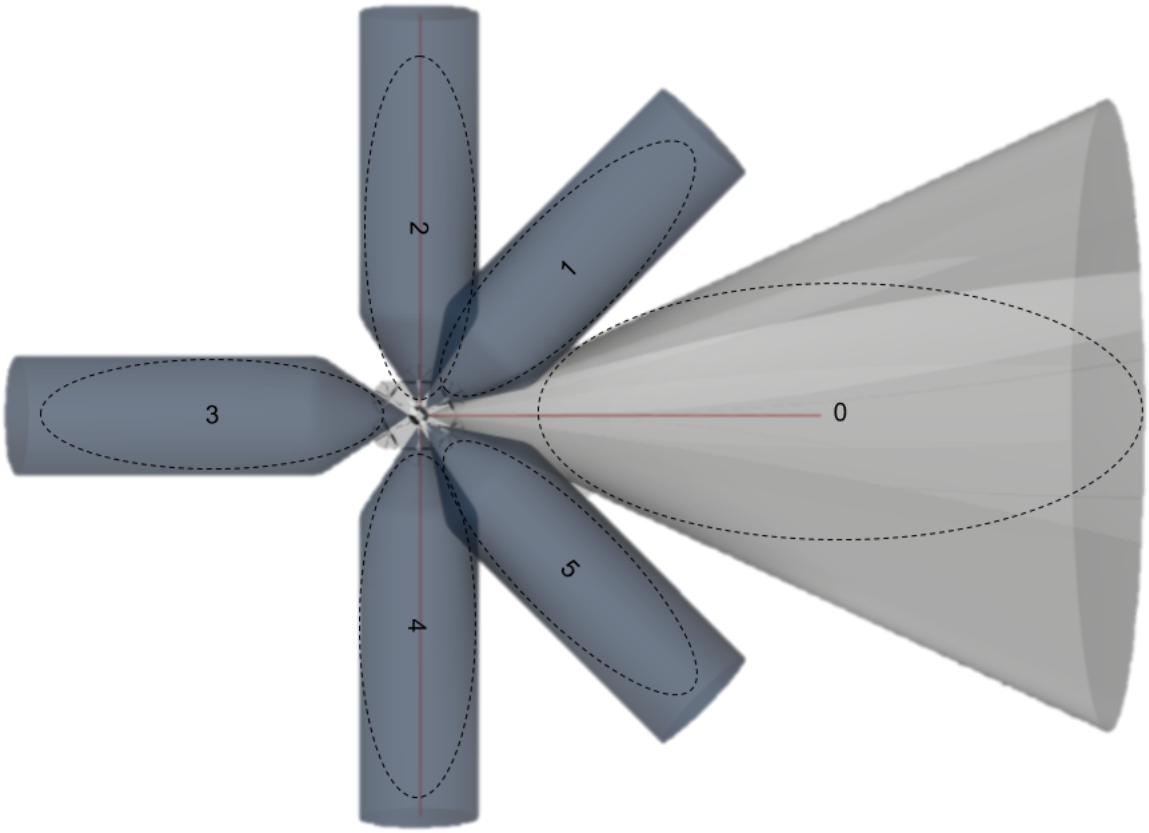


Figure 3.2: Drawing of the drone sensors' areas, 0° corresponds to the line in area 0. Radar cover zone is represented in light gray, the sonars in dark gray and the IRs in red.

and grey or "empty"; the control is based on these states to take the right decision.

3.3.3 Programming

The first step is to gather the data from the different sensors, however, they use different protocols. For instance, IR sensors are giving the distance with an analog output which is converted through a Analog to Digital Converter (ADC) to be sent by I^2C to the Raspberry Pi. The sonars are directly sending the measures through I^2C. Concerning the radar, the transmission is handle with UART serial communication. The use of a bus for communication allows a high flexibility, the sensors can be swapped, updated and replaced easily. In fact, every sensors is given a fix number which stays fix no matter where it is connected.

Radar
The radar provides directly the distance in meters as well as the signal noise ratio and the checksum. Therefore, it is not necessary to apply any correction to the measures. In order to obtain robust measurements, it is important to analyze if the distance returned by the sensors is whether noise or actual distance. An error handler is implemented in the code, the sensor is detected as faulty if 5 measures do not fulfill the checksum. If the measurement is below the minimum range ($40cm$), the value is not taken into account for the average. Finally, if the distance is higher than a threshold ($6m$), the zone is declared as empty. The update rate is 25Hz, but could be increased if needed.

Sonars
The sonar is directly sending the distance in meters or inches with two bytes accuracy. In order to increase the readable range (limited by the two bytes), it has been decided to obtain the distance in inches and then convert into meters. At boot up, the sensor operates an autotuning, which calibrates the sensor and give the minimal range. This value is used afterwards in the program to detect whether the sonar is faulty. As for the radar, a high range threshold is defined ($5m$). The update frequency of the sensor is 25Hz, however a faster rate could be achieved.

Infrared sensors
The analog values from the infrared sensors are converted to a digital representation with an 12Bit ADC converter 4 channels. The gain of the converter can be defined to select the range and has be set to $\pm 4.096\text{V}$. The conversion between digital to analog is then:

$$Voltage = \frac{4.096}{2047} digitalInput \tag{3.1}$$

The relation between the distance and the voltage for an infrared sensor is non-linear, as it can be seen on Figure 3.3. Actually, the distance is proportional to the inverse of the voltage $L = 1/V$. The exact function relating the voltage to the distance must be measured (see section 4). Another important property of the non-linearity is that the accuracy of the sensor is extremely good for close distances but decrease incrementally when reaching a further distance. Concerning the error handling, it is a bit different than other sensors as the values given are not the direct distances but the voltage. By consequence, the measure is declared faulty if the voltage is out of range (below $0.2V$ or higher than $3V$). If the measure is below 0.5V the zones is declared as empty. The IR sensors are the only ones to be analog, thus an analog filter could be applied to remove disturbances. However, after testing it has been defined that the impact was limited. The update rate of the sensor is 25Hz, it could be slightly increased up to 50Hz due to the time needed for the measurements.

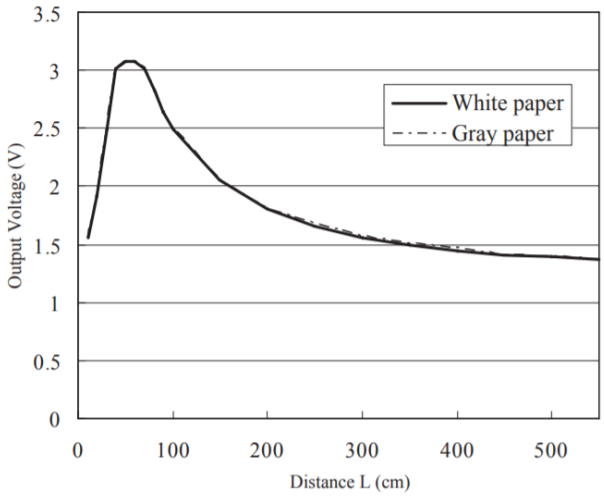


Figure 3.3: Relation between the distance and the voltage for an IR sensor (SHARP GP2Y0A710K0F) [5]

Filtering
Even with the error handler for each individual sensor, there are still some values that are incorrect/noisy. In order to smoothen the output signal, a median moving filter is applied. The moving filter works by using an array of defined size and replacing the older value by the latest measure at every sample time. It has the advantage to provide an output at every sampling, in opposite to a normal filter where it waits to gather x values before to give the outcome. It can also be discussed the interest of a median filter over an averaging. After analyses of the output from the sensors it seems that the noise often appears as spikes of short time length. By applying an average, the spike will have an important impact if for instance only 5 values are averaged and the final result is going to be biased. Whereas the median filter is taking the middle value, which has the advantage of disregarding extremum. However, the output of the filter has the drawback to be slightly less smooth.

3.3.4 Sensors fusion

The purpose of the sensors fusion node is to gather the measurements from all the **9** proximity sensors the drone is equipped with, to combine those pieces of information and to determine *in fine* if there is an obstacle nearby the drone and if it's the case where and how close it is.

The 3D space surrounding the drone was sliced in 6 different zones (see Figure 3.2), roughly corresponding to the cone of vision of each sensor to simplify the collision avoidance system but also because it's not possible to detect more accurately the position of an obstacle relative to the drone as each sensor is outputting just a distance information and not an angle. Based on the zone in which an obstacle is detected and how close it is, the corresponding overriding actions are taken, that is for instance the pitch angle (corresponding to how fast the drone is flying forward or backward) is limited to a certain upper boundary. The control algorithm is explained more thoroughly in the next section. To simplify this algorithm, we decided as well to bin the distance measurement monitored in each zone in **4** well defined, so-called, proximity states. So, for instance, if the distance measured in a zone is over a certain given reference, it's assumed that there is no object and a gray color property is attributed to that zone (numbered 0). If the distance is below the smallest reference, the object is considered to be very close (hence the situation is dangerous and immediate actions must be taken to avoid crash) and a red color property is set to the zone (equivalent to an integer numbered 3). If the distance is comprised between those two extremities references, the zone can either be assigned a green or yellow color (integer value of 1 or 2), depending on other references threshold values. The reasons we opted for splitting the proximity measurements in 4 different, well defined states was, as explained before, to simplify the algorithm on one hand but also to reduce the computational resources required on the other hand. Indeed, the Raspberry-Pi has limited power and advanced PID algorithms are quite demanding. In our scenario, identical control actions are taken whenever an object is bounded to the same zone and proximity state. Therefore, actions only need to be modified when a zone changes to another state. Evidently, this comes with the downside of a rough, inaccurate and quite brutal control at time. However, the advantages and ease of implementation of this mechanism easily overweight the downsides considering the available resources available.

The sensors fusion node is also in charge of monitoring the sensors' status and of warning the pilot of a faulty sensor by modifying the corresponding status icon on the heads-up display. Additionally, it performs the regular mode check-up and sends a sleep instruction to all the sensors nodes if the manual mode is selected by the user. To see how the sensor fusions node interacts with the other nodes, go to Appendix B.

The following will attempt to describe the working principle of this node. During boot-up, the node waits $10\ s$ until all the sensor's nodes have finished their initialization. Each time a sensor node initialization is finished, it calls a service in the sensor fusion node in charge of incrementing a counter. When the counter reaches 9 (the number of sensors node), the program reads all the messages sent by the sensors node to assess whether they're working on not. Once the boot-up is done, the program enters the main loop whose it never exits unless the user specifically asks for it (by typing CTRL+C in a terminal or by shutting down the hardware on which the program is running).

At each iteration of the main loop, the node retrieves the messages sent by all the sensors nodes and compares those values to the proximity boundaries that were defined and set as parameters in the parameters server during boot-up of the ROS core. Let's assume that the green and yellow thresholds are respectively 4 and $2\ meters$ for the example (note that the advantage of having them defined in the parameter server is that the algorithm can be tuned just by modifying those values in one place,that is in a text file). If the sensor monitoring the back zone (numbered 4) sends a distance of $4.5\ meters$ the first time, the zone is considered to be gray and a message containing the zone and its corresponding proximity status is sent to the decisions taker nodes in the form of an array of 2 unsigned chars : $[4, 0]$. The decisions taker node sends an acknowledgement message to confirm it has received the zone message. If the message wasn't received, the sensors fusion node will keep sending the message at each loop iteration until it's finally caught by the other node. If the message is received, the sensor fusion node will only send one message and won't send any others till the state of the zone is modified.

Therefore, if at the 10th iteration of the main loop, the distance measured in the rear zone switches from 4.1 to $3.9\ m$, the proximity status will transition from gray to green and a new message : $[4, 1]$ will be posted on the zones topic to be read by the decisions taker node. Requesting for a confirmation of reception is a neat way to limit unnecessary traffic on the topic.

Lastly, it's worth mentioning that the 3 most critical zones (the upfront ones) are monitored by two sensors to provide redundancy. Consequently, even if one of the two sensors stops working, the proximity information can still be retrieved. As with only two sensors measuring one zone, there is no mean to determine which one is faulty (if any), therefore the most critical distance is always considered. Hence, if one sensor detects an obstacle with a green proximity and the other one with a yellow proximity in a given zone, the zone is set with the later proximity.

3.4 Obstacle detection and avoidance

The live cameras display can be used by the pilot to guide the drone. However, it could be necessary to control the drone in a semi-autonomous mode considering that the pilot's vision is not going to be reliable enough at all times due to the harsh environment. The sensor fusion node gives information about the zones and the proximity of an object if detected within the pertinent zone, and therefore this feedback has been adopted to implement an autonomous obstacle avoidance algorithm with the aim of helping maneuvering the drone.

The algorithm developed to come up with this autonomous obstacle avoidance is based on [11]. The decision taker node receives the information of a zone and its color each time the proximity of a zone is updated, and the joystick control commands from the pilot which contains information about throttle, yaw, pitch and roll in addition to some other parameters. In contrast to [11], it has been decided to carry out the control regarding the Euler angles instead the angular velocities due to the lack of information about position and velocity as a GPS cannot be used inside buildings. In addition, the control has been performed only horizontally, but it would be simple to implement the vertical obstacle avoidance as well.

Once the previous mentioned data are processed, the algorithm overrides the values sent by the joystick regarding the color of the zone and if this corresponds to front, back, right or left. For the case of empty, green and yellow, the algorithm multiplies the roll or pitch command from the joystick by 1, 0.7 and 0.5 respectively. If the resulting value is higher than a specified maximum angle for each zone, the pitch or the roll is defined as that maximum angle. If the area is red, then the pitch or roll is set to $\pm 5\%$. The reason for this value is that even if the angle is set to zero, the drone can keep a constant velocity towards the direction of the obstacle and bump into it. Setting the roll of pith to a small value in the opposite direction ensures that the drone is not going to continue drifting in the direction of the obstacle.

Finally the overridden values are sent to the Pixhawk in the same format as the message received from the joystick before going through the decision taker algorithm. The values which are not modified (i.e. throttle, yaw, kill...) remain the same.

3.5 GUI Application

The Graphical User Interface (GUI) application is one of the mean for the drone to share its information with the end user (here visually), so (s)he can make the proper decisions, whether they are related purely to the flying experience or to the rescue situation more broadly.

The interface displays the following information:

- Visible spectrum video feed;
- Infrared video feed;
- Obstacles information (zones and proximity);
- Sensors' status;
- Flying mode (radio button pressed by the user);
- Connected, armed and guided status;
- CPU temperature;
- Data relative to the battery :
  - Voltage;
  - Current;
  - Percentage of charge.
- Compass Heading in degrees

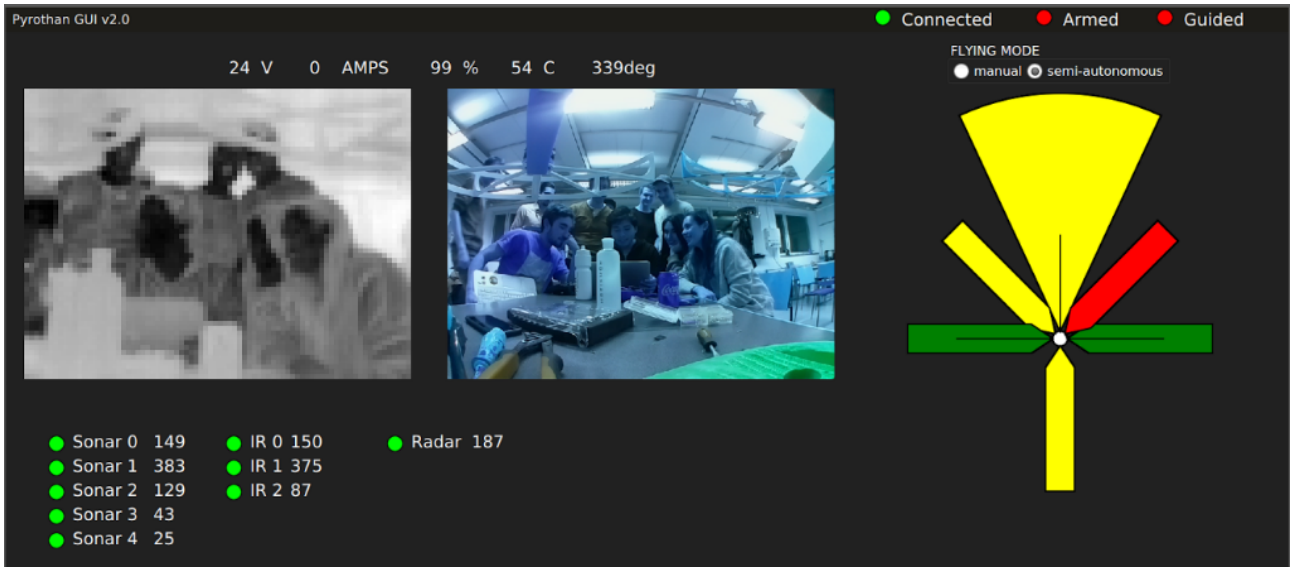


Figure 3.4: Heads-up display

The central window shows the video stream from the visible spectrum camera and is used primarily as a mean of navigation. That camera is basically the drone's (and therefore the user's) eye. The left box displays the feed from the infrared camera. It's used to determine the hot objects in the building under supervision and whether there are still any form of living beings battling against the flames and stuck inside that might need to be rescued. Additionally, it can be used as a supplementary mean of navigation when the drone is flying in smoky and fuzzy environments (when the regular camera barely sees anything else than a veil of fog).

Under those boxes, the sensors' status and distances are shown. A white circle means no information has yet been retrieved (likely because the communication between the ground station and the drone hasn't been established yet). A green icon identifies a functional sensor while a red one warns the user of the faultiness of the corresponding sensor. The "n/a" shown next to the sensor's name is the rough distance value measured by the sensor. It was used mainly as a mean of debugging but we left this information because the pilot might want to read accurate distance values.

On top of the windows, critical information relative to the battery status is shown, such as the battery voltage, its outputted current or its charge percentage. Indeed, it's vital that the pilot keeps an eye on those values to make sure the drone can be flown back safely to its ground station. The CPU temperature is also displayed because if it gets too high, the on-board computer in charge of the flight control could start malfunctioning, which could obviously lead to dramatic consequences (without considering the material loss of the drone itself).

On the top right, the user gets a visual feedback on whether the connection with the drone is established, whether the drone is armed (that is, ready to fly) and whether it's set up in guided flying mode. Just below, the user can choose with two radio buttons whether the drone is in manual or semi-autonomous mode. The former doesn't use the collision avoidance algorithm and one might want to use it if some sensors are malfunctioning or to get a finer control of the drone (at his own risks obviously).

Lastly, the proximity status of each zones are displayed on the figure. As stated above, a white color signifies that no information was retrieved because the connection is not established. Otherwise, each zone can take one out of four different colors (between gray, green, yellow or red as a reminder) depending on the proximity of the obstacle.

In terms of implementation, the Qt 4 and 5 libraries were extensively used to create the GUI. Specifically the Qt 5 library was selected because it quickly allows programmers to display information in a user friendly and easily configurable fashion. The senor data could have very well been displayed within the ROS framework using various nodes and libraries available by the ROS community. However, the team felt a single, professional, and well-designed GUI was not only an important aspect for the end-user, but an relevant technical skill to develop.

3.6 Flight Control

Multirotors are inherently unstable systems. This section presents and motivates the design choices made in terms of Hardware and Firmware to make the drone capable of stable flight.

3.6.1 Hardware

Here all of the flight critical hardware components will be presented and the reasoning behind the component selection will be explained.

Pixhawk
As mentioned earlier, the drone uses three different processing units. Two Raspberry Pi 3s and a Pixhawk 2.1. The reasoning behind this decision is because the Pixhawk runs a real time operating system called NuttX, which means that there are no background tasks that are running and slowing down the processor. The code that is loaded onto the Pixhawk is the only thing that is executing and therefore there is no delay when executing commands.

Multirotor aircrafts are very unstable and need a lot of control inputs to stabilize them. For this reason the Pixhawk is a suitable choice for the stabilizing control loop as the real time properties minimize latency. However, due to the limiting processing power of the Pixhawk, it is insufficient to handle more advanced control algorithms. For example, handling multiple sensor readings and obstacle avoidance requires a lot of processing power and therefore this is handled on one of the Raspberry Pis.

The Pixhawk 2.1 is an open source flight controller and is widely used in the RC-flight community. The large community behind the Pixhawk has led to crowd-sourcing software development of flight controllers that are designed to specifically run on the Pixhawk. The two mainstream flight controllers are ArduPilot and PX4. The difference between these will be discussed in 3.6.2. The Pixhawk has its own integrated triple redundancy IMU which makes it very robust and was one of the reasons to opt for using the Pixhawk. Another benefit that the Pixhawk brings, is its versatility. It has a lot of different i/o ports and it supports multiple communication protocols for example, CAN, Serial, I^2C etc. This greatly facilitates the development process and makes it easier to make design changes to the project at later stages.

Motors

When choosing appropriate motors an early weight estimation of $3000\ g$ was made. The frame that had been selected was $550\ mm$ in diameter which limited the propeller size to 9". With this knowledge, an estimation on how much lift could be generated with a certain motor was made with an online RC thrust calculator [17]. It was found that the DJI 3508 motor would generate approximately $5400\ g$ of lift and thus the decision to use that motor was made. See appendix C for full calculations. It is a brushless out-runner and it is equipped with multiple mounting solutions for propellers, including a self tightening hot-swap design. The motors were part of the DJI E600 Multirotor Propulsion System package that includes the motors, electronic speed controllers, ESCs, and propellers. One of the reasons for selecting this packet was that it made sure that the motors were compatible with the ESCs and also DJI have been in the drone businesses for a long time and therefore they have built a repetition for creating efficient and stable products [18].

Electronic Speed Controllers
The ESCs are of the model DJI 20A ESC Injection Molded. They can handle currents up to $20\ A$. They have advance control algorithms that generate very short response times which is crucial for stable flight. The ESCs are also one of the lightest ESCs that can handle 6S high voltage power system [18].

Propellers
Originally the choice of using 9" propellers was made in order to keep the diameter of the drone as small as possible. However, as the project elapsed, the realization that the weight of the drone would increase more than anticipated set in. To solve this problem the drone needed bigger propellers that could generate more lift. In an effort to save time and money the decision to use the propellers that came with the DJI E600 Multirotor Propulsion System packet was made. The propeller are 12" propellers with a 4.2" pitch. The mounting system fits the motors hot-swap design making them very fast to assemble and dissemble [18]. The bigger propellers did not fit on the prototype frame and this was one of the bigger factors that contributed to redesigning the whole frame.

Transmitter
The drone can be flown in two ways either via radio control or wi-fi. When flying with RC the pilot will not receive any assistance from the external sensors. This means that the pilot always have to see the drone when flying. The implementation of RC was done in order to quickly be able to flight tests and learn how the drone flight dynamics are in order to tune it. It is also a good way to learn how to fly it. The hand control chosen for RC flight is a Turnigy TGY-i6, which is simple RC-transmitter with two joysticks, two rotational encoders and four switches. It supports six 2.4GHz communication channels. This specific transmitter was chosen because it has all necessary features needed for basic flight control and at the same time being cost effective.

Joystick control
When the UAV is controlled via wi-fi a joystick, connected to the Ground Station, is used, see Figure 3.5. The joystick controller mimics the RC-transmitter in regards of throttle, yaw, pitch and roll. The arm, disarm and kill commands are implemented on the joystick as seen in the figure. An additional start button is implemented as a safety measure that has to be pressed after the kill button has been pressed.

3.6.2 Firmware

There are a few open-source flight controlling programs developed by different developers. The two most popular are ArduPilot and PX4. There is not a big difference between these two in terms of flight performance for regular quadcopters or hexacopters. Therefore, the choice of which one to use comes down to the surrounding features and licensing.

PX4
PX4 is a complete end-to-end flight UAV platform which includes the real-time operating system NuttX, middleware and flight control. It was first launched in 2012. The firmware supports control for several fixed-wing and multi-rotor frames, including hexacopters which is used in the project. The user guide[6] provides neces-

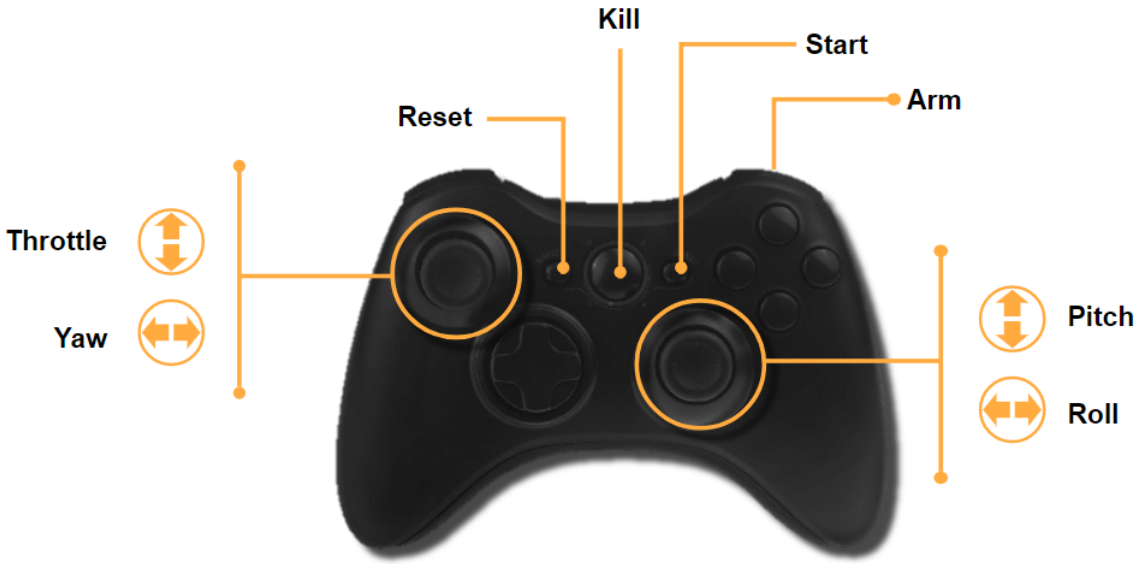


Figure 3.5: Joystick Controller.

sary information to setup and fly the drone and the developer guide[19] describes how to work inside the PX4 system architecture. The licensing for PX4 is BSD, which means it is free to use and modify without haven to redistribute the modification. This was the main reason for using PX4 in the beginning of the project.

Ardupilot
Much like PX4, ArduPilot uses the real-time operating system NuttX. ArduPilot was first launched in 2009 and is being continuously developed and updated. ArduPilot uses the licensing GPLv3, which means that if it is used in a commercially sold product you have to provide the source code, including any modifications that have been made [20]. As mentioned earlier, PX4 was used in the beginning of the project for its more unrestricted licensing. However, as the project developed the realization that ArduPilots licensing contributed to more developers distributing their modifications and improving the functionality at a faster pace than PX4. For this reason, the decision to change to ArduPilot was made. The two biggest factors in this decision were that ArduPilot handled motor loss better and it was easier to use the communication protocol MAVLink when communicating between the Raspberry Pi and the Pixhawk.

QGroundControl
QGroundControl is a ground control station with a user friendly interface that provides full flight control and vehicle setup. It was created to be used with PX4, however it has been updated to also include support for ArduPilot [19]. QGroundControl lets the user make an fast initial setup to make the drone flight ready. This initial setup consists of selecting the intended flight frame to be used, calibrating all the internal sensors i.e. accelerometer, gyroscope, compass and barometer. Moreover, it also includes calibrating the radio controller and selecting different safety precautions. For example, choosing the flight behavior when losing RC connection or the battery is low.

QGroundControl also enables the user full control over the parameters configuration. By changing parameters, the user can make more advanced configuration to fit the needs of their vehicle. For example, tuning the the PID-controller to get a more desirable flight dynamics or manually calibrating the on-board sensors to ensure perfect leveling and minimize drift.

Every time the Pixhawk is armed, it records a flight log. The flight log contains all the information needed to analyze the flight performance with the most important data being actuators outputs, desired roll, pitch and yaw angles from the pilot and actual roll, pitch and yaw angles. This information is very useful when tuning the vehicle, since it is easy to see the response time, overshoot and steady-state error. QGroundControl allows the

user to download these logs and graph them for debugging and analytics. However, it only supports graphing the logs for PX4. To graph ArduPilot logs, a third party program needs to be used.

Mission Planner
Mission Planner is similar to QGroundControl as it's also a ground control station. The big difference is that Mission Planner has been designed only to support ArduPilot [21]. It offers almost the same functionality as QGroundControl. However, it is a little more user friendly to work with ArduPilot in Mission Planner compared to working with ArduPilot in QGroundControl. As mentioned earlier it is not possible to graph ArduPilot flight logs in QGroundControl, but in Mission Planner it is supported. Moreover, Mission Planner also have a more sophisticated sensor calibration software. It gives you visual feedback that makes it easier to know how the Pixhawk should be moved in order to fully calibrate the internal senors.

The choice between QGroundControl and Mission Planner in this project came down to what flight control software was being used and what features in the control station was desired. Since the project started out with using PX4, the initial decision was to use QGroundControl. When the switch to using ArduPilot was made, Mission Planner was used for the initial setup. However, since the group members working on the project already had gotten familiar with QGroundControl, it was continued to be used for manual parameter configuration. After the initial setup Mission Planner was only used to download and graph the flight logs.

Calibration in QGroundControl
In QGroundControl the initial setup for PX4 is fairly straightforward. It starts with selecting the desired airframe. There are lots of different airframes to choose from including how many propellers, in what pattern the propellers are fixed and even different versions of fixed wing aircrafts. In this project, a hexcopter frame is used. The next step is to calibrate the internal sensors. The compass is calibrated by turning the Pixhawk on all of its six sides and slowly rotating it. As seen in 3.6 the user gets feedback when each side is done and also which side to do next. After that, the accelerometer needs to be calibrated, this is done by flipping the Pixhawk on each side in the same way as for the compass calibration. The only difference is that instead of rotating the Pixhawk it should be kept as still as possible. The gyroscope does not need to be calibrated on its own, hence the next step in the setup process is to do a level horizon. It consists of placing the Pixhawk in its take off position and keeping it still. This is done so the Pixhawk has a reference to a level plane. When that is completed the sensor setup is finished.

There are a few things left to do before the Pixhawk is flight ready. The radio controller has to be calibrated. This is done by giving different joystick inputs on the radio transmitter to match the inputs given in the QGroundControl interface. 3.7 demonstrates how the interface looks like during the radio calibration.

Calibration in Mission Planner
When using the ArduPilot and Mission Planner the initial setup is almost identical in terms of how the calibration process is performed. The only significant difference is the user interface when calibrating the compass. Instead of doing each side at a time in a certain order as it is done in QGroundControl, the Mission Planner interface shows graphics on how the Pixhawk moves. It then gives points in the graphic that the Pixhawk should aim towards. This is more clearly shown in 3.8

Flight Modes and Saftey
The last thing that needs to be complete to make the Pixhawk flight ready is to select flight modes and select safety protocol. This is done in the same way for both QGroundControl and Mission Planner. Starting with flight modes. Flight modes dictate how the drone will behave and how the pilot interacts with drone. In this project, two different flight modes have been used : stabilized and altitude lock.

In stabilized mode, the pilot has full control over the throttle input and also control yaw, pitch and roll angles. However, the control algorithm helps to keep the drone stable. This means that if the pilot only gives throttle input the drone will level out regardless of previous angles. This mode is beneficial to make some pre-flight tests with, since the pilot has full control over the throttle it is easy to spin the motors slowly ensuring that they

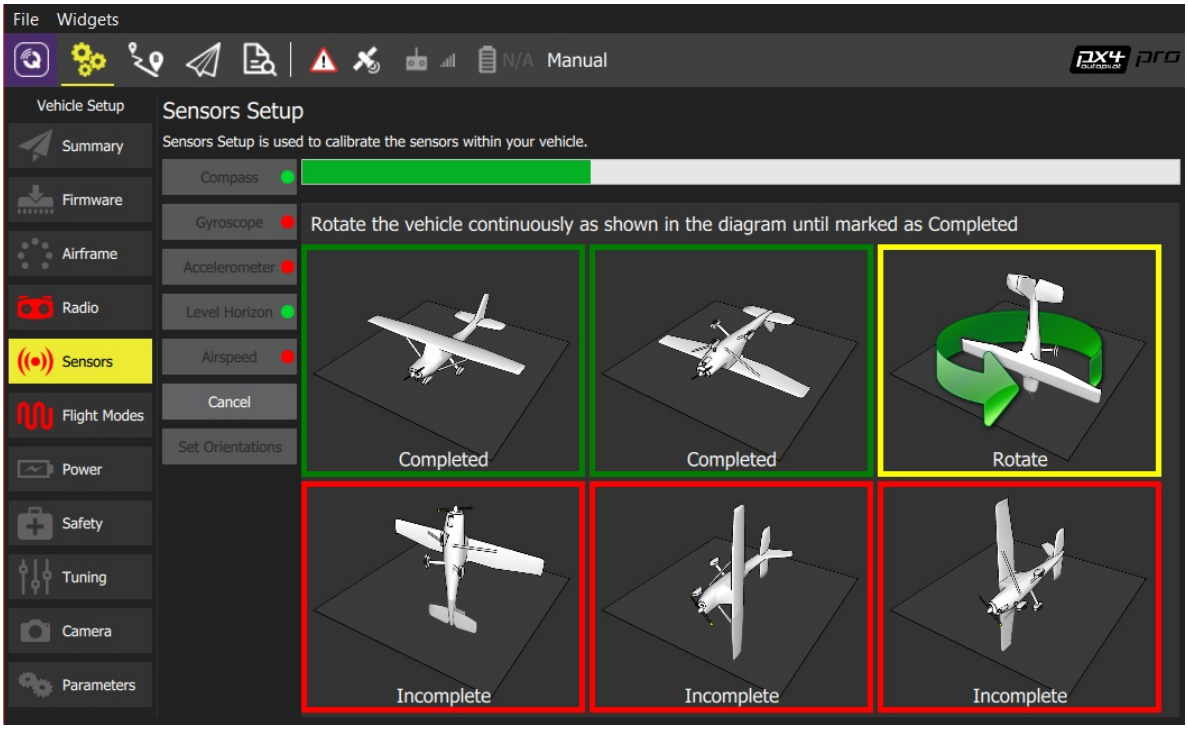


Figure 3.6: Compass calibration process in QGroundControl [6].

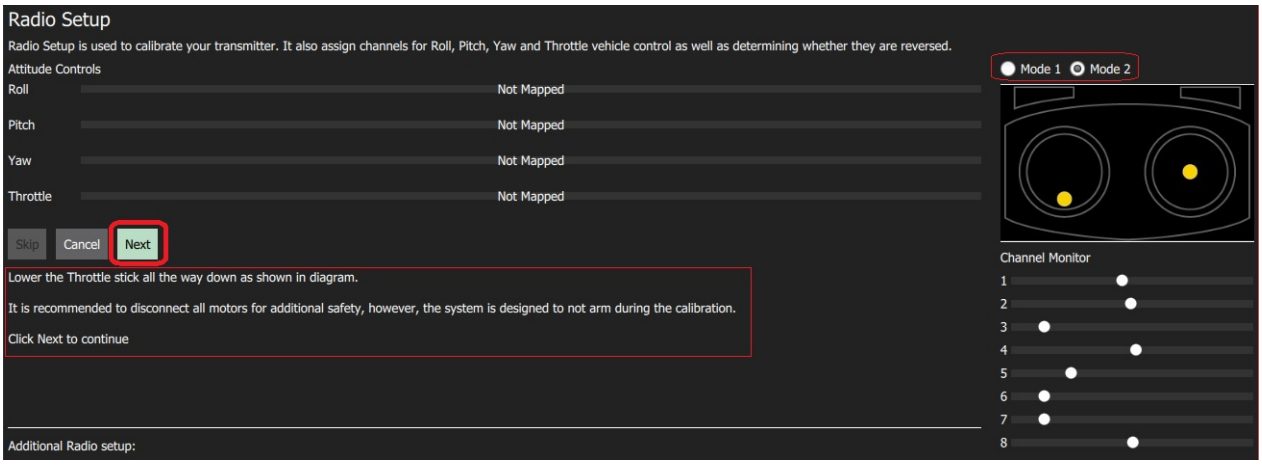


Figure 3.7: Radio controller calibration process in QGroundControl, the picture to the right is the joystick input to be matched [6].

all work as intended. The pilot could also give just enough throttle to barely lift off and see if the drone keeps its level or if it starts to lean a certain way. If the drone starts leaning it could be due to the ground not being leveled or the Pixhawk has not been calibrated correctly. The biggest issue with stabilized mode is that due to the pilot controlling the throttle it takes a lot of practice to be able to hold the drone at a steady height.

Altitude lock mode shares the same stabilization algorithm as stabilized mode. However, the pilot does not control the throttle directly. Instead, the pilot controls a set altitude. This means that on the hand controller the joystick corresponding to throttle has to be increased over 50 % for the drone to lift. The drone will increase its altitude with a certain velocity depending on how much the throttle stick is increased. If the throttle stick is under 50 % the drone will start to decrease its height and if the throttle is kept at 50 % the drone will hold its position. This flight mode makes it a lot easier for the pilot to fly the drone, since once the drone reach the desired altitude the pilot only needs to control yaw, pitch and roll. It also makes it possible to do very controlled

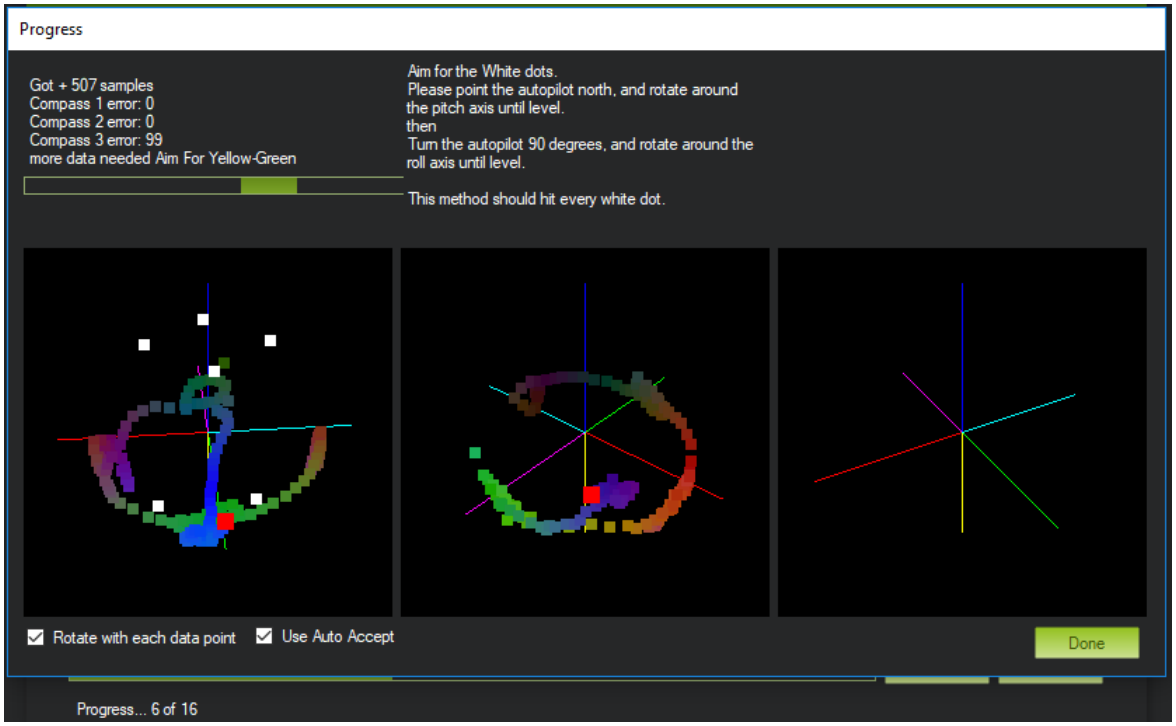


Figure 3.8: Compass calibration process in Mission Planner, the white dots is the points in which to aim the Pixhawk towards.

landings, since it is possible to set the height decrease speed very slow.

The last thing to do before the flying is to select the safety protocol. The safety protocol dictates how the drone should behave if errors occur during flight. These errors could be loss of communication, low battery or loss of GPS signal. Since there is no GPS on the drone the ”loss of GPS signal” safety setting is ignored. The fact that there is no GPS equipped also limits the choices for safety protocol for the two other errors. Therefore the safety protocol that is being used in this project is ”Land”. Which means that the drone will slowly lower its altitude until it lands if one of the errors ”loss of communication” or ”low battery” occurs. If a GPS was used, the safety protocol could be either ”hold position” or ”return to home” which would make it safer.

3.7 Pixhawk-ROS Communication

To communicate between the ROS nodes and the Pixhawk, the Mavlink communication protocol is used through the communication driver package mavros. All the communications between the Pixhawk, through the mavros node, are handled through a single pixhawk interface node. This node converts all flight information from the Pixhawk to a value that is displayed on the GUI interface. The node sets the Pixhawk mode and arm, disarm and kill commands. The node also handles the command output to the Pixhawk from the joystick by specifying the interval for throttle, roll, pitch and yaw which can be tuned to get a desired joystick sensitivity. The throttle was set to 0-100%, roll and pitch -45-45°and yaw -1-1rad/s.

3.8 Simulation

The unstable nature of UAV’s poses great risks for catastrophic failures during testing and operation. A simulation environment in the robot simulation tool Gazebo was therefore set up to heavily reduce the risks associated

with UAV development and to familiarize the pilot with the controller inputs. Gazebo can generate sensor data, including cameras and sonar, and supports dynamics simulations with custom robot models[22]. This enables the possibility to rapidly test algorithms on a modeled UAV in the simulation prior to hardware testing.

Both PX4 and Ardupilot come with respective Software In The Loop simulator[23],[24] which runs the flight stack on the computer with simulated flight-critical sensor data from flight dynamics models. Both the SITL and Gazebo support ROS which enables simulation of the flight stack in Gazebo and seamless interaction with the ROS architecture through MAVROS. Since both the simulation and the flight firmware on the pixhawk hardware communicate with the ROS architecture utilizing MAVROS, there is no hurdle between running the simulation and operating on the real hardware.

Due to time limitations, the full potential of Gazebo and the SITL was not utilized. Pre-defined multirotor models created for the PX4 and Ardupilot SITL were used which introduced a difference between the simulation and the real UAV when it comes to the dynamic behavior. The sensor's mounted(IR, sonar, radar and camera) on the real UAV were consequently not implemented in the simulation. However, this setup were sufficient for the development in this project as flight behavior can be simulated with any UAV model and real sensor data can interact with the simulation.

3.9 Design

The design requirements put on the drone required both a modular and hot-swappable configuration with an ability to withstand high temperatures for a limited amount of time. In this section the approach to meet these requirements is presented.

3.9.1 Modularity & Hot-Swap Design

One of the main features of the drone is its modularity. This ability to customize the drone allows the user to use it for different purposes. The design reflects this modular feature by dividing the drone into different units: a main unit, several propeller units, a battery unit and several sensor units that together make the sensor package unit. These units are illustrated in Figure 3.9, with the six propeller units and the main unit on top, followed by the battery unit and the sensor package unit underneath them. This way of breaking down the drone into subsystem has made it possible to develop the different units simultaneously and to a certain extent independently.

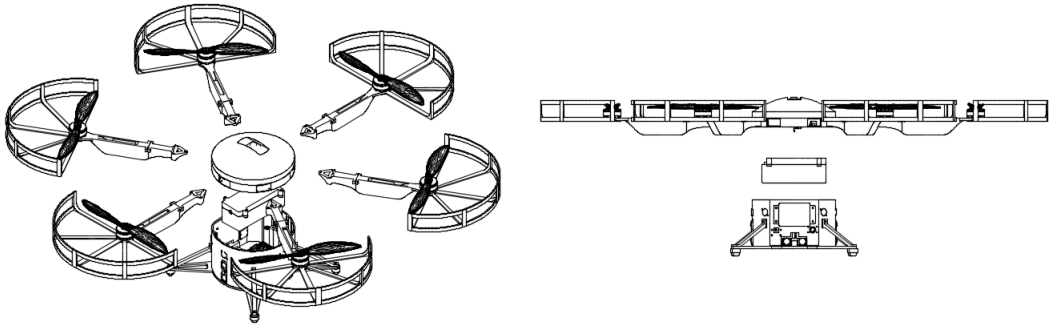


Figure 3.9: Left: Perspective view of a partly exploded view of the drone showing the six propeller units, the main unit, the battery unit and the sensor package unit. Right: Front view of a partly exploded view of the drone.

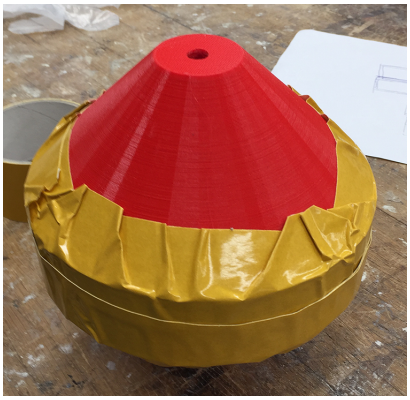


Figure 3.11: 3D printed mold.

The main unit consists of the main components needed to fly: Raspberry Pis, Pixhawk and power boards, with the addition of science boards that are used to communicate with the cameras and sensors. The battery unit is attached to the main unit but is surrounded by the sensor case, as can be seen in Figure 3.9. The propeller units are also attached to the main unit and together with the former mentioned units make up the flight crucial parts. The drone can fly and function properly without the addition of the sensor package unit, which will be described below.

The sensor package case is the holder of all sensors and cameras. These units basically make the drone intelligent in the sense that it allows for a semi-autonomous mode and allows the pilot to fly the drone beyond their visual line-of-sight. The positioning of the individual sensors has been presented in Subsection 3.3.2 and the cameras are positioned on the front of the drone. The idea is that these units could be swapped out depending on the need of the pilot and additional sensors or features could be added.

A hot-swap design was desired to fully take advantage of the drone's modular capabilities. To make this possible, an easy to remove "lid" was designed, which can be seen in Figure 3.10. The lid is secured to the drone by twisting it in place.

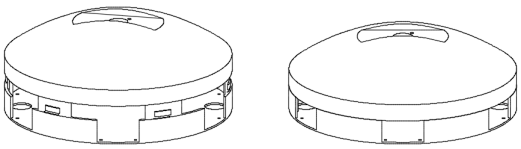


Figure 3.10: Images of the top part of the drone - the main unit. Left: The lid before enclosing. The top is secured to the drone by pushing it down and twisting it. Right: The lid when secured to the drone.

3.9.2 Heat Resistance (Robustness)

Several different materials and constructions were considered such as the aramid fibers used in the Faros drone [1] and fire resistant spray foam with a layer of aluminum coating to be able to handle the heat. The initial choice, the spray foam solution, was tested by creating a 3D printed mold, see Figure 3.11, which the foam was then sprayed into. This unfortunately resulted in a non-hardened shape with a lot of cavities and therefore was not used for the final prototype due to our inability to properly shape and test it.

The final construction consists of a sandwich construction which is chosen for the prototype and the concept can be seen in Figure 3.12. The reflective material is on outside of the isolating material to provide both fire resistance and reflect radiation from a fire. As a reflective material aluminum is chosen because of the low density and easy construction when using 80 µm aluminum tape [25]. The isolation isolates the inside of the

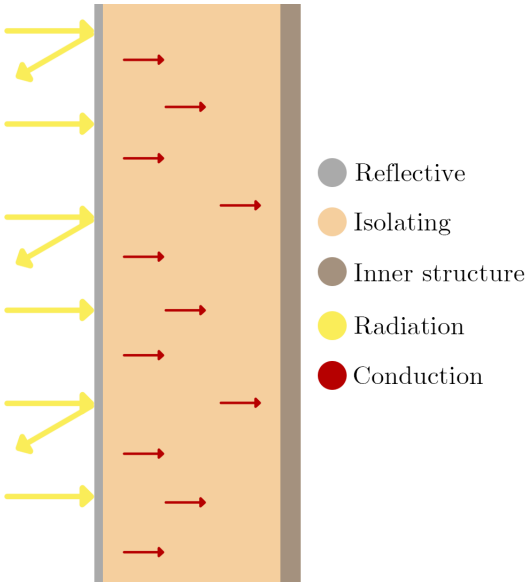


Figure 3.12: Isolation for the drone.

drone from convection and conduction and as an insulating material a foam material [26] was chosen. The foam has low density and low thermal conductivity, making it suitable as an isolating material. These materials are easy to shape as well which wouldn't cause any restrictions on the main construction and can be added after the major design decisions have been made.

The sandwich construction was seen as superior to other constructions since this approach doesn't require any active cooling and can be built with simple materials.

When using a closed system the system at hand is isolated from the outside but that also means the other way around, the inside is isolated against the outside. This means that the heat generated by the components in the enclosure will heat up the inside air since there is no air cycling through the enclosure. This has to be considered when using such a system.

3.10 Electronic Architecture

In order to comply with the modularity requirements that was given by Cybercom, it was decided to make it able for the user to remove and exchange the sensors attached to the frame of the UAV. Hence, if the user wishes, (s)he can easily exchange a malfunctioning sensor for reparation of the UAV. Furthermore, the user would be able to exchange the sensor to another type in order to give new functionality to the drone. To achieve modularity as well as reducing cabling inside the UAV, decision of designing printed circuit boards (PCB) was made. The PCBs are designed so that connectors could be soldered on to them. The connectors was then used to connect all cameras and sensors as well as connecting the PCBs together. Two different types of connectors (from two different types of manufacturers) were used:

- JST GH, right angle surface mounted connectors;
- Molex Mini-Fit Jr, 90° connectors.

The JST GH connectors has a pitch of 1.25mm and a current rating of 1A whereas the Molex Mini-Fit Jr connectors has a pitch of 4.2mm and a current rating of 9A. Since the Molex connectors can carry more current, they where used for the coupling between the PCBs. The sensors are plugged in using the JST GH connectors

and since different sensors use different communication interfaces, they demand a different amount of pins on the connectors.

Both the Pixhawk and the Radar communicate through universal asynchronous receiver-transmitter (UART) interface. Furthermore, the Raspberry Pi 3 model B has only one pair of transmission and receiving (Tx/Rx) pins which in turn means that it only allows for one device to be coupled and transfer/receive data at once. Consequently, it was decided to use two Raspberry Pi – one that receives and transmits data from the radar and one the handles the communication with the Pixhawk. Moreover, it was decided to have all the sensor used for obstacles avoidance on the Raspberry Pi allocated on the bottom part of the UAV and the raspberry pi placed on the top part of the UAV was used for camera computation.

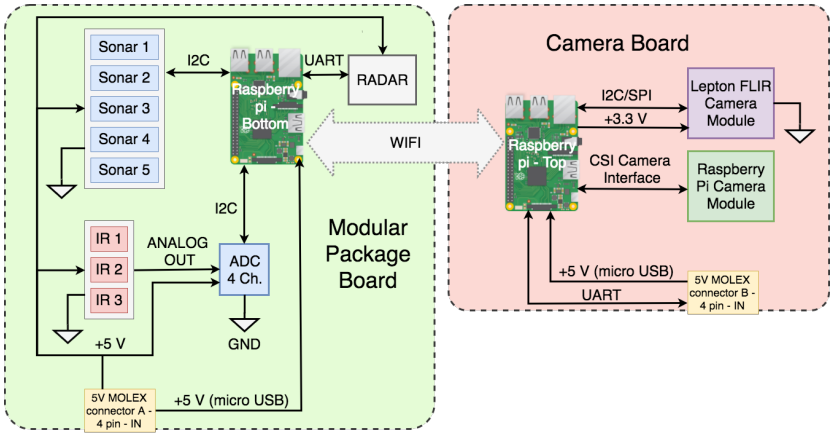


Figure 3.13: The Modular package board and the Camera board.

The electronics were divided into three PCBs – one for the obstacle avoidance sensors, one for the cameras, and one for the power distribution. Figure 3.13 displays the first PCB, used for the obstacle avoidance sensors. The PCB was designed to couple the sonars to the I^2C pins and the radar to the UART pins on the raspberry pi. The IR sensors was coupled to a 12-bit, 4 channel Analog to Digital Converter (ADC) that converts the analog signal from the IR sensors into digital signals that was sent out on the I^2C bus. The maximum amount of devices that can communicate on an I^2C bus is 127 for a standard 7-bit identifier, however, this was not an issue since the amount of coupled devices was not close to 127. The other board in Figure 3.13 displays the communication approach for the camera. The Raspberry pi camera module is *directly* connected to the Raspberry Pi via the Camera Serial Interface (CSI). The Lepton FLIR thermal camera module is connected to both the I^2C and Serial Peripheral Interface (SPI) on the Raspberry Pi. The reason for that is because the FLIR camera is calibrated using the I^2C protocol, whereas the actual footage is communicated through the SPI protocol.

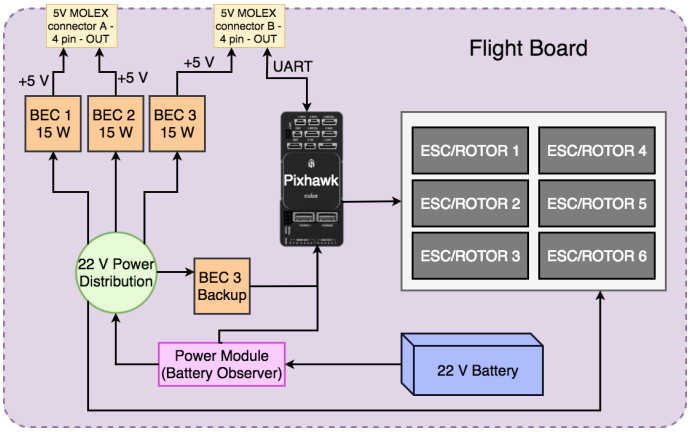


Figure 3.14: The flight board.

The flight board in Figure 3.14 is mainly used for power distribution for the different circuits as well as for the ESCs that controls the rotors. The flight board is also used to couple the Raspberry Pi on the camera board with the Pixhawk using the UART interface. The flight board contains Battery Eliminator Circuits (BEC) that drops the voltage level from the battery (22 V) to 5 V – which is then used to power the sensors and the Raspberry Pis.

3.11 The Prototype

This section is intended to show a few pictures of the prototype in the final state.

Figure 3.15: The first iteration of the prototype frame.

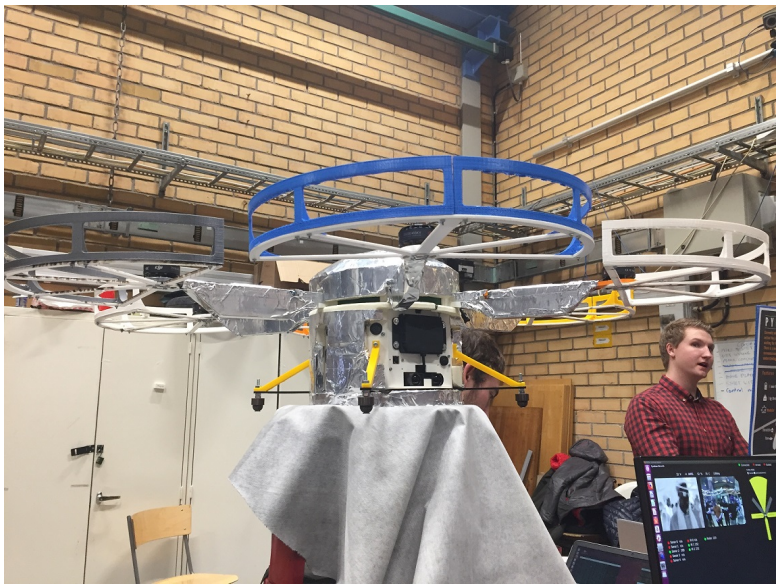


Figure 3.16: The final state of the prototype.

Chapter 4

Verification & Testing

This chapter shows how the requirements have been verified and tested. The first section corresponds to the flight performance and it explains the methods utilized to test the stability, the flight time and the loss of a motor. The following part covers the calibration and verification of the sensors, in particular the IRs. Next, the obstacle avoidance section explains how the decision taker algorithm has been tested and verified its behaviour. The fourth part expounds how the requirement of thermal resistance has been tested, and the final one deals with the modularity demands.

4.1 Flight Performance

This subsection covers the tests set up to verify the flight performance of the developed UAV in terms of stability, flight time and loss of motor.

4.1.1 Stability

In order to test other requirements, it was important to test the stability of the UAV during flight. The integrated system was first tested in the set up SITL simulation environment. The UAV was in manual mode without the obstacle avoidance activated. The test consisted of a take-off, flight for a few minutes and then landing, using the joystick. This test verified if and how the SITL reacted to the controller input from the joystick. The SITL test was performed several times.

The integrated system could then be tested on the real hardware in an outside environment with unknown factors such as wind speed and temperature. The UAV was configured as previously and the test was performed by taking of, flying for a several seconds and then landing. This test verified how the real UAV reacted to the controller input from the joystick. The hardware test was performed once due to time constraint.

4.1.2 Flight time

The requirement to fly for 15 minutes was tested on a prototype frame, weighing 2.6 kg. The test was done without any raspberry pi connected and with the RC controller. The test was performed outside by taking off with 100% battery and then flying for 15 minutes. After landing, the battery status was checked to verify the remaining percentage.

4.1.3 Loss of motor

In order to test the requirement for loss of one motor, a test case was set up in an outside environment. The test was performed on a prototype frame, weighing $2.6\ kg$, and with the RC controller. The test consisted of a take-off, short flight and then landing.

4.2 Sensors

The radar and the sonars do not need any specific calibration, so only a simple moving average filter has been implemented to output a median value. However, the IRs need to be calibrated for the reason that the distance is not linear to voltage.

For verifying and calibrating the IRs, the voltage output of each sensor and its corresponding distance according to the datasheet has been compared to the real distance and written down in an Excel sheet. The voltage measurements have been taken increasing the distance of an obstacle by 10 cm each time. Next, the inverse of the voltage and the distance values were displayed in a graph in Excel. A regression has been carried out to come up with a tendency line whose provides the function of the calibrated distance given the voltage.

The short IR distance can be linearized to $1/V$, hence the final distance measurement corresponds to (4.1).

$$L = \left(\frac{1}{V} + 0.0073\right) \cdot \frac{1}{0.017} \tag{4.1}$$

The long IR distance measurement is not linearly proportional to $1/V$, it is actually exponential. The equation to obtain the distance as a function of voltage is shown in (4.2)

$$L = \exp\left(\left(\frac{1}{V} + 0.4776\right) \cdot \frac{1}{0.193}\right) \tag{4.2}$$

A simple moving median filter has been implemented as well for the IR sensors. The calibrated and real distance curves can be observed in Figure 4.1. It can be observed that the calibrated lines are comparable to the real ones.

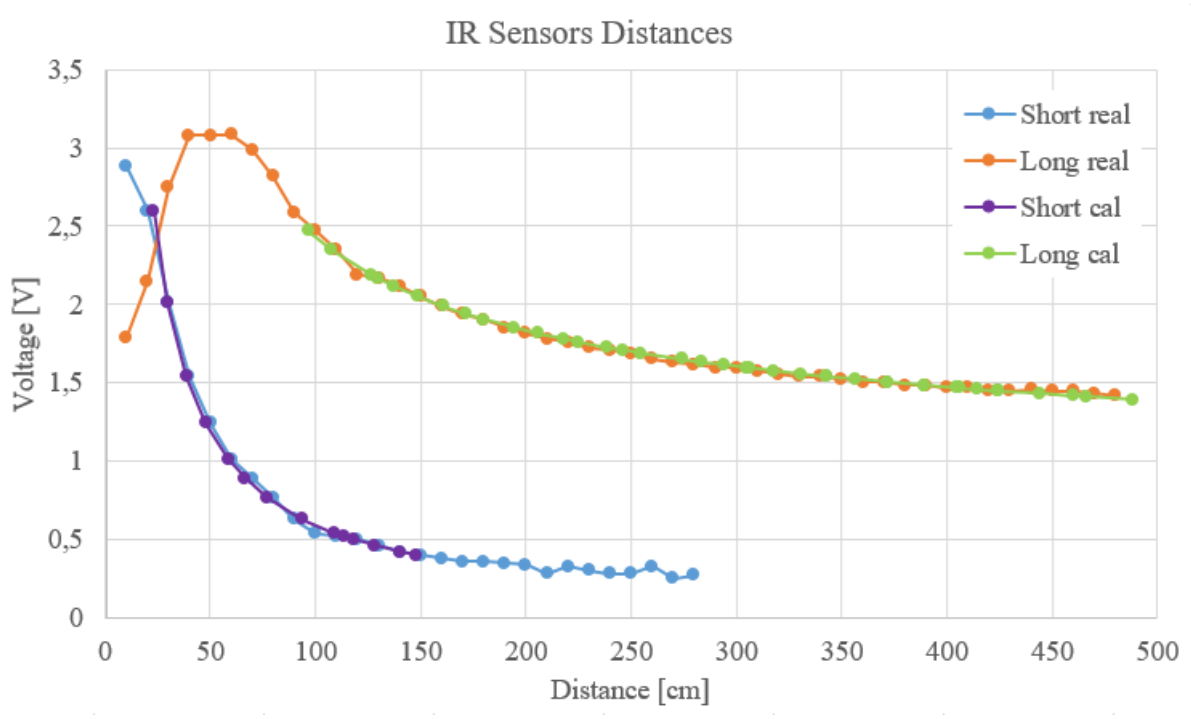


Figure 4.1: Calibrated and real distance curves as function of voltage

4.3 Obstacle Avoidance

All the flights performed until now were made in manual mode. To test the obstacle avoidance the manual mode was switched to autonomous mode when flying and only the radar was used in order to elude further problems. A person moved slowly closer to the drone holding an object it could detect. When the obstacle was close enough (around one meter) the drone started drifting backwards. This is the behavior which corresponds to the red area, therefore it seemed to work. However, when the person holding the object moved away, the color of the area changed and the decision taker algorithm overrode again the command of the pilot but this time not setting the angle to $\pm 5\%$. The sudden angle's variation made the pilot lose control of the drone and it finally crashed.

More tests should have been performed and the obstacle avoidance algorithm needs much more improvements but the lack of time made it impossible.

4.4 Thermal Resistance

For the thermal resistance, we measured it by using a testing equipment instead of using the real prototype due to the total loss of the prototype that could occur in a real test that wouldn't go well. We created a heat and flame resistant box as can be seen in figure 4.2 based on the sandwich construction mentioned in 3.9.2. The average thickness of the insulation was chosen as 15 mm. A test in a household oven was then performed with a temperature probe in the center of the enclosed jig and was tested with two different environmental temperature profiles.

Figure 4.2: Box made of the mentioned sandwich construction to test the heat resistance

The temperature rise of the air caused by the components can be calculated by the equation 4.3.

$$\Delta T = V \cdot \rho \cdot E \cdot c_p \tag{4.3}$$

Where ΔT is the change of temperature, V is the volume of air in the enclosure, ρ the density of air, E the energy added to the system and c_p the specific heat capacity. This equations hold if the following is assumed.

- All power drawn from the components becomes heat.
- The air in the enclosure assumes have no temperature differences.
- The total power drawn is calculated based on the average power drawn of the components.
- The enclosure is air sealed.
- The specific heat capacity doesn't change with temperature.

4.5 Deployment Time

To test the requirement concerning the rapid deployment under 1 minute, a test case was set up. The deployment time was tested by measuring the time taken to start relevant programs on the raspberry pis and the ground station, followed by a take-off. Assumptions were made that the person executing is familiar with the systems and that the drone is fully assembled and powered.

4.6 Modularity

The Modularity requirements were tested by measuring the time taken to disassemble vital components. The tests were done by one person with all the required tools and replacement parts available. The following components were tested:

- Sensor Package Unit
- Pixhawk
- Battery
- Motor
- Propeller
- Guards

The test was only performed once due to time constraints.

Chapter 5

Results

The aim of this chapter is to present the results obtained when testing the requirements. In contrast to Chapter 4, only the factual findings are explained. The chapter is divided into four sections: flight performance, obstacle avoidance, thermal considerations and modularity, and each part explains which are the outcomes extracted from those tests.

5.1 Flight Performance

This section presents the results of the Flight Performance tests.

5.1.1 Stability

The SITL tests showed an overall expected result. The simulated UAV moved in the expected directions based on the controller inputs, albeit quite sensitive (big movements from controller input). However, the take-offs were quite aggressive and the UAV lifted to a few meters depending on the throttle input from the joystick. There were also a latency between the throttle input and the actual take-off. After take-off, the modeled UAV was quite stable. It held its z-position good but had a small drift in the xy-plane. The simulated UAV also showed that the roll and pitch angle control commands translates to an acceleration in the UAV's xy-plane. The UAV would therefore not immediately react to a command in the opposite moving direction.

The results from the hardware test was successful in that it performed the flight without crashing. However, the results that could be extracted from the actual test with regards to flight performance were quite limited. From the flight log saved on the Pixhawk, some graphs could be extracted.

The UAV's altitude in relation to the throttle controller input is shown in Figure 5.1. As observed in the simulation, the take-off is aggressive rising to 1.8m with an initial throttle input of 100%. There is also a clear latency between the throttle command and the actual change in altitude, not only in the take-off but during the entire flight. Furthermore the altitude shows oscillatory behavior with an amplitude of approximately 0.1-0.2m.

The UAV's roll and pitch is shown in Figure 5.2. The roll and pitch reacted according to expectations based on controller input. However it is depicted from the figure that the UAV's flight performance in terms of roll and pitch is quite oscillatory, with an amplitude of approximately 2 degrees.

5.1.2 Flight Time

The battery had 48% of full charge after 15 minutes of flight, thus meeting the requirement.

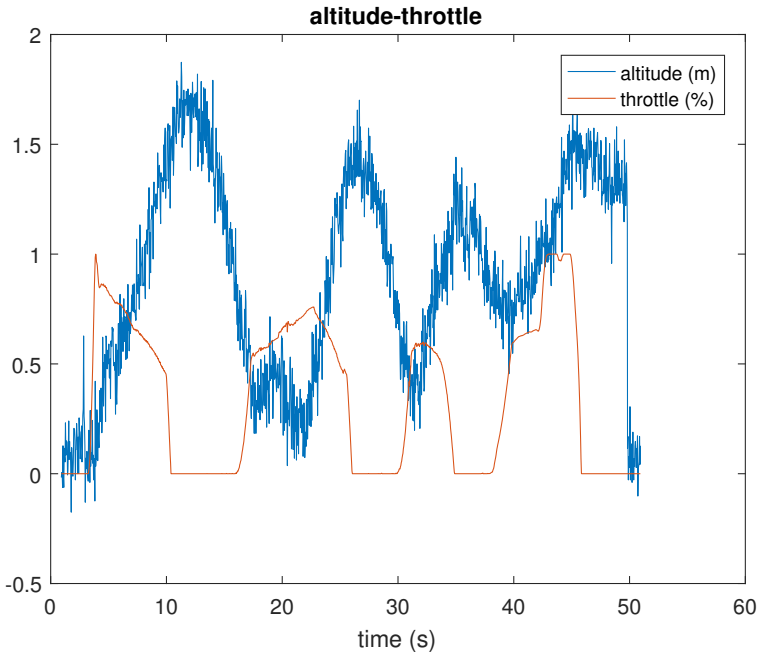


Figure 5.1: UAV's Altitude and controller input throttle from the flight test with the hardware.

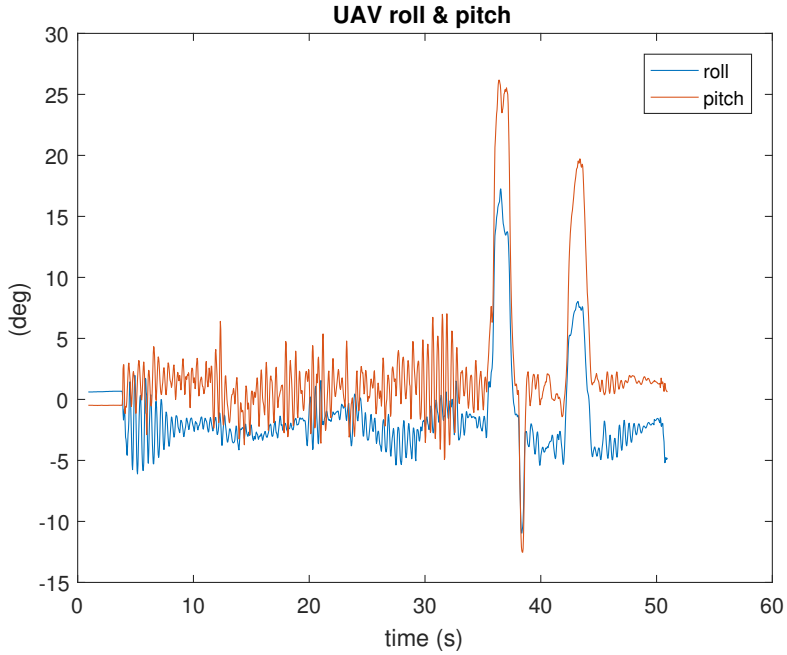


Figure 5.2: UAV's roll and pitch from the flight test with the hardware.

5.1.3 Loss of Motor

When performing the flight test with one motor disconnected the lift off was very unstable. The drone started tipping to the side where the motor was disconnected and it was at an approximated 30° angle before leveling out and taking off. Once it was in the air it could maintain its altitude, however it was very unstable and drifted heavily towards the disconnected motor side, forcing the pilot to do a lot of manual stabilization. Due to the instability, a soft landing was hard to achieve. The pilot had to make a fast decent in order to prevent the drone from landing while drifting sideways, which would cause it to flip over.

5.2 Obstacle avoidance

When looking at the results for the obstacle avoidance, it is difficult to conclude that it is an indisputable and full success. In fact, a plethora of problems occurred while developing this feature. First, the sensors ended up being less reliable than expected. Even by applying the different filters and error handling algorithm as discussed in section 3, the values are fluctuating a lot. For instance, without moving anything, the measurement of the radar would move from close proximity to really far within seconds. This occurs mainly in an environment cluttered with several objects. Thus, the reflection of the waves against those can certainly lead to this kind of inaccuracy. The problem is that it is difficult to assess which value is correct in that scenario. One of the observation we made throughout our testing schemes is that the sensors give more reliable values when measuring moving objects.
The sonars are the most rugged of all the sensors types, the output is stable and the distances between different sonars are comparable. A disadvantage may be the range that is limited to $5\ m$. A monumental integration problem appeared with the sonars, they have a cone of vision of approximately 60°, which is very useful to cover the entire drone. However, the design of new legs including a case for the ESC are coming to close to the sonars and interfere with them. Therefore the sensors should be repositioned lower in the casing. Concerning the IRs, they provide accurate values in perfect condition but start to fluctuate if the environment becomes harsher. Also, the main disadvantage is the "dead zone" from 0 to $1\ m$ that is very difficult to detect as the voltage given is similar to further distance (see figure 3.1) and can be misinterpreted.

With all the tests that were performed to validate that the sensors fusion node was behaving correctly (remember, it's *the* critical component of the collision avoidance algorithm), we noticed that the proximity of the zones monitored by two sensors weren't reliable and the team recommends optimizing the algorithm . The other functionalities work exactly as expected and no bugs have been observed, that is not to say that there is none.

The entire system, where the algorithm takes control over the pilot, has been only tested once. The drone did back up when the obstacle in the front was detected in the red zone. However, an unexpected problem occurred when the zone went back to yellow. As the control are overridden during "red zone" mode, the pilot doesn't have any feedback of what (s)he is sending and once the drone go back to a normal zone, the pilot values are considered again, even if they appear to be extreme. Which can lead to an uncontrolled situation. A solution would be to include a ramp when exiting the "red zone".

5.3 Thermal Considerations

Results for the thermal resistance constructions are shown in figure 5.3. The inside of the box reached a total temperature of 38° C after 5 minutes in 100 ° C ambient temperature and 55° C after 5 minutes in a 200 ° C environment.

The total temperature rise generated by the components was calculated to 19.8 ° C using equation 4.3 with the values shown in table 5.1.

In summary, based on the requirement that the drone should handle 5 minutes in 100 °C ambient temperature, the team was satisfied with the prototype's performance in regard to components within the thermally insulated shell.

However, components such as the BECs, motors, and accompanying wires were not tested to withstand such a high temperature and cannot be verified. Additionally, although the melting temperature of the PLA plastic used to create the frame is around 210 °C [27] - the mechanical integrity in such an environment was also not tested.

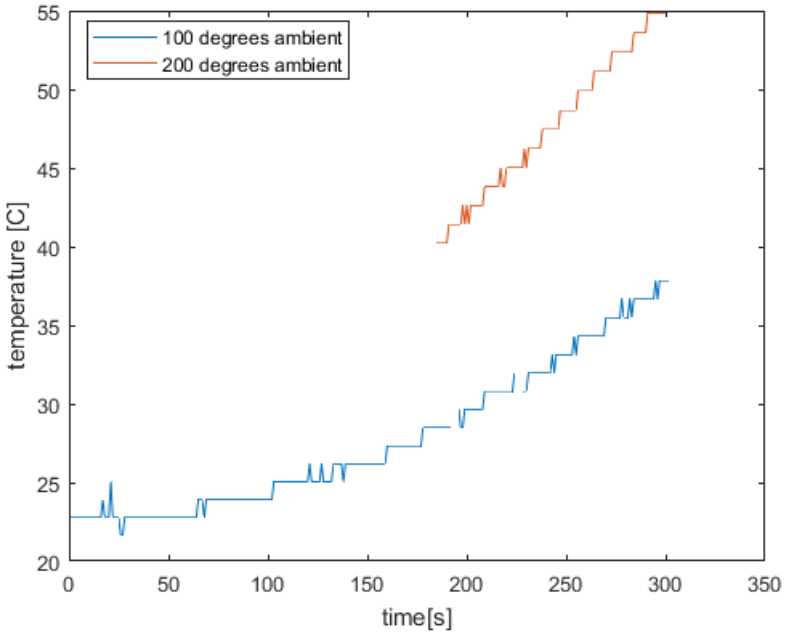


Figure 5.3: Temperature in the constructed test box in household oven test.

Table 5.1: Parameters for calculating the heat generated from components

Parameter	Value
V	$0.0040\ m^2$
ρ	$1.205\ \frac{kg}{m^3}$
E	$13.5\ W$
c_p	$1.005\ \frac{kJ}{kg \cdot K}$

5.4 Deployment Time

The time taken to deploy the UAV according to the test set up was 1 minutes 28 seconds, thus not meeting the requirement of 1 minute.

5.5 Modularity

Table 5.2 shows the time taken to exchange a component. The sensor package unit takes a significant time to disassemble since it also provides heat protection to internal components. Since the Pixhawk and Battery follows the disassembly of the sensor package unit, the time for those components are also long. The results for these components show that it doesn't meet the requirement of 5 minutes for vital equipment and 2 minutes for the battery. The propellers takes the least time followed by motor and guards since they are outside of the casing and does not have heat any external heat protection. These components meet the requirement of 5 minutes.

Table 5.2: The time taken to exchange components.

Component	Time	Requirement
Sensor package unit	11 min 25 sec	5 min
Pixhawk	13 min 7 sec	5 min
Battery	13 min 12 sec	2 min
Motor	1 min 11 sec	5 min
Propeller	11 sec	5 min
Guards	1 min 9 sec	5 min

Chapter 6

Discussion & Conclusion

In this chapter the conclusions from the outcomes of the project are highlighted and discussed. The quality of the results are analyzed and compared to the expected. This chapter is divided into four sections, flight performance, obstacle detection & avoidance, thermal resistance, and modularity.

6.1 Flight Performance

The flight performance of the integrated system is functional and satisfactory when the drone is in line of sight and with a trained pilot. It was also verified that the UAV handles a flight time of 15 minutes and that the Ardupilot flight controller supports flying with 5 motors. However, there are several points that can be improved upon.

The joystick, as described in the results, is sensitive to controller input which could lead to catastrophic failures if the pilot looses control over the UAV. The solution is not simply to reduce the max controller commands (throttle, roll, pitch and yaw) in the pixhawk interface node since this would reduce the agility of the UAV when changing direction of movement. With the joystick, there is an apparent trade-off between sensitivity and agility which would require more time to test and tune the controller parameters. Another problem with the control is the latency between command and actual movement which was also seen in the result. This was discovered late in the project and there was no time to investigate the cause of it.

The flight time was tested with a considerably lighter frame of $2.6\ kg$ compared to the full frame of $4.6\ kg$. It can be argued that since the full frame is only 77% heavier than the test frame, the test would also yield successful results with the full frame given that 48% battery remained with the lighter frame. However, the test was set up without the raspberry pis, sensors and joystick which could contribute to the power consumption. Another uncertainty is the power drain from the battery at different percentage. Therefore, more time was needed to fully verify the flight time of 15 minutes.

Since the test with 5 motors was carried out on the lighter frame of 2.6 kg, it could only verify that the Ardupilot firmware supports flying with 5 motors with a hexacopter configuration. However, the test was carried out with the RC controller and not the fully integrated system with the raspberry pis and the joystick. The test did not cover loss of motor in mid-flight as well.

Due to lack of time, the bump protection was not tested. Thus the requirement could not be verified.

6.2 Obstacle detection & avoidance

The obstacle detection has been one of the most challenging developments and the progress encountered several problems. As discussed in the results (section 5), the accuracy and precision of the sensors were not as good as expected and different solutions had to be found to counter their inaccuracies. However, even with all the different implementations, the detection of obstacles stays inaccurate. One of the root causes is the quality of the sensors, in fact, mostly all the sensors are relatively cheap ones, intended for hobbyist. Concerning the radar,

which is the most expensive sensor, should have performed well, however the imprecision certainly comes from the fact that it is made to be used outdoor with large environments and not indoor with many obstacles. In comparison to similar work, the team can say that the obstacle detection performed worse than expected. For instance, the article [9] uses similar sensors and achieve autonomous flight and collision avoidance. In order to improve the quality of the obstacle detection while keeping cheap sensors, it would be required more redundancy and the implementation of an advance weighted filter that fuses the different measures (IMU, IR, Sonar, ..) or a more computationally costly Kalman filter.

The basic decision taker algorithm implemented for this project works as expected, however a significant improvement can be done to reach a more advance obstacle avoidance. The performance is not comparable to [11] as they use the angular velocity, easier to control than the Euler angles. The problem with using angles is that even if they are set to zero, the drone can keep a constant velocity (due to the inertia) in the direction of the obstacle. The solution we came up with was to set the angle to a small value in the opposite direction, which is not efficient and reliable as it doesn't take into account the initial velocity but works to an extent. In addition, when we tested the obstacle avoidance, we realized than the change from the red proximity (close object) to another one is excessively violent, so a ramp to smoothly increment the angle should be applied.

The best solution for this feature of the project would be to create something similar to [10], where a PID controller is used. Nevertheless, the velocity or position data are necessary which is difficult to obtain for this project as the drone evolve indoor and in harsh conditions. The implementation would then require time, research and extended knowledge.

6.3 Thermal resistance

The thermal insulation in this product was created to stay within a budget which did not allow for more expensive materials, a short time frame and a strict weight budget. With these considerations there are several points which lack the thermal insulation which is needed to reach the requirement. In the final prototype no kind of insulation was used for the arms and ESCs and therefore would probably be the weakest link of the drone. But as a proof of concept the thermal insulating construction that was created met the requirements that was set.

If the solution created for the thermal resistance is compared with other work such as FAROS [1] it is hard to argue that our solution was better if we look at it's performance against heat. One must though once again consider the pricing and simplicity of the construction made. The aluminum tape in theory would provide a flame resistance surface, however this too was not fully tested to come to any confident conclusions.

If budget was larger, some other materials as carbon fiber or the dielectric case used in FAROS could have been afforded to build the entire drone. This would have definitively met the thermal requirements for the whole product and would have decreased the weight of the drone, achieving a more advanced product.

6.4 Deployment Time

The results showed that the deployment time requirement was not met. Due to time limitations the programs on the raspberry pis and the ground station had to be started separately which increased the time of deployment. It is possible to launch the programs on the raspberry pis from the ground station which would vastly decrease the deployment time.

6.5 Modularity

It is important to note that the tests regarding the modularity of the drone were only conducted in the disassembling phase due to lack of time. It is assumed that the whole process, including the assembly, would take about twice to triple as much time as given by Table 5.2. However, the results clearly show that some components were already exceeding the time limit without having to test the entire process, and the others would stay within the limit even if the elapsed time was to be tripled.

The thermal resistant feature of the drone puts some constraints on the modularity and hot-swap design as the insulation makes it more difficult to access the inside of the drone. This trade-off between modularity and thermal resistance highlights the main purpose of the drone, which is assisting firefighters in fires. Furthermore, the firefighters have a critical time window of 15 minutes. The critical time window limits the need for swapping out components. The only component that is crucial to change is the battery if its power level is low. However, it is assumed that the drone would arrive at the site fully charged and with functioning components. With a flight time of at least 15 minutes and a critical time window of 15 minutes, the issue is eliminated. It is also assumed that if the drone would crash inside, its parts would not be replaced since that would mean that the firefighters would have to go into the facility to replace them, which would beat the purpose of the drone.

Chapter 7

Future Work

For the future, first and foremost the usability aspect must be improved. With the current prototype it is arduous to control the drone, one must have some training before getting a hold of it. Some things to make it easier to control are to reduce the sensitivity and to implement an autonomous take off and landing. Taking advantage of the full capabilities of the simulation environment Gazebo by implementing a more accurate model and a realistic environment would not only ease development and testing, but also be a vital tool for flight training.

When looking at the semi-autonomous drive there are several points of interest to work further on and is possibly the largest area of potential growth. The first modification that should be implemented in a future version is the placement of the sensors compared to the arms. The cone of vision of every sensors should be more carefully studied before to define the position or designing the frame. The second point of improvement would be to purchase more accurate sensors. Also, after deep testing, it appears that the sensors fluctuate even after calibration. It would then be necessary to add full redundancy for every sensors and implement a fusion filter to compensate the fluctuation. One of the big problem encountered and discussed previously, is the impossibility to measure the velocity of the drone (due to the lack of GPS), and therefore, the control of the drone is more complicated. It is possible to implement specific sensors to measure the velocity, such as optical flow sensors or more complex systems using SLAM to estimate the position/velocity of the drone, and therefore implement a more advance obstacle avoidance algorithm based on a PID controller. In addition, for this project it was intended to develop a mapping algorithm, however due the lack of time and the impossibility to get the position of the drone, the group decided to skip this feature. Finally, it would be amazing if the automatic control is carried out in the three dimensions and a trajectory planner algorithm is implemented, leading to a fully autonomous drone.

The hot swap and modularity of the drone can also be drastically improved. In the current prototype there are many screws and nuts which some are really hard to reach and therefore it takes some time to disassemble. This could be changed to some kind of ”snap-on” mechanism which might not even require tools to take off and on and should be put in a easily attainable position.

If you take a look at a broader picture it might be interesting to have a fleet of several drones flying in synchrony and able to map the scenario in the smallest amount of time possible. This would reduce the mapping time and the firefighters could recognize quickly if someone is inside the building/warehouse/tunnel or if it is too dangerous to come into it.

Bibliography

[1] KAIST. A firefighter drone that flies and crawls up walls. Accessed: 2017-05-15. [Online]. Available: http://www.kaist.edu/_prog/_board/?mode=V&no=46765&code=ed_news&site_dvs_cd=en&menu_dvs_cd=0601&list_typ=B&skey=&sval=&smonth=&site_dvs=&GotoPage

[2] DJI. Phantom 3. Accessed: 2017-05-15. [Online]. Available: https://www.dji.com/phantom-3-pro

[3] New York Times. New york city's firefighting arsenal will soon include drones. Accessed: 2017-05-15. [Online]. Available: https://www.nytimes.com/2016/09/09/nyregion/new-york-city-fire-department-drones.html?_r=1

[4] Västervik Kommun. Första insatsperson i gamleby och drönare - nyheter hos räddningstjänsten. Accessed: 2017-05-07. [Online]. Available: https://www.vastervik.se/Omsorg-stod-och-hjalp/Trygg-och-saker/arkiv-nyheter-trygg-och-saker/Forsta-insatsperson-i-Gamleby-och-dronare---nyheter-hos-Raddningstjansten/

[5] *Distance Measuring Sensor Unit Measuring distance: 100 to 550 cm Analog output type*, SHARP, 12 2006.

[6] Px4 autopilot user guide. Accessed: 2017-10-30. [Online]. Available: https://docs.px4.io/en/

[7] MSB. Nytt reportage från 90 sekunder om drönare. Accessed: 2017-05-06. [Online]. Available: https://www.msb.se/sv/Om-MSB/Nyheter-och-press/Nyheter/Nyheter---RIB/90-sekunder1/

[8] Pitchup AB. Drönare för räddningstjänsten. Accessed: 2017-05-06. [Online]. Available: https://www.pitchup.se/dronare-for-raddningstjanst

[9] N. Gageik et al, "Obstacle Detection and Collision Avoidance for a UAV," in *IEEE Access (Volume:3)*, 2015, pp. 599–609.

[10] N. Gageik et al, "Obstacle detection and collision avoidance using ultrasonic distance sensors for an autonomous quadrocopter," in *UAVveek*, 2012.

[11] Agathangelos Plastropoulos, "Next generation radiation mapping using uas assisted dynamics monitoring networks," 2015.

[12] Altera. Enhanced temperature device support. Accessed: 2017-12-20. [Online]. Available: https://www.altera.com/products/common/temperature/ind-temp.html

[13] DroneZon. Dji mavic pro highlights. Accessed: 2017-04-08. [Online]. Available: https://www.dronezon.com/drone-reviews/dji-mavic-pro-highlights-review-frequently-asked-questions/

[14] FLIR. Flir-mlr 100. Accessed: 2017-12-13. [Online]. Available: http://www.flir.com/uploadedFiles/OEM/Products/Laser-Systems/FLIR-MLR100-Datasheet.pdf

[15] MaxBotic. Mb7040 i2cxl-maxsonar-wr. Accessed: 2017-12-13. [Online]. Available: http://www.flir.com/uploadedFiles/OEM/Products/Laser-Systems/FLIR-MLR100-Datasheet.pdf

[16] Azom. Dielectric foam – last-a-foam. Accessed: 2017-12-13. [Online]. Available: https://www.azom.com/equipment-details.aspx?EquipID=5238

[17] eCalc. ecalc. Accessed: 2017-12-22. [Online]. Available: https://www.ecalc.ch/

[18] DJI. E600 tuned propulsion system for multirotor. Accessed: 2017-12-22. [Online]. Available: https://www.dji.com/e600

[19] Px4 development guide. Accessed: 2017-10-30. [Online]. Available: https://dev.px4.io/en/

[20] A. development crew. Ardupilot user manual introduction. Accessed: 2017-12-22. [Online]. Available: http://ardupilot.org/plane/docs/introduction.html

[21] ——. Mission planner home. Accessed: 2017-12-22. [Online]. Available: http://ardupilot.org/planner/

[22] Gazebo. Accessed: 2017-12-18. [Online]. Available: http://gazebosim.org/

[23] Simulation. Accessed: 2017-12-18. [Online]. Available: https://dev.px4.io/en/simulation/

[24] Sitl simulator (software in the loop). Accessed: 2017-12-18. [Online]. Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[25] ELFA. Tesa aluminum tape. Accessed: 2017-12-22. [Online]. Available: https://www.elfa.se/en/aluminium-tape-aluminium-50-mmx50-tesa-50575-50mm-50-transpar/p/11052072?q=110-52-072&page=1&origPos=1&origPageSize=25&simi=99.65&no-cache=true

[26] ABIC. Renshape board materials. Accessed: 2017-12-22. [Online]. Available: http://www.abic.se/pdf/RenShapeBM_TABELL.pdf

[27] C. Prototypes. Pla prototypes. Accessed: 2017-12-22. [Online]. Available: https://www.creativemechanisms.com/blog/learn-about-polylactic-acid-pla-prototypes

[28] T. Noergaard, "The embedded systems model," in *Embedded Systems Architecture*, 2005, p. 447.

[29] Dronecode software platform. Accessed: 2017-05-13. [Online]. Available: https://www.dronecode.org/dronecode-software-platform

[30] Ros-introduction. Accessed: 2017-10-19. [Online]. Available: http://wiki.ros.org/ROS/Introduction

[31] Qgroundcontrol user guide. Accessed: 2017-10-30. [Online]. Available: https://docs.qgroundcontrol.com/en/

[32] Px4 basic configuration. Accessed: 2017-10-30. [Online]. Available: https://docs.px4.io/en/config/

[33] Px4 advanced flight controller orientation tuning. Accessed: 2017-10-30. [Online]. Available: https://docs.px4.io/en/advanced_config/advanced_flight_controller_orientation_leveling.html

[34] Px4 flight modes. Accessed: 2017-10-30. [Online]. Available: https://docs.px4.io/en/flight_modes/

Appendix A

Initial Product Pitch by Cybercom

Firefighting Assistant System

Custom Design Drone

Infra-Red Camera

Sensor Fusion

Internet-of-Things

Cybercom Group AB
P.O. Box 7574
SE-103 93 Stockholm, Sweden

Corporate ID 556544-6522
Phone +46 8 578 646 00
Fax +46 8 578 646 10

www.cybercom.com
info@cybercom.com

Problem statement

Currently when firefighters go into action they often do not know what is waiting for them inside the building. There is a need to make a quick reconnaissance to give them a better view of the situation. We want to improve their vision while they are in the field by creating a Firefighting Assistant System.

Especially during fires in buildings, it is required to know where the fire has spread or if there is a danger of the construction collapsing. A robust UAV equipped with cameras and sensors is a perfect solution to check it.

Project description

The first part of the project is to design and build a drone. It should be a highly customizable machine, that will be a good base for future improvements, while still being able to handle the harsh environment surrounding a fire. The UAV should be able to fly over the area on fire to get the best view of the situation while handling the turbulence and heat. The image from a normal and an infrared camera should be streamed to the operator's goggles/screens giving an up-to-date view of the situation. The data from any equipped sensors should be transferred the same way.

Since the drone will be operating in harsh environment and might hit obstacles or get hit by something, it is important that the drone can withstand collisions to some extent. Regaining full control over the drone after a hit and continue flying would be a must, as well as handle turbulence. One of the design goals is that it is able to handle losing one of the motors without crashing. Because the drone might encounter people inside a building, it should also have safety mechanisms. Possibility of getting hurt by the propeller blades must be reduced to a minimum.

The modularity of the system should enable it to later be equipped with additional subsystems such as:

- sensors, batteries antennas etc. depending on the situation
- making the drone controllable from outside if it flies inside buildings
- a framework for a concept of a swarm of smaller UAVs

Requirements

The drone shall:

- Cope with wind speeds of up to 10 m/s.
- Have a turnaround time of maximum 2 minutes with replacement of batteries.
- Have a turnaround time of maximum 5 minutes with replacement of vital equipment.
- Withstand bumping into objects
- Withstand one motor malfunction in flight mode
- Produce a steady video stream from both cameras
- Air time at least 10-15 minutes

The system shall:

- Provide an interface with up-to-date information to the firefighter

Disclaimer: The pictures above does not belong to Cybercom and will not be used in any commercial project. The pictures are added to give inspiration for students.

About Cybercom Group

Cybercom is an IT consulting company that was founded in 1995.

Cybercom is an innovative IT consulting company with 20 years of experience in IT and communications technology. Our consultants enable businesses and organizations to benefit from the opportunities of the connected world, to enhance their competitiveness or to achieve efficiency gains. Cybercom's expertise spans the entire ecosystem of this communication – Connectivity – and our delivery is both local and global.

Cybercom's principal market is the Nordic region, with established operations in Sweden, Finland and Denmark. Poland, India, Dubai and Singapore are international centers of expertise that partly support the Nordic business and partly represent their own specialized business. The group has approximately 1,300 employees in seven countries.

Kista office

The Kista Office is primarily focused on Embedded Systems and have several clients such as Ericsson, Scania and SAAB.

The workforce consists of approximately 100 employees. During the first half of 2017 Cybercom Kista will host 10 Thesis workers of which five are KTH Mechatronics Master students.

Cybercom will also introduce an Innovation Zone at the Kista office during the spring of 2017 to extend our focused work with Innovation and Sustainability.

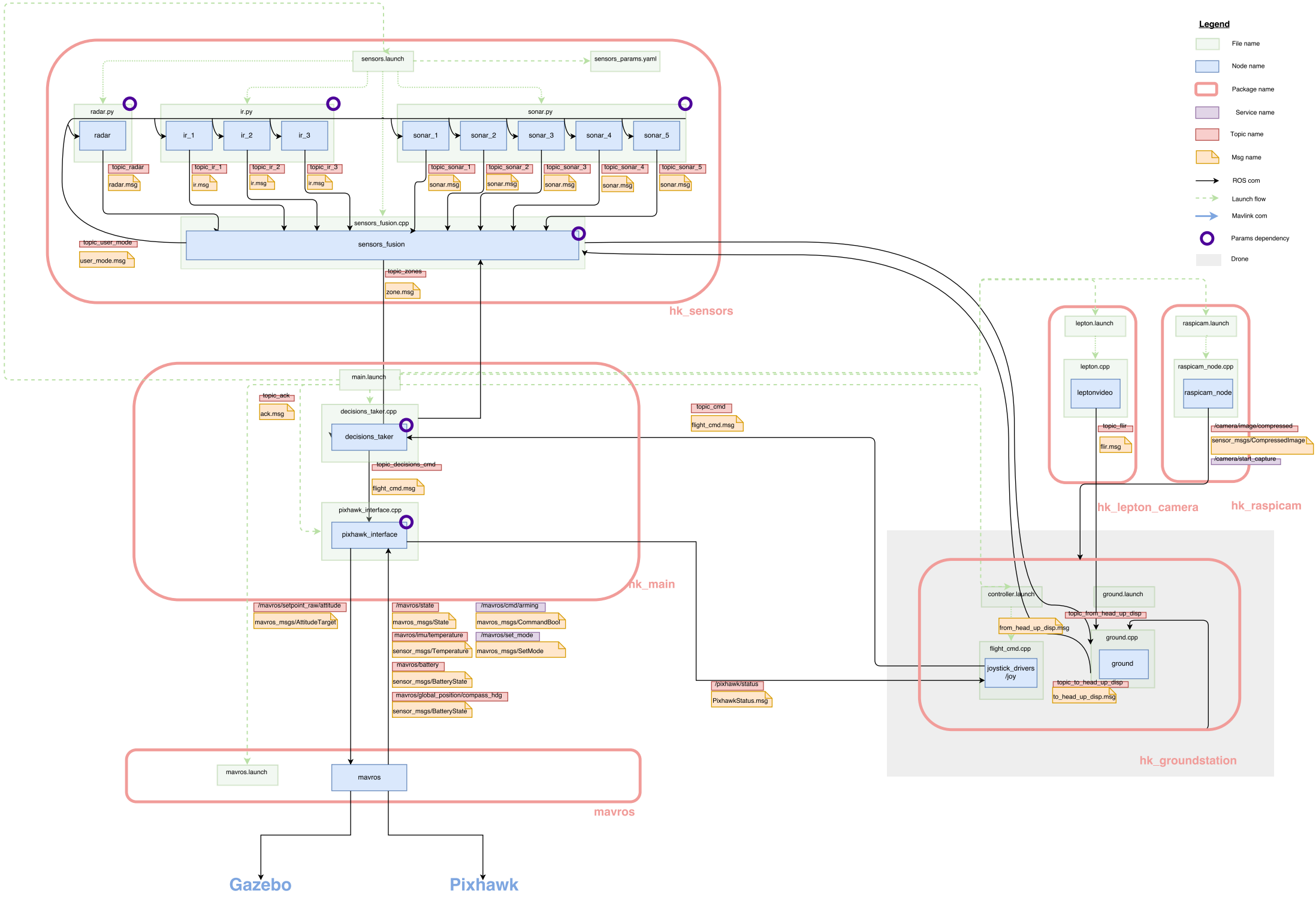
Contact Persons:

Erik.Bergdahl@Cybercom.com

Bartosz.Libner@Cybercom.com

Appendix B

ROS nodes diagram


Legend
File name
Node name
Package name
Service name
Topic name
Msg name
ROS com
Launch flow
Mavlink com
Params dependency
Drone
sensors.launch
sensors_params.yaml
radar.py
ir.py
sonar.py
radar
ir_1
ir_2
ir_3
sonar_1
sonar_2
sonar_3
sonar_4
sonar_5
topic_radar
radar.msg
topic_ir_1
ir.msg
topic_ir_2
ir.msg
topic_ir_3
ir.msg
topic_sonar_1
sonar.msg
topic_sonar_2
sonar.msg
topic_sonar_3
sonar.msg
topic_sonar_4
sonar.msg
topic_sonar_5
sonar.msg
sensors_fusion.cpp
sensors_fusion
topic_user_mode
user_mode.msg
topic_zones
zone.msg
hk_sensors
lepton.launch
lepton.cpp
leptonvideo
topic_flir
flir.msg
raspicam.launch
raspicam_node.cpp
raspicam_node
/camera/image/compressed
sensor_msgs/CompressedImage
/camera/start_capture
hk_lepton_camera
hk_raspicam
main.launch
topic_ack
ack.msg
decisions_taker.cpp
decisions_taker
topic_cmd
flight_cmd.msg
topic_decisions_cmd
flight_cmd.msg
pixhawk_interface.cpp
pixhawk_interface
hk_main
/mavros/setpoint_raw/attitude
mavros_msgs/AttitudeTarget
/mavros/state
mavros_msgs/State
/mavros/cmd/arming
mavros_msgs/CommandBool
mavros/imu/temperature
sensor_msgs/Temperature
/mavros/set_mode
mavros_msgs/SetMode
mavros/battery
sensor_msgs/BatteryState
mavros/global_position/compass_hdg
sensor_msgs/BatteryState
/pixhawk/status
PixhawkStatus.msg
controller.launch
ground.launch
topic_from_head_up_disp
from_head_up_disp.msg
ground.cpp
flight_cmd.cpp
joystick_drivers
/joy
ground
topic_to_head_up_disp
to_head_up_disp.msg
hk_groundstation
mavros.launch
mavros
mavros
Gazebo
Pixhawk

Appendix C

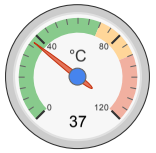
Thrust approximation

General	Motor Cooling: good	# of Rotors: 4 flat	Model Weight: 3000 g 105.8 oz	incl. Drive	Frame Size: 550 mm 21.65 inch	FCU Tilt Limit: no limit	Field Elevation: 500 m ASL 1640 ft ASL	Air Temperature: 25 °C 77 °F	Pressure (QNH): 1013 hPa 29.91 inHg		
Battery Cell	Type (Cont. / max. C) - charge state: LiPo 6000mAh - 65/100C - normal		Configuration: 6 S 1 P	Cell Capacity: 6000 mAh 6000 mAh total	max. discharge: 85%	Resistance: 0.0018 Ohm	Voltage: 3.7 V	C-Rate: 65 C cont. 100 C max	Weight: 167 g 5.9 oz		
Controller	Type: max 20A		Current: 20 A cont. 20 A max	Resistance: 0.01 Ohm	Weight: 25 g 0.9 oz		**Accessories**	Current drain: 0 A	Weight: 0 g 0 oz		
Motor	Manufacturer - Type (Kv): Gartt ML3510-600 (600) search... Prop-Kv-Wizard		KV (w/o torque): 600 rpm/V	no-load Current: 0.4 A @ 10 V	Limit (up to 15s): 430 W	Resistance: 0.0763 Ohm	Case Length: 28 mm 1.1 inch	# mag. Poles: 14	Weight: 120 g 4.2 oz		
Propeller	Type - yoke twist: DJI - 0°		Diameter: 9 inch 228.6 mm	Pitch: 4.7 inch 119 mm	# Blades: 2	PConst / TConst: 1.10 / 1.0	Gear Ratio: 1 : 1		calculate		

Load:

Hover Flight Time:

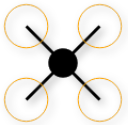
electric Power:



est. Temperature:

Thrust-Weight:

specific Thrust:


Configuration

Remarks:

Battery		**Motor @ Optimum Efficiency**		**Motor @ Maximum**		**Motor @ Hover**		**Total Drive**		**Multicopter**	
Load:	8.01 C	Current:	13.57 A	Current:	12.02 A	Current:	6.65 A	Drive Weight:	1740 g	All-up Weight:	3000 g
Voltage:	21.68 V	Voltage:	21.48 V	Voltage:	21.56 V	Voltage:	21.85 V		61.4 oz		105.8 oz
Rated Voltage:	22.20 V	Revolutions*:	12212 rpm	Revolutions*:	12339 rpm	Revolutions*:	9062 rpm	Thrust-Weight:	1.5 : 1	add. Payload:	765 g
Energy:	133.2 Wh	electric Power:	291.4 W	electric Power:	259.2 W	Throttle (log):	64 %	Current @ Hover:	26.60 A		27 oz
Total Capacity:	6000 mAh	mech. Power:	262.2 W	mech. Power:	232.9 W	Throttle (linear):	73 %	P(in) @ Hover:	590.6 W	max Tilt:	37 °
Used Capacity:	5100 mAh	Efficiency:	90.0 %	Power-Weight:	345.6 W/kg	electric Power:	145.3 W	P(out) @ Hover:	522.4 W	max. Speed:	43 km/h
min. Flight Time:	6.4 min				156.8 W/lb	mech. Power:	130.6 W	Efficiency @ Hover:	88.5 %		26.7 mph
Mixed Flight Time:	9.8 min			Efficiency:	89.9 %	Power-Weight:	196.9 W/kg	Current @ max:	48.09 A	est. rate of climb:	5.0 m/s

Appendix D

Components list

Components list

Supplier	Quantity	Art nr	Description	Price/unit (excl tax)	Price total (excl tax)
aerotenna	1	uSharp	uSharp	3300	3300
Adafruit	1	1294	SD Card 8Go	80	80
Adafruit	1	1031	IR distance sensor (20cm-150cm)	127	127
Adafruit	1	3055	Rapsberry Pi 3	320	320
Adafruit	1	979	Ultrasonic sensor	215	215
Adafruit	1	1568	IR Disance Sensor (100cm-500cm)	200	200
Rctech.se	1	46064	Pixhawk 2.1	1996	1996
Lawicel	1	ADA-1083	ADS1015 12-Bit ADC 4 Channel - I2C	99.2	99.2
Lawicel	1	CAB-11489	Raspberry Pi - GPIO Cable	10.4	10.4
Lawicel	1	SD-32GB-10	Kingston microSDHC 32GB Class 10	143.2	143.2
Hobbyking	1	426000015-0	Frame	360	360
Hobbyking	1	9114000020-0	Transmitter/hand controller	399	399
Hobbyking	6	505000002-0	Multirotor Carbon Fiber T-Style Propeller 9x5.5	58.2	349.2
Hobbyking	1	150000095-0	Turnigy High Quality 16AWG Silicone Wire 6m (Red)	34.5	34.5
Hobbyking	1	150000096-0	Turnigy High Quality 16AWG Silicone Wire 6m (Black)	34.5	34.5
Hobbyking	1	171000713-0	Turnigy High Quality 10AWG Silicone Wire 1m (Black)	25.8	25.8
Hobbyking	1	171000712-0	Turnigy High Quality 10AWG Silicone Wire 1m (red)	25.8	25.8
Hobbyking	2	15000030	HXT 4mm Gold Connector w/Pre-installed Bullets (10pcs/set)	60.75	121.5
RC Online	10	752-1797	JST GH Series, 1.25mm Male	3.974	39.74
RC Online	10	752-1731	JST GH Series 1.25mm female	1.052	10.52
RC Online	100	752-1725	JST SSHL Series Crimp	0.364	36.4
Hobbyking	1	9114000063-0	Turnigy iA6C PPM/SBUS	88	88
Hobbyking	1	9114000065-0	Turnigy iA6C Receiver Cables	12	12
Hobbyking	6	445000011	Carbon Fiber Propeller 10x4.7	62	372
OneDrone	1	dji-e600set-6	DJI E600 Kit	3769	3769
OneDrone	1	MAU-022	Mauch 2-6S BEC 5.35V 3A	229	229
OneDrone	1		Shipping from onedrone	173	173
Lawicel	6	SRF02	Ultrasonic Range Finder - I2C	110.19	661.12
Hobbyking	8	450000113-0	10 Inch Universal Multi-Rotor Propeller Guard	30	240
Hobbyking	3	171001279-0	HobbyKing™ Micro BEC 3A/5v	32.5	97.5
Hobbyking	6	505000003-0	Multirotor Carbon Fiber T-Style Propeller 10x5.5	67.5	405
Hobbyking	2	835000002-0	Stainless Steel 316 16mm Illuminated Off/On Switch	39	78
Adafruit	2	GP2Y0A710K0F	IR Distance Sensor	203.77	407.54
Elfa	30	144-02-241	molex connector	0.89	26.77
Elfa	30	144-01-90	molex connector	0.99	29.64
Elfa	14	144-02-260	molex connector	1.96	27.47
Elfa	14	144-02-170	molex connector	4.54	63.56
Elfa	6	144-01-998	molex connector	1.38	8.3
Elfa	1	300-85-264	Raspberry Pi 3 modell B	348	348
Rctech.se	1	62545	APM Power Module	158	158
Verktygshandlaren	1	118623	Fogskum 90 brandklassad Essve	395	395
Verktygshandlaren	1		Shipping	99	99
Sparkfun	1	KIT-13233	FLiR Dev Kit	2 106,86	2 106,86
Sparkfun	1	SEN-11610	LinkSprite JPEG Color Camera TTL Interface - Infrared	405,17	405,17
elfa	1	176-68-867	thermistor	19	19
elfa	1	176-68-924	thermistor	35	35
elfa	1	110-52-072	aluminum tape	341	341
rctech	1	49978	Airspeed sensor	396	396

Components list

Supplier	Quantity	Art nr	Description	Price/unit (excl tax)	Price total (excl tax)
	2		Raspberry pi power supply	67	134
rs-online	4	642-5001	Cable Mount IDC Connector	24.21	96.84
Farnell	1	191-2801-140	Flat ribbon cable	34.14	34.14
rs-online	2	827-7772	Pin header 2 row 40 way	21.54	43.08
rs-online	5	752-1793	JST GH Series, 1.25mm Pitch, 3 Way	2.614	13.07
rs-online	10	752-1737	JST GH Series 1.25mm Pitch 3 Way connector housing	1.469	14.69
rs-online	10	752-1797	JST GH Series, 1.25mm Pitch, 4 Way	3.974	39.74
rs-online	10	752-1731	JST GH Series 1.25mm Pitch 4 Way connector housing	1.052	10.52
rs-online	5	752-1807	JST GH Series, 1.25mm Pitch, 5 Way	4.98	24.9
rs-online	10	752-1740	JST GH Series 1.25mm Pitch 5 Way connector housing	0.963	9.63
rs-online	5	752-1800	JST GH Series, 1.25mm Pitch, 6 Way	4.618	23.09
rs-online	10	752-1743	JST GH Series 1.25mm Pitch 6 Way connector housing	1.141	11.41
RCtech	1	62545	Powermodule with XT	198	198
Elfa	2	300-37-322	Flex cable for Raspberry Pi Camera	32	63
Elfa	1	300-37-327	Raspberry pi camera module, wide angle	331	331
elfa	4	300-65-116	connector	30	30
elfa	1	300-37-327	camera	281	281
elfa	2	300-37-322	camera cable	54	54
elfa	2	173-81-798	sd card	344	344
elfa	1	300-97-225	shrinktube	116	116
rs-online	10	215-5821	Molex MINI-FIT JR. Series, Series Number 5569, 4.2mm Pitch, 4 Way 2 Row	5.66	56.6
rs-online	1	215-5887	Molex MINI-FIT Series, 5556 Series Number, Crimp Contact, Female,	153.11	153.11
rs-online	10	484-1754	Molex MINI-FIT JR. Series 5557 Series Number 4.2mm Pitch 4 Way 2 Row Female Straight Crimp Connector Housing	2.416	24.16
rs-online	80	820-1393	JST Cable assembly with Crimp Connectors	4	308.16
elfa	1	300-37-327	rpi	610	610
elfa	2	300-63-012	pi cameras	331	662
elfa	2	175-05-100	led	10	20
elfa	2	135-65-546	push button	15	30
KTH production	1	/	Printed frame	2,400	2400
Conrad	2	GP2Y0A710K0F	Long range IR sensors	235	469.96
KTH production	1	/	Printed Arms	1715	1715