UCLouvain

epl

École polytechnique de Louvain

Design process of repurposing discarded smartphones for quadcopter application

Addressing performance and integration challenges of postmarketOS

Author: **Jean-Baptiste Denoël**
Supervisor: **Jean-Pierre Raskin**
Readers: **David Bol, Sébastien Toussaint, Jean-Brieuc Feron, Nicolas Brusselmans**
Academic year 2023–2024
Master [120] in Electro-mechanical Engineering

Abstract

This thesis explores the innovative reuse of discarded smartphones to create a functional quadcopter. Given the rapid obsolescence and disposal of smartphones, this project aims to address environmental concerns by repurposing these devices as Unmanned Aerial Vehicles (UAV) flight control systems. Indeed, several smartphone's features in terms of sensing, processing, communicating can potentially be reused. The research objectives are to assess the feasibility, design, construction and performance of a smartphone-based UAV, while also demonstrating the sustainability benefits of reducing electronic waste.

The development process involves understanding the fundamentals of UAVs, including components and control systems. This includes selecting and integrating hardware, utilising IMU sensors (gyroscope, accelerometer, magnetometer) and algorithms to estimate orientation angles, implementing PID controller, and establishing Wi-Fi communication. To repurpose discarded smartphones, initial attempts using postmarkeOS (a Linux Open source software designed to support discarded smartphones) faced technical challenges in terms of hardware compatibility, leading to revised designs incorporating additional electronics for better performance. The final prototype employed two discarded smartphones: one as remote controller and the other onboard as receiver. In this design, internal IMU sensors, processor and Wi-Fi module of discarded phones are reused. Although partially operational, this prototype demonstrates that, under certain conditions, discarded smartphones can replace traditional UAV electronics, offering a cost-effective and environmentally friendly alternative.

Acknowledgement

I would like to express my deepest gratitude to my supervisor, Jean-Pierre Raskin, who gave me the chance to work on such an interesting and challenging project.

I would also like to extend my sincere thanks to Nicolas Brusselmans, who guided me throughout the work. His support, valuable advice, and constructive feedback greatly contributed to the success of this project.

Additionally, I am deeply thankful to my loved ones, for their constant encouragements. Their were a source of motivation throughout this challenging yet rewarding journey. I am also grateful to everyone who took the time and effort to read the final work and provide constructive feedback. Your contributions have been invaluable in refining and improving the project.

Lastly, I want to recognise the role of AI in this thesis for synthesising information and reviewing text formulation, which significantly enhanced the clarity and efficiency of my work.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

AOSP Android Open Source Project

BDLC Brushless Direct Current

CAD Computer-aided design

CF Complementary Filter

CPU Central Processing Unit

DOF Degree Of Freedom

ESC Electronic Speed Controller

E-waste Electronic waste

FC Flight Computer

FP2 Fairphone 2

GPIO General Purpose Input/Output

GPS Global Positioning System

IMU Inertial Measurement Unit

LCA Life Cycle Assessment

Li-Po Lithium-ion Polymer

LIDAR Light Detection And Ranging

NFC Near Field Communication

OS Operating System

PID Proportional, Integral and Derivative

PLA Polylactic Acid

pmOS PostmarketOS

SA3 Samsung galaxy A3

SA6+ Samsung galaxy A6+

UAV Unmanned Aerial Vehicles

USB - OTG Universal Serial Bus - On the Go

WEEE Waste from Electrical and Electronic Equipment

Chapter 1

Introduction

Every year, billions of smartphones are thrown away, most of them still in perfect working condition [1]. More than 70 % of these devices, will not be properly recycled and will end up in landfill or incinerate [2]. This phenomenon is both an ecological and sanitary disaster.

The massive number of discarded smartphones comes from their essential role in modern society, revolutionising the way we communicate, and transforming various aspects of daily life. Nowadays, smartphones are deeply integrated into our daily routines, with approximately 6.94 billion smartphones in use worldwide by 2019, representing 85% of the global population [3].

One significant issue with the disposal of smartphones is the loss of valuable resources, both in terms of raw materials and functional electronics. Since the release of the first iPhone in 2007, smartphones have experienced significant technological advancements [4]. They have evolved to include advanced feature like position tracking system, face ID, fast charging, larger displays and batteries, more cameras and sensors, and greater storage and memory capacity. Ultimately, what we are carrying in our pockets is a concentrated piece of advanced technology. The problem is that these devices typically stay with us for only two to three years before being replaced [5].

Alternatively, Unmanned Aerial Vehicles (UAV), typically called *"drone"* experienced a remarkable growth in recent years, boosted by advancements in technology and a wide range of potential applications. By their versatility and efficiency they offer a multitude of benefits and opportunities across various fields, including military, film-making, security, maintenance, delivery, health, environment monitoring, planet exploration and more [6]. This attention is also highlighted by an increasing number of research projects and mention in scientific paper as illustrated in Figure 1.1.

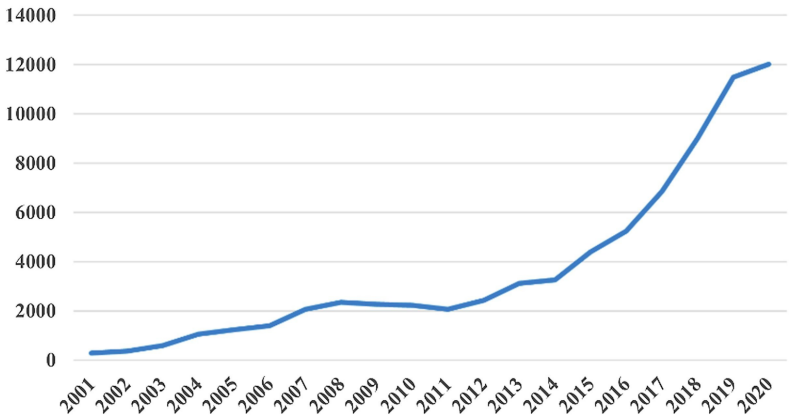


Figure 1.1: Growing trend line over time showing the number of scientific papers published using UAV or synonyms as keyword in the title, adopted from [7].

Moreover, this growing interest is also reflected in the UAV market. A 2022 study estimated a significant growth of the global UAV market, projecting it to increase from USD 31.70 billion in 2023 to reach USD 91 billion by 2030, exhibiting a compound annual growth rate of approximately 16.3% [8].

The particularity with UAVs is that they share a great amount of features with smartphones, such as geo-localisation, computing tasks, wireless communication, and sensing capabilities, among others. These drone's internal features require specifics electronics, which will contribute to the growing pile of electronic waste (e-waste), especially with their growing widespread adoption.

In this context, this thesis explores the innovative repurposing of discarded smartphones to design and build a functional *quadcopter*, a drone characterised by four motors. By reusing the technological capabilities of smartphones, we will study the possibility of replacing the onboard electronics (typically the *"brain"*) of UAVs. This approach aims to limit the production of new electronics and extend the functional lifespan of smartphones, thereby reducing e-waste, minimising environmental impact and demonstrating sustainable technology reuse.

Additionally, replacing the onboard electronics involves a complete redesign and implementation of the control system. This design process, from the construction of the prototype to the programming part, will enable us to analyse the potential limitations in term of hardware and software.
Moreover, following the sustainability goal and boosted by the widespread availability of smartphones, this project aims to create a cost-effective and accessible drone. Therefore, the design should be affordable and easy to build in terms of components, manufacturing and software. Open-source components and software will be privileged. Specifically, the use of postmarketOS, a Linux Open source software designed to support old smartphones, will be studied.

The main objectives of this thesis are resumed as follows :

- **Feasibility study**: Evaluate the feasibility of using discarded smartphones powered by postmarketOS as the core component for UAV development, completed by the use of accessible and open-source components.
- **Design and build:** Construct a quadcopter using the internal features of the discarded smartphone as flight control system.
- **Performance analysis**: Assess the performance and integration challenges of repurposing a smartphone in quadcopter application.
- **Sustainability** : Demonstrate the potential of repurposing electronic waste for advanced technological applications like UAV.

Finally, this thesis is presented according to the evolution process of the design. Especially, the development of the prototype can be divided into three parts. The first two chapters (after Introduction) provide the necessary knowledge for further development (gather in *fundamentals*). The Chapter 4 describes the *hardware* selection and the construction of the drone and finally Chapters 5-6-7 develop the *software* programming of the control system.
The global structure is illustrated in Figure 1.2.

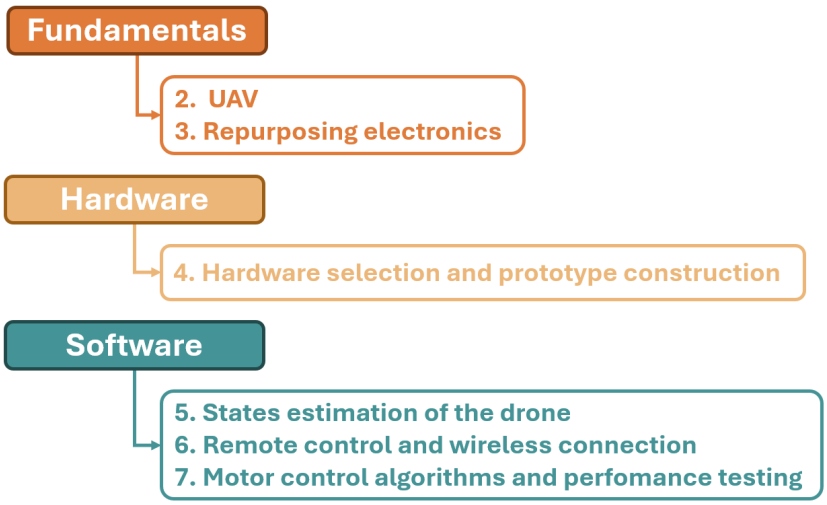


Figure 1.2: Structure of the thesis and breakdown of the prototype development into 3 main parts: fundamentals, hardware and software.

The description of each chapter, their role in the design process and construction of the thesis are outlined as follows:

Chapter 1 **Introduction.**

Chapter 2 **Unmanned Aerial Vehicles.** This chapter provides an overview of UAVs, their components, and flight control mechanisms. This chapter is essential for the future design and construction process of the prototype. Additionally an analyse of the various similar projects already developed in literature is realised.

Chapter 3 **Repurposing Electronics.** Before reusing smartphone's features into drone components we need to established which features could potentially be repurposed and if any type of discarded smartphones could be eligible for such process. Moreover, this chapter delimits the potential beneficial aspect of reducing e-waste by repurposing discarded smartphones. Finally a selection of an open source phone Operating System (OS) is made.

Chapter 4 **Hardware selection and prototype construction.** In this chapter all the hardware components are selected, from the discarded smartphones to the selection of the motors. Following the selection process, the prototype will be modelled and built.

Chapter 5 **States estimation of the drone.** Once all the components are selected and the drone is built, the design of the control system begins with this chapter. Here, the states (orientation status) of the drone are estimated, a process that requires the utilisation of the onboard sensors. This chapter will cover the integration of sensor data and the development of algorithms necessary to accurately determine the drone's orientation.

Chapter 6 **Remote control and wireless connection.** This chapter explains the setup and implementation of the remote controller as well as the wireless communication between the remote control and the drone.

Chapter 7 **Motors control algorithms and performance testing.** This last section of the prototype developments develop how the algorithms control the motors of the drone. Additionally, the flight performances are demonstrated through experimental setup. This process illustrates the successful integration of various components in terms of design and control.

Chapter 8 **Discussion.** Finally, the findings are summarised, and a discussion is provided regarding the design prototype and the implications of repurposing discarded smartphones for UAVs. This chapter also suggests future research directions based on the outcomes and insights gained from this study and prototype realised.

Chapter 9 **Conclusion.**

Chapter 2

Unmanned Aerial Vehicle

In this chapter a general overview of the UAVs is given, in term of model, components and control. Additionally an analyse of the various similar projects already developed in literature is realised.

2.1 Overview

UAV model can be classified in four main categories : single rotor, multi-rotor, fixed-wing and hybrid (see Figure 2.1). Each model has its advantages and disadvantages. Depending on these, certain models are better suited to certain functions than others. For example, multi-rotor UAV are dominating the global consumer and commercial market share. In contrast, the military market continues to largely employ fixed wing UAVs [9].

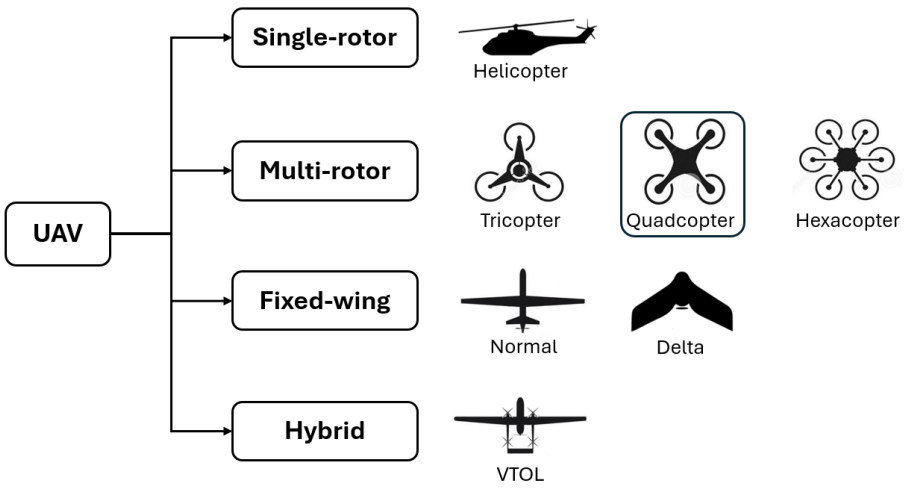


Figure 2.1: UAVs main categories, with the quadcopter highlighted.

For our project, we will focus on multi-rotor UAV and more specifically on **quadcopters**. This UAV model is characterised by its four motors and is attractive for its simplicity of design and control. Additionally, the popularity of quadcopters provide us with extensive resources to assist in the design as well as with analysing and comparing the achieved performances.

2.1.1 Components

Regarding quadcopters, the essential components are resumed as follows:

- **Frame**: This forms the base of the drone, providing support for mounting the other components. Quadcopters are characterised by a x-frame shape, typically constructed from plastic or carbon fibers.
- **Battery**: The battery provides the energy needed for the motors. The common type of battery is Li-Po batteries which offer a high energy density, lightweight design, and ability to deliver high current when needed.
- **Propellers**: Propellers are essential components of a drone, as they generate the thrust necessary to lift and manoeuvre the aircraft. They come in various sizes and shapes, with different configurations depending on the drone's design and intended use. They are characterised by the number of the blades, their diameter and the angle of inclination.
- **Motors**: Electric motors are used to spin the propellers to produce the thrust to lift the drone. Most of them are Brushless Direct Current (BDLC) motors which have better efficiency and higher durability than conventional brushed DC motors. BLDC motors are commonly made of permanent magnets mounted on the rotor and a stationary stator with coils of wire. They are alimented with a three-phase winding.
- **Electronic Speed Controllers** : ESCs are used to supply and control the motors. It basically consists of a network of transistors and by adjusting their switching frequency, the speed of the motors can be modified. The ESCs are alimented in PWM (Pulse Width Mudulation) signals, which vary the duty cycle to regulate motor speed (see Figure 2.2).
- **Inertial Measurement Unit**: IMU is a key element of the drone's flight control system. Usually composed of a gyroscope, accelerometer and magnetometer. There are used to measure the orientation of the drone required to develop autonomous control.
- **Remote communication system**: This system consists of a receiver and a transmitter. It is used to receive command inputs from a base unit and potentially send back information, such as video recordings and the drone's status. Depending on the application, various communication channels can be utilised, including different types of radio signals such as Wi-Fi.

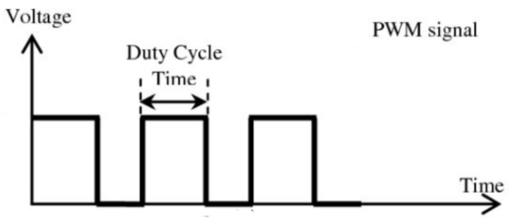


Figure 2.2: PWM signal over time, where the duty cycle is represented by the width of the high voltage period relative to the total period. A higher duty cycle results in a higher average voltage supplied to the motor, thereby increasing the motor speed.

- **Power distribution board**: This component distributes electrical power from the main battery to various subsystems such as motors, actuators and control electronics. It ensures that each part of the UAV receives the necessary power for optimal performance.
- **Others**: Additional components commonly found on drones include GPS, camera, various extra sensors such as LIDAR, ultrasound sensors, air pressure sensor and more. These components enhance the drone's capabilities and performances.

Figure 2.3 provides a comprehensive view of the essential hardware components of a quadcopter. Key elements include the BLDC motors combined with propellers, Electronic Speed Controllers, a GPS module, a wireless communication module, a power distribution board, a battery and the central processor with various sensors.

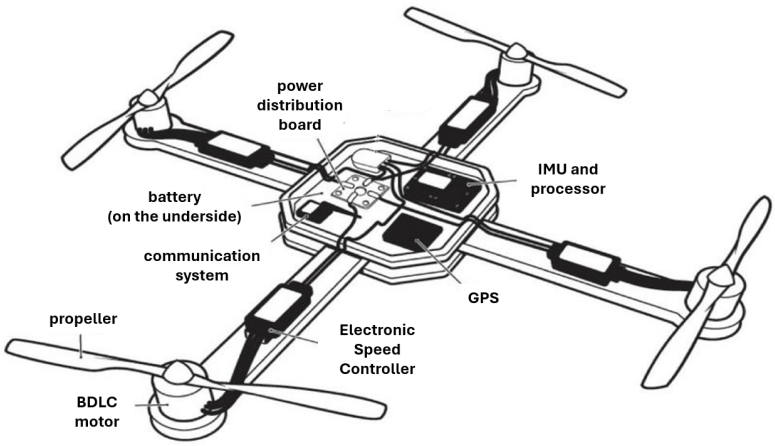


Figure 2.3: Hardware overview of a quadcopter resuming the key components, adapted from *UAV Fundamentals* [7].

To simplify further discussion, all the low-power electronics such as the processor, telemetry module (communication system), sensors, and GPS will be grouped under the term **Flight Computer** (FC), see Figure 2.4.

This module will be responsible for:

1. Determining the UAV's orientation and position.
2. Exchanging control command and status with the remote controller.
3. Processing information and controlling the motors.

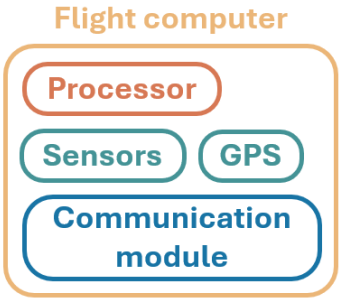


Figure 2.4: Flight Computer with internal features.

Finally, Figure 2.5 illustrates the connectivity between the different components of the quadcopter. The power electronics consist of four motors, each powered by an ESC on a three-phase winding. The low power electronics, represented by the flight computer, receive control commands remotely and control the ESCs with PWM signals. In this ecosystem, the quadcopter's battery powers the four ESCs as well as the flight computer. Since they require different power supplies, this is managed via a power distribution board, which is not represented in the diagram.

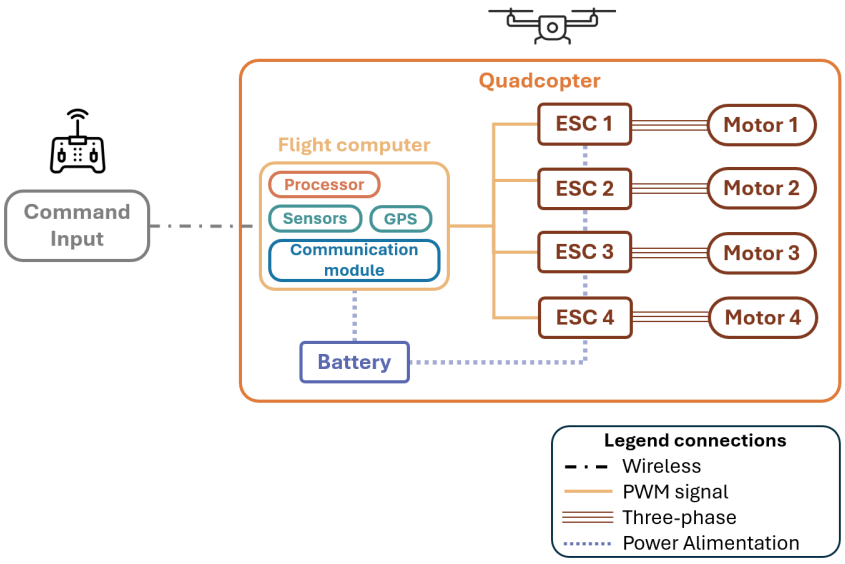


Figure 2.5: Quadcopter block diagram showing the connections between the command input, flight computer, ESCs, motors and battery.

2.1.2 Flight Control

The governing principles of the flight mechanics differs among UAVs but the goal remains the same : controlling the behaviour of the aircraft in an autonomous way. Usually, the UAV control is derived in two different control loop. A control loop, as illustrated in Figure 2.6 operates by comparing the measured states (obtained through the sensors) with the desired values (received from the command). Subsequently, the controller will try to eliminate or reduce this error by adjusting the motors. Many types of controllers exist, the best known is the PID controller, later developed in section 7.2. The three key blocks (Command input, Controller and Sensors) represented in Figure 2.6 will be individually described in the following Chapter 5, 6 and 7.

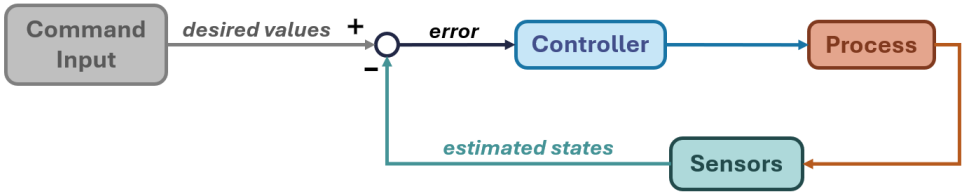


Figure 2.6: Control loop diagram illustrating the relation between the different blocks (input, controller, process and sensors).

First, ***the attitude control*** loop is fundamental to stabilising and manoeuvring the UAV. It is responsible for managing the orientation of the UAV relative to its inertial frame in terms of its roll (ϕ), pitch (θ) and yaw (ψ) angle. These three Euler angles correspond to the rotation along the three axes of the body frame (X^B, Y^B, Z^B) as referenced in Figure 2.7. The attitude control system typically involves the use of sensors like gyroscopes, accelerometers and magnetometer to measure the UAV's current orientation.

On top of this loop, there is the ***position control*** which is responsible for managing the UAV's location in 3D space, altitude and horizontal coordinates. This control ensures that the UAV reaches and stays at a specified position or follows a planned path. Additionally to an IMU, this system often relies on GPS data, pressure sensors or other positioning technologies aids to maintain precise control over the UAV's positioning.

The position control (outer loop) and attitude control (inner loop) are resumed in Figure 2.8. As observed, the control loops are interconnected. The position control loop generates pitch, roll, and yaw inputs (θ_i, ϕ_i, ψ_i) based on the error between the desired position (X_i, Y_i, Z_i) and the estimated position ($\hat{X}, \hat{Y}, \hat{Z}$). The attitude control loop then adjusts the motor outputs based on the error between the desired pitch, roll, and yaw angles (θ_i, ϕ_i, ψ_i) and the estimated angles $\hat{\theta}, \hat{\phi}, \hat{\psi}$.

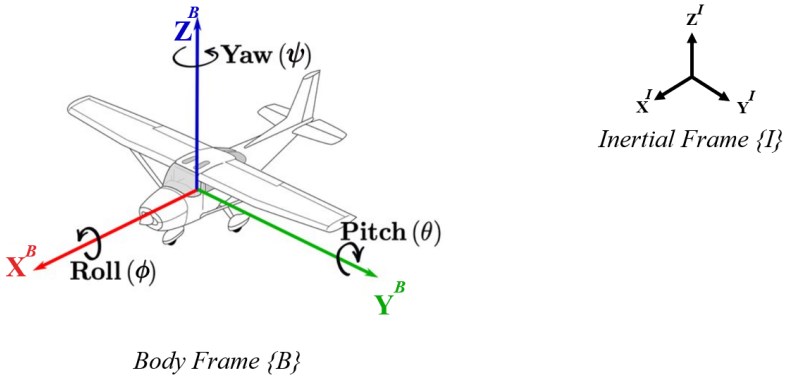


Figure 2.7: Roll, pitch and yaw angle representation, adapted from [10].

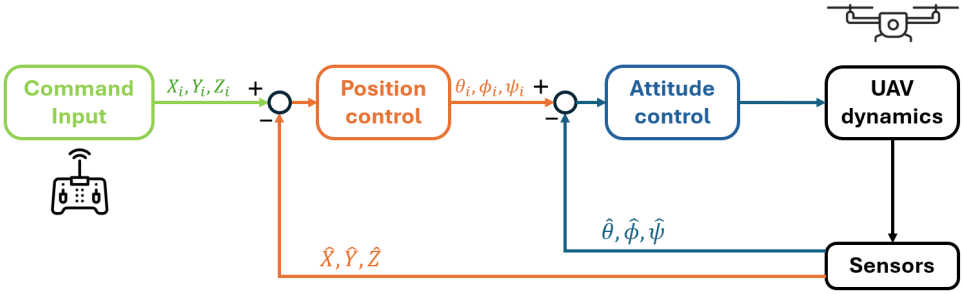


Figure 2.8: Block diagram of a cascade control loop applied for UAV control with the inner loop in blue controlling the attitude states (θ, ϕ, ψ) and the outer loop in orange controlling the position states (X, Y, Z).

2.2 Related work

This section provides an overview of various projects related to this master thesis objective: building an UAV using discarded smartphones. Here, a particular attention is given to the design process: the hardware and software used, the control algorithms, as well as the results and limitations of each project. These specifications regarding each of related work are summarised at the end in Table 2.1.

One of the first related application was proposed in 2009 [11]. In this report, researchers demonstrated the potential utilities of mobile phone. The phone used was a Nokia N95 with Symbian OS programmable in Symbian C++. The UAV was designed to orients itself using extracted features oh the phone's camera and an external GPS module. This primitive experiment is really limited but already shows some interests.

In 2015, researchers investigated the sensors availability within a smartphone to integrate computing and sensing capabilities to the UAV in a cost-effective way [12]. The hardware implementation is displayed in Figure 2.9. In this setup, the Android device sends sensor and vehicle data to the base station via Wi-Fi and receives control signals in return. The base station, processes operator inputs from the Xbox 360 controller and sends corresponding commands back to the Android device, which relays them to the *Arduino ADK* to control the quadcopter's motors.

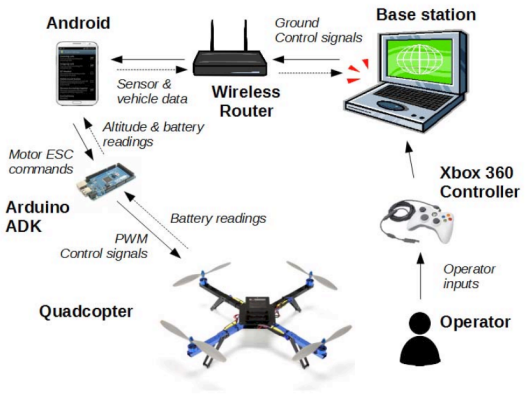


Figure 2.9: System overview from *UAV with smartphone sensors* [12]

The arduino ADK is a microcontroller, which is specifically designed to work with Android devices. As the motors are controlled in PWM signal its presence is required since smartphones are not capable of generating such signals. The smartphone's onboard accelerometer, gyroscope, and GPS are used to track vehicle movement and orientation, an external barometer chip is also connected to derive altimetry. In addition, the android device's camera is used to provide a small (240x180) live video stream during vehicle operation.

The quadcopter successfully hovered for short flights at low altitude with the GPS disabled. It came out that the sensor's data recorded (GPS and IMU) were consistent but the control algorithm implemented could benefit from additional sensor fusion techniques to allow for a more robust and stable flight system. Addition of a more accuracy altitude sensor would improve the control.

Afterwards, the researchs [13] and [14] demonstrated that the sensing, sensor fusion, control and planning can be provided by a off-the-shelf Android smartphone in GPS-denied environments like buildings. The developed cost-effective quadcopters orients themself by combining the data from the IMU (accelerometer and gyroscope) and the camera of the phone. The algorithms are executed through an Android app and communicate with the power hardware via a programmable Arduino microcontroller. Overall the drones were able to perform autonomous flight but in [14], latency and accuracy problem due to real time video treatment were reported.

In 2016, an analyse of the possibility of reusing discarded smartphones into drones is published [15]. In this study, they identified the availability of discarded smartphones and their remaining capabilities. They statistically demonstrated that 87.5% of discarded smartphone capabilities matched with the requirements of quadcopters. As proof of concept their demonstrated the validity of their research

by implementing several hardware and software application for quadcopters utilising smartphone features such as Wi-Fi and 3G communication, GPS, video transmission. The implemented system is represented in Figure 2.10. In this design, the states estimation and the control are made using dedicated commercial hardware. The smartphones are used to send and receive wireless data.

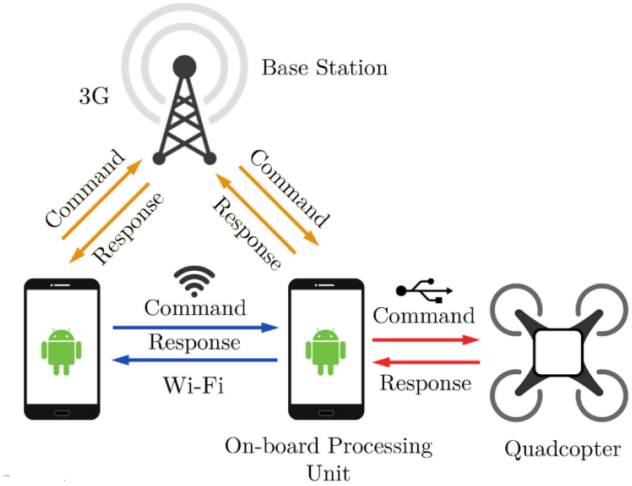


Figure 2.10: Repurposing smartphones utilities, in drone control, including sending/receiving control and response signals via Wi-Fi or 3G network, and exchanging USB commands and responses with the quadcopter. Adopted from *Reusing discarded smartphones* [15]

Regarding the results, the researchers reported that Wi-Fi is more suitable than 3G communication for short range real time control (less than 150m) and that 3G communication is more appropriate in long-range communication for non-real time tasks such as path planning. Moreover, the researchers faced software limitations, including challenges with real-time processing and a certain percentage (12%) of unsuccessful command transmissions. These issues were identified as being related to the constraints within the Android OS and the specific Android app developed.

Furthermore, in [16], the development of a quadcopter exclusively controlled by an onboard smartphone using its internal sensors (without the camera) and computational capabilities is proposed. The hardware architecture is made of an Arduino Mega ADK controlling the motors of the drone. The orientation angles estimation result from the fusion of accelerometer, gyroscope and magnetometer raw data. For the altitude estimation another fusion filter is implemented, coupling the barometer raw data with the linear acceleration from the accelerometer. Here, the developed controller use the derivative of the roll, pitch and yaw as inner loop control, see Figure 2.11.

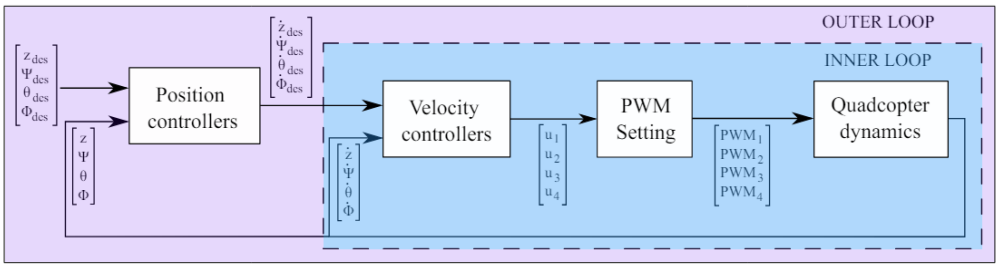


Figure 2.11: Inner and outer loops of the controller for position and velocity control implemented in *Cascade PID controller* [16].

The results of this prototype show software limitation such as real-time data processing. However the designed controller presents a good performance in disturbance rejection of the orientation and reference changes in the altitude and heading of the quadcopter.

Following this article, the same research group proposed an optimal tracking and robust controller, allowing the smartphone-based quadcopter to stabilise its altitude and accomplish missions like following a path through imposed waypoints [17]. Moreover, a fusion algorithm was also defined to improve the precision of the measured data.

Figure 2.12: The Phonedrone project [18]

Finally, it's interesting to mention the ambitious Kickstarter project of PhoneDrone Ethos, released in 2016, that proposed to turn any iPhone or Android phone in an autonomous aerial camera. The concept is to mount the user's phone on an external device which, with the help of a downloadable application, could be remotely controlled by another phone (as illustrated in Figure 2.12). According to the company, the external mounting case only serve as motor so all the computation and states estimation are done onboard the phone. Unfortunately the project never ended but it show the real potential that smartphones can create cost effective and accessible drones.

In conclusion, these different studies support the idea of repurposing a smartphone as part of a quadcopter, demonstrating the potential of various smartphone features for such applications. Despite certain timing, software and control limitations, most of the UAVs developed are performing correctly. Indeed, smartphones seem to be adapted to support drone's operation and control. These different projects highlight the versatility of smartphones in term of states estimation, com-

mand interface or running algorithms. Furthermore, most of the smartphones used in these applications ran on Android software and every design required extra electronics (e.g. a microcontroller to control the motors).

Finally almost all of the projects are focused on the development of quadcopters but each of them used a different control architecture, highlighting the diversity of possible options. Our project will differ by closely examining both the hardware and software design, as well as the control implementation of the drone. Additionally, we will provide an overview of the consequences of repurposing smartphones and their environmental impact.
The characteristics of the different project are resumed in the following Table 2.1:

Project	**Components**	**Achievements**	**Weaknesses**
Flyphone [11] - 2009	• Nokia N95 • Symbian O • GPS	Orient itself using phone's camera and external GPS	Limited capabilities
UAV with Smartphone sensors [12] - 2015	• Android device • Arduino ADK • Xbox controller • Barometer	• Consistent phone's sensor data utilisation (acc, gyro, GPS, camera) • Cost-effective • Successful short flights	Control algorithms needs improvements
Autonomous flight without GPS [13], [14] - 2013/2015	• Android device • Microcontroller	• Autonomous flight using smartphone's IMU, camera and sensor fusion • Cost Effective	Latency and accuracy issues
Reusing discarded smartphones [15] - 2016	• Discarded Android smartphones • Commercial hardware controller	• Reuse of discarded smartphones • Wi-Fi and 3G data transmission	Real-time processing challenges
Cascade PID controller [16] - 2017	• Android smartphone • Arduino Mega ADK	• Postion and velocity control loop • Phones's sensors states estimation • Good disturbance rejection	Real-time processing challenges
Tracking phone [17] - 2017	• Android smartphone • Arduino Mega ADK	• Improved altitude • Path following	Computational constraints
PhoneDrone Ethos [18] - 2016	• iPhone/Android smartphone • External mounting case	Cost-effective drone	Project not completed

Table 2.1: Summary and comparison of related UAV projects using smartphones

Chapter 3

Repurposing Electronics

In this chapter we explore the potential utility and limitation of repurposing electronics. Especially, we will study the repurposing smartphones and their suitability for replacing flight computers. Finally an overview of smartphone open-source software is developed.

Repurposing electronics involves reusing whole electronic devices or their components for new applications instead of recycling them to recover their raw materials. This process extends reusing, which is part of the 3 R's of sustainability : Reduce, Reuse, Recycle [19]. The interest of reuse has been driven by the high turnover rate of electronics, which are often replaced even though they are still in working condition [20]. Moreover, these discarded devices are contributing to the WEEE (Waste from Electrical and Electronic Equipment) which is classified as the fastest-growing waste stream in the world [21].

In terms of environmental impact, repurposing electronics offers notable benefits. A life Cycle Assessment (LCA) reveals that the manufacturing of electronics devices dominates their lifetime carbon footprint. This is especially true in devices with short use cycles [22, 23]. Therefore, the environmental impacts of reused electronics are significantly lower than those associated with new produced electronics. Consequently, repurposing electronics helps to decrease the demand for raw materials, and the energy required for transport and manufacturing, along with the emissions from these activities [24, 25]. Moreover, by reusing a device, its lifespan is extended, substantially decreasing the volume of e-waste and the amount of hazardous materials entering the environment [19, 26, 27] .

While repurposing electronics has many benefits, there are inherent limitations to consider. Not all components from discarded devices are compatible with new applications, which limit the scope of repurposing [28]. Over time, electronic components may be degraded, potentially compromising their performance in new settings. Furthermore, the environmental and economic benefits of repurposing are only maximised if it results in a reduction in the production of new electronics [29]. If new production continues at the same rate, the benefits are diminished. This

is also the case if new electronics are needed to complete the repurposed devices. Trying to save costs by using less energy and materials can also lead to a rebound effect. This phenomena happens when the saving creates an increase in production instead of a decrease, cancelling out the potential positive impact.

3.1 Repurposing smartphones

The high turnover of smartphones make them significant contributors of the WEEE. Shorter lifespans also increase the carbon intensity of these devices, since between 70% and 84% of the greenhouse gases associated with smartphones is released during the manufacturing process [22, 30, 31]. Furthermore, smartphones cannot be easily recycled by traditional way, even when they reach e-waste recycling facilities, these are often unregulated and hazardous [32]. Therefore, repurposing smartphones is a great example of the potential benefits of extending their use [33]. As discussed in this section, repurposing such devices also offer other advantages.

3.1.1 Reusing smartphone's features

Smartphones are particularly well-suited for repurposing due to their advanced technology and multifunctionality. In addition to an advanced screen and camera technology, modern smartphones are equipped with powerful processors capable of handling complex tasks as well as a variety of sensors and multiple connectivity options that can be utilised in different applications, making them highly versatile for various innovative uses [20, 34, 35].

To determine if all smartphones share the same components and to evaluate the potential for repurposing both older and newer models, a detailed analysis of their features was conducted. This research focused on the 10 best-selling smartphones each year from 2013 to 2023, representing 25% of the global market. This way, we survey *"new"* phones (manufactured less than 5 years ago) and "old" phones (manufactured more than 5 years ago). The data are collected from each phone's specifications available in [36]. The analysis categorised the features into three groups: *sensors*, *connectivity options*, and *advanced features*. The study does not take into account the presence of various elements such as the battery, touch-screen, microphone, speaker, camera that characterise smartphones and are considered included by default.

Sensors

Taking all the sensors referenced in the surveyed smartphones, it can be established that the proximity, light and accelerometer sensors are by default available (see Figure 3.1). However, the availability of the other sensors (magnetometer, gyro and barometer) is not universal and may vary over model release time (see Figure 3.2).

This variation seems more or less constant over time and can be assimilated to variability between the phone's brand or model.

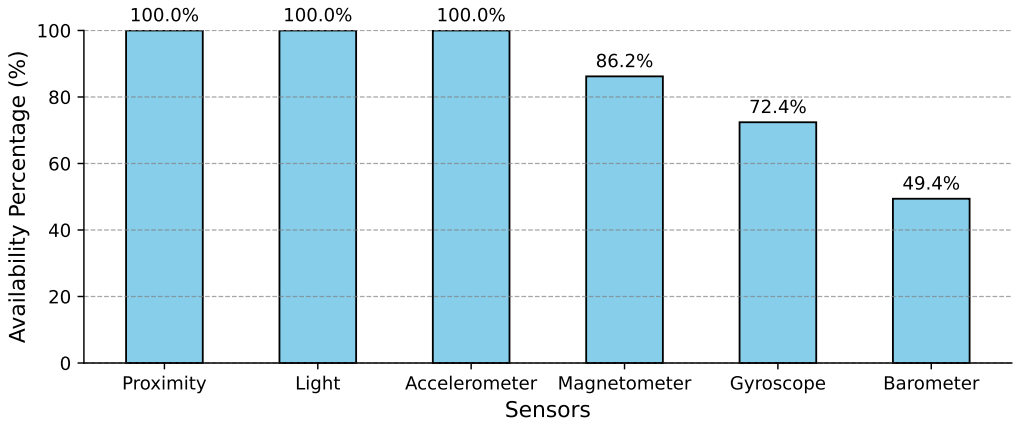


Figure 3.1: Sensor availability in surveyed smartphones, showing high presence for most sensors except the barometer. Data collected from [36].

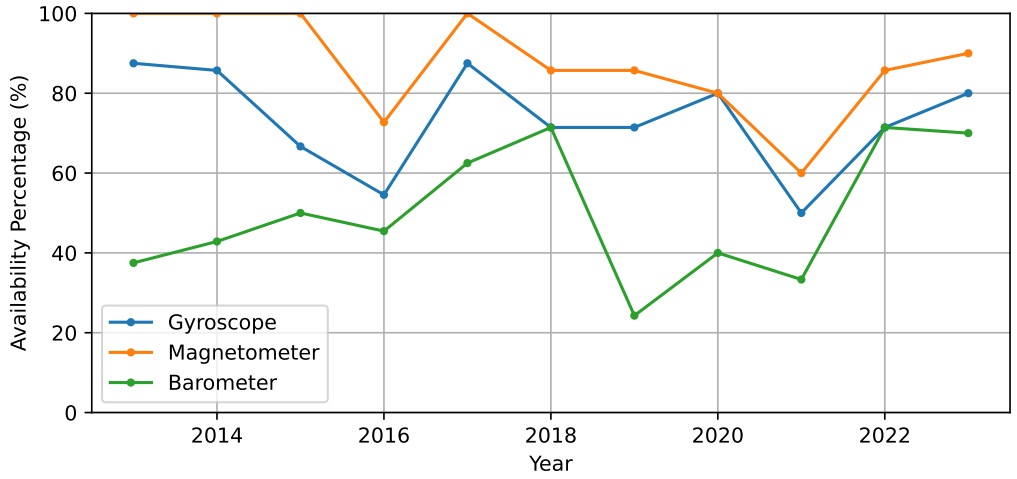


Figure 3.2: Evolving smartphone's sensors availability over time, highlighting a consistent presence in both old and new smartphones. Data collected from [36].

Connectivity options

As it can be observed in the Figure 3.3, almost all of the smartphones offer Wi-Fi, Bluetooth, and support for 2G/3G/4G networks. However, the availability of FM reveiver is declining over the recent years, while the 5G network support is increasing (see Figure 3.4). This trend can be attributed to evolving consumer preferences and technological advancements. 5G is a relatively new technology, and its availability is expending as it becomes more and more adopted. This phenomenon is similar to the expansion of the 4G network 10 years before. The decreasing of FM receiver

potentially results from the lack of interest in this technology observed in the last years [37].

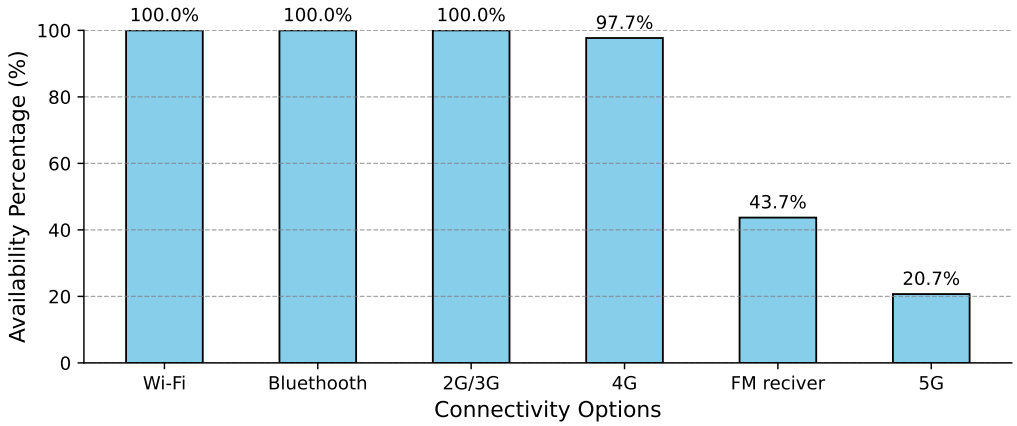


Figure 3.3: Surveyed smartphone connection option availability, showing high presence of Wi-Fi, Bluetooth, and 2G/3G, with lower availability for FM receiver and 5G. Data collected from [36].

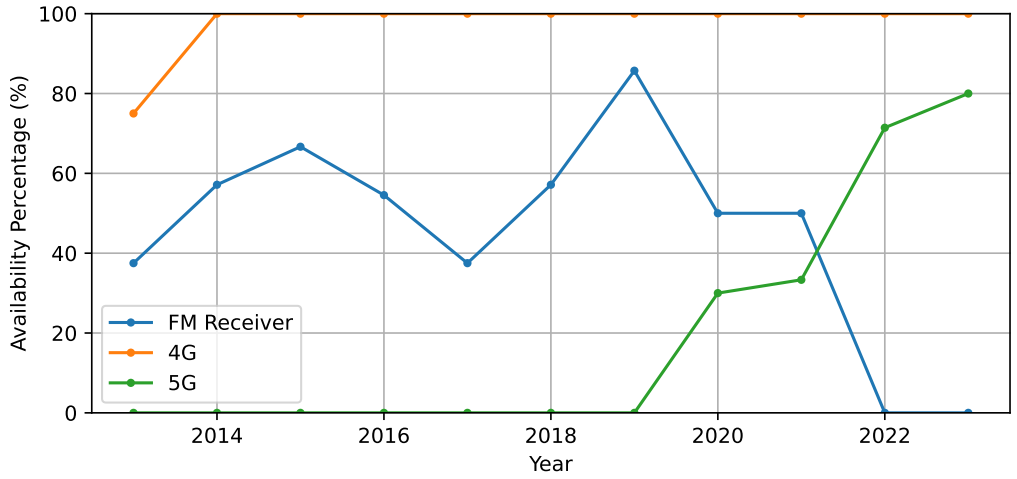


Figure 3.4: Surveyed smartphone connection option availability over time, highlighting high trends in 4G and recently in 5G while FM receiver's availability is decreasing. Data collected from [36].

Advanced Features

Finally, advanced features refer to the functionalities that extend beyond basic connectivity and sensors. These features are less equally spread among the different models as shown in Figure 3.5. More explicitely, as described by Figure 3.6, while the presence of some features, like Near Field Communication (NFC), remain constant, others such as wireless charging and Face ID are increasing. This rise can

also be attributed to the broader availability of these technologies. On the other hand, as Face ID becomes more popular, fewer phones include fingerprint sensors, indicating a potential replacement. Other technologies were introduced like heart rate sensing but quickly removed, probably due to customer interest.

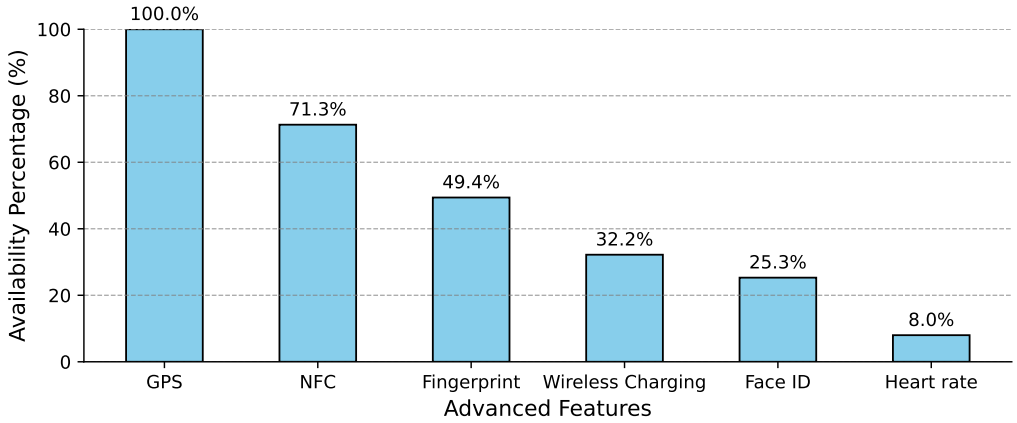


Figure 3.5: Surveyed smartphone advanced features availability, showing high presence of GPS and NFC, with lower availability for the other. Data collected from [36].

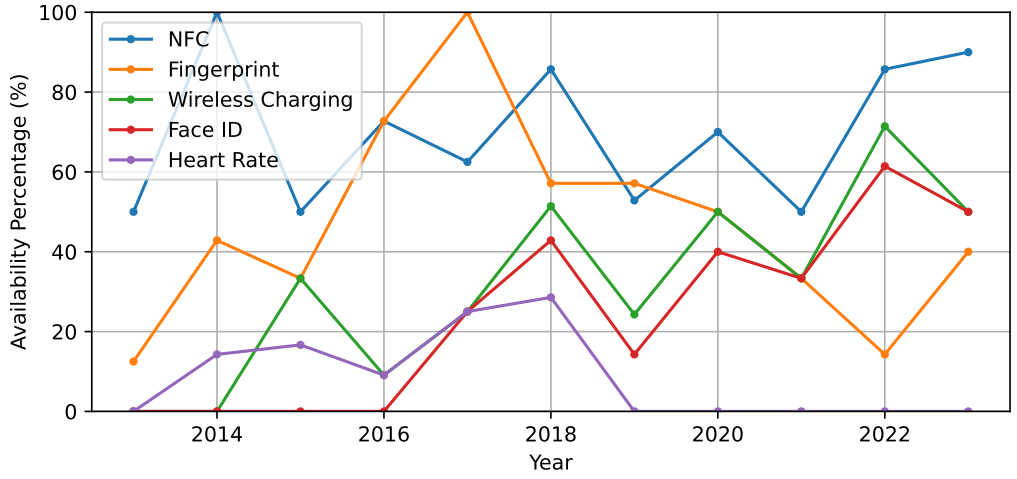


Figure 3.6: Surveyed smartphone advanced features availability over time, showing increasing trends in wireless charging and Face ID, a recent decrease in fingerprint and heart rate availability, and consistent presence of the NFC feature. Data collected from [36].

In conclusion, this study shows the wide range of available features in smartphones. While some technologies are consistently present across all devices (old and new), other fluctuate over time and across different brands. Looking at these variations some features remain stable over time while others show trends of increasing or decreasing availability. These trends vary regarding the global interest

and adoption of the technology from the customers.
Consequently, by taking a random discarded smartphone, there is still a high probability to find certain features as resumed in the Table 3.1.

Availibility ∼100%	Availibility >70%
Proximity sensor	Magnetometer
Light sensor	Gyroscope
Accelerometer	NFC
Wi-Fi	
Bluetooth	
2G/3G/4G network support	
GPS	

Table 3.1: Features highly likely to be available in smartphones produced between 2013 and 2023.

Additionally, the results of this survey may be heavily biased by the low percentage of representation of the telephone market. From one year to the next, the results may vary according to the brand of the smartphone surveyed and its market-range. For example the years when iPhones dominate the market, the tendency of feature's presence tend to rise. A more transversal survey characterising the smartphones in term of market-range could enhance the results.

Overall, studies confirm this trend and highlight the fact that the smartphone industry is evolving towards *"bigger and better"* [4]. In addition to the inclusion of new features inside smartphones the technology advancement of the smartphone's industry can be observed through the increasing performances and energy efficiency of the components. Examples include larger screens, more powerful batteries or processors, better cameras and greater storage. Unfortunately this progress also leads to increase the consumption of energy and critical materials during the manufacturing [38, 39].

3.1.2 Discarded smartphones overview

It's observed that users are changing their phone every two or three years [5]. In addition to the omnipresence of smartphones in today's society, this results to billions of discarded smartphones being disposed of every year [1, 40]. Therefore, smartphones present the advantage of being cheap electronics, easily accessible and exploitable for repurposing. However, it is crucial to ensure that they are reusable.

Unfortunately, there are limited scientific researches on this subject. Nevertheless, this question can be partially answered. First, it is interesting to note that studies show that most electronic components are designed to last longer than the actual average smartphone lifespan [41]. Furthermore, some research

explored the repurposing of discarded smartphone sensors and processors, demonstrating relatively low hardware limitations, even for older phones than 8 years [12, 20, 35]. Finally, there are a number of initiatives offering smartphone reconditioning services that consider most components to be still functional after initial use [42].

These results are highlighted in the Figure 3.7. In this survey, the main defective elements tend to be the screen (display, LCD, touchscreen) follows by the battery. Figure 3.8 confirms that these two components (screen and battery) tend to be main reasons why users are replacing their smartphones rather than defective electronics components like the features analysed before (see Table 3.1).

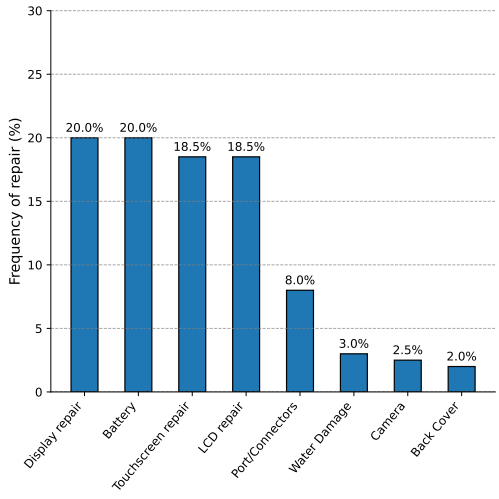


Figure 3.7: Frequency of common repair requests for smartphones on the market for 5 years showing a higher frequency for display and battery. Adapted from [43].

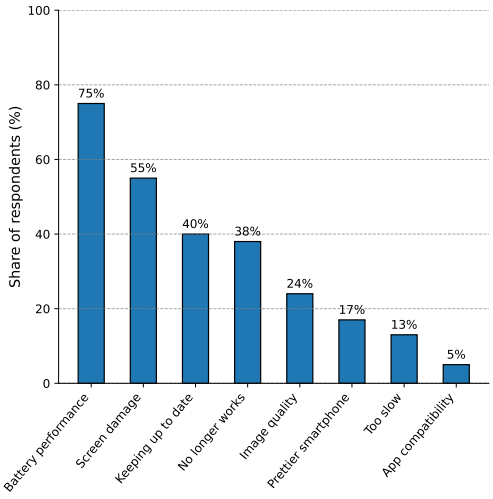


Figure 3.8: Survey: main reasons for upgrading smartphone showing higher reason for display and battery [2023]. Adapted from [5].

To conclude this section, repurposing smartphones appears to be highly beneficial. Their large numbers make them easily accessible and inexpensive. Furthermore, most phones contain numerous components and features that can be reused. Additionally, there is ample evidence that these components remain functional and effective for various applications.

3.2 Smartphones as UAVs

Smartphones possess several functionalities that make them suitable for replacing traditional flight computers in UAVs, introduced in Section 2.1.1. Indeed, there is a high degree of similarity between a flight computer and a smartphone. Both are equipped with components necessary for navigation, computational tasks, and communication, such as IMUs, GPS, high-performance processors, and connectivity options as resumed in Table 3.2.

In this scenario, aside from the barometer, almost all components found in conventional drones are present in smartphones (see Table 3.1). These results are confirmed by [15] who identified 87.5% of the discarded phone market match with the requirements of quadcopters. Additionally, the communication module of the flight computer can reuse various options from the discarded smartphone such as Wi-Fi, Bluetooth, or 2G/3G/4G. Finally, the phone's camera can be reused by the flight computer for state estimation as proposed in related work [13].

Components	**Discarded smartphone**	**Flight computer**
Communication Module	✓	✓
Accelerometer	✓	✓
Gyroscope	✓	✓
Magnetometer	✓	✓
GPS	✓	✓
Barometer	✓	✓
Camera	✓	✓

Table 3.2: Available conventional smartphone's features with Required flight controller requirement

However, there are several challenges. As discussed in the previous Chapter 2, building a UAV from smartphones requires additional electronics such as microcontrollers, ESCs, motors, and batteries. This could reduce the benefits of repurposing smartphones if it necessitates the production of a significant amount of new electronics. While the current trend in UAV deployment presents an opportunity for repurposing electronics (see Figure 1.1), the cost-effectiveness and large availability of discarded smartphones could also lead to even greater production of UAVs (rebound effect), potentially negating the anticipated benefits [44].

3.3 Phone Operating System (OS)

By repurposing smartphones we want to reuse their components and other functionalities. This can be achieved by reusing the different components individually but this process is really complex and there is a risk of damage during recovery. Moreover, some of the elements require particular attention in terms of power supply and communication. Therefore, it is easier to use the phone in its entirety and communicate with the components via the operating system.

The OS is a system software that manages all the resources of the computer. As shown in Figure 3.9, the OS acts as an interface between the hardware and the software. On top this layer, application and programs are developed enabling user interaction. Currently, the smartphone OS market is dominated by Android (Google) with nearly 70% and iOS (Apple) with about the last 30% of the market

[45]. Therefore, the software's and updates on the majority of the smartphone are managed by these two companies.

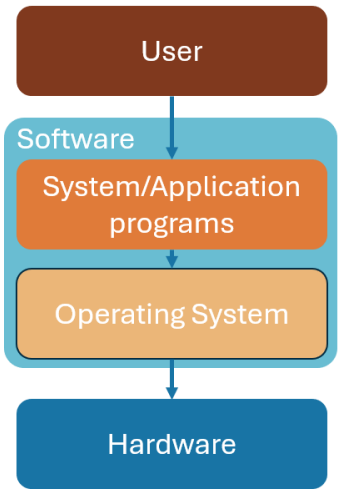


Figure 3.9: Operating system simplified architecture in 3 main layers : user, software and hardware.

Again, repurposing smartphones is not without any constraints. Indeed, all of the targeted phones are second-hand or obsolete. The main drawback of these (discarded) devices is that, after a few years both Android and iOS tend to stop providing updates for older devices, leaving them vulnerable to security flaws and applications incompatibility. Hence, using one of these operating systems can potentially lead to various restrictive problems over time.

This lack of support can be attributed to economical considerations. Due to its architecture, the operating system is specific to each devices, necessitating updates to be adapted to each specific version of the OS which can become very costly for the developers. Furthermore, the motivation for keeping old phones updated decreases as the number of users is going down. In parallel, by limiting software's updates for older models, they indirectly encourage users to purchase the latest smartphone [43].

The Android ecosystem is built on an adapted Linux kernel and a mix of open-source and proprietary elements. These proprietary elements such as closed-source drivers and Google services, are integrated by manufacturers to optimise Android for their specific hardware. Moreover, the Android's openness leads to a diverse ecosystem with thousands of devices models from various manufacturers. This diversity results in fragmentation: different devices run different version of Android which leads to infrequent, non-general updates. Overall, depending on the brand, a software support of 2 to 7 years can be expected [46, 47]. While for iOS, the ecosystem is completely closed, leaving Apple to have a full control on both software and hardware. They usually provides up to 7 years of software supports [48].

To overcome this issue and to have a full control on the software, different open source operating system have been developed with different purpose. All of them are primarily design for Android devices (since the closed aspect of iOS makes customisation impossible). Here are the most popular :

- **LineageOS** is a custom Android distribution launched in 2016. It's based on the Android Open Source Project (AOSP) and provides personnalisation, privacy and security. LineageOS offers support for a wide range of recent devices, extending their software supports and providing users with greater control over their devices. Moreover, users can still benefit from the Android environment except for the Google services. The downside are the limited official support with older devices and the occasional compatibility issues with certain apps and features [49].

- **Ubuntu touch** is a mobile version of the Ubuntu operating system created in 2011. It emphasises convergence, aiming to provide a seamless experience across different devices, such as transitioning between a smartphone and a desktop computer. However, this OS is limited by the small app ecosystem and hardware compatibility issues. Currently only a few devices are compatible and stable [50] .

- **PostmarketOS**, (pmOS) is designed with long-term support in mind, aiming to extend the lifespan of smartphones beyond the manufacturer's typical support period while maintaining performance and security. It is based on the lightweight Alpine Linux distribution which offer a full control over hardware resources and usage. Although this OS is still in its early stages of development, it faces challenges with hardware compatibility, resulting in a limited number of compatible Android devices.[51] .

3.4 PostmarketOS

From the different software developed earlier, postmarketOS emerged as the best option for our project. Its focus on long-term support and sustainability allow us to act on a great number of discarded smartphones including the oldest. Additionally, pmOS benefits from the Alpine Linux toolbox, providing control over the repurposed hardware and the necessary tools to develop the required code. Furthermore, its active development and growing community offer reassurance for ongoing improvements.

The architecture of pmOS completely dispenses the Android build system. Instead of building a unified system system image for each device, the entire OS is divided into small packages. These same packages can be installed on all devices that share the same CPU architecture. Ideally there is only one device package, device-specific parts are kept as minimal as possible. On top of that, different

possible graphical environment (user interface) can be developed. This modular design allows for greater flexibility and easier maintenance across different devices.

Furthermore, postmarketOS tries to replace the kernel provided by the device vendor (the downstream kernel), with a version close to the mainlined version of Linux. This requires to rewrite the necessary drivers to interact with the hardware. Through community effort there is a growing number of devices that are compatible but currently the OS is limited to a small number of compatible smartphones. The overall architecture is represented in Figure 3.10.

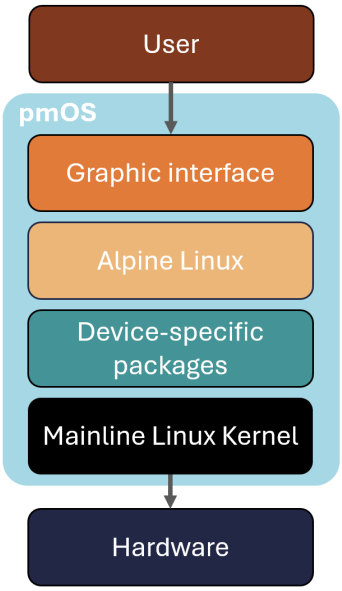


Figure 3.10: Visualisation of the postmarketOS architecture, illustrating the interaction between the different layers. Based on [51].

Chapter 4

Hardware selection and prototype construction

This chapter emphasises the selection of components for developing a UAV prototype and discusses the challenges faced in repurposing discarded smartphones, leading to the consequent evolution of the design.

The purpose of this project is to design and build a quadcopter using discarded smartphone. As developed in Section 3.2, discarded smartphone can easily replace the flight computer module of the quadcopter allowing us to reuse many internal features. Moreover, we ended up by selecting postmarketOS as the ideally Operating System to interact with discarded smartphone (see Section 3.4).

As with most technical projects, not everything goes according to plan, and modifications are often necessary to compensate for any technical problems. This was also the case for our prototype, which, as shown in Figure 4.1, underwent a redesign due to a series of technical challenges that modified the progress of the original design.

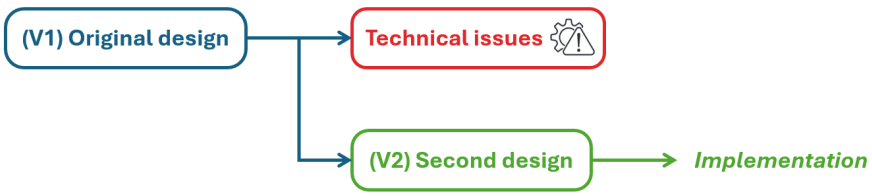


Figure 4.1: Design process evolution starting with the original design V1 and ending with the second design V2 due to technical issues.

Consequently, this chapter begins with a description of the initial design attempted (V1), Section 4.1. Subsequently, we detail the first implementation effort in Section 4.2, which unfortunately ended in failure, but which will enable us to restart development with a second design (V2), presented after in Section 4.3.

4.1 Original prototype design (V1)

The first version of the developed quadcopter is illustrated in Figure 4.2. This diagram resumed the main components necessary to fly a drone developed in Section 2.1.1.

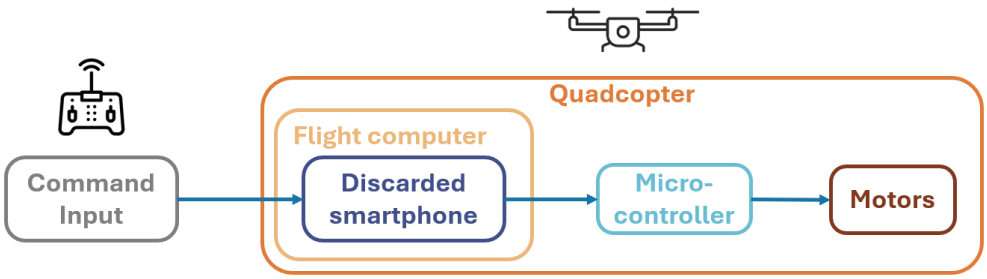


Figure 4.2: Prototype Version 1 block diagram illustrating the main components and interactions.

In this design the flight computer is replaced by a discarded smartphone performing all the necessary control tasks (already introduced in Section 2.1.2). The **state's estimation** are done using the internal sensors and GPS, **algorithms control** are running on the onboard processor, and **command inputs** are received from the remote controller using Wi-Fi for short-range communication and 4G for long-range. Since the smartphone can't generate PWM signals, a microcontroller is still required to interface with the motors[1].

In this design, the position and attitude control are implemented, therefore, a consequent number of internal features of the discarded phone are repurposed like Wi-Fi interface, 4G support, processor, sensors and GPS suggesting a positive impact in term of reducing electronic waste.

4.2 OS implementation

To develop the prototype, the first step consists in the selection of the discarded smartphone and the implementation of postmarketOS, the selected phone OS (Section 3.4).

PostmarketOS is implemented on three different discarded smartphones (see Figure 4.3). The installation process consists of rewriting the phone's memory and replacing the actual Android OS with the postmarketOS image. Unfortunately, as described bellow, none of them meet the necessary requirements.

[1]In this and subsequent representations, the ESCs have been grouped into the "motors block" to simplify the diagram.

Figure 4.3: Picture of the three selected discarded smartphones with pmOS installed with the minimal graphic interface selected.

- **Fairphone 2 (2014)**. With postmarketOS, the smartphone is able to communicate using WiFi, 4G or USB OTG communication but unfortunately there is no access made to the internal sensors. This is due to the lack of the dedicated drivers (the specific program that interacts with the sensors). Developing these drivers requires a deep understanding of the Fairphone's Soc (System-on-a-Chip) architecture. Preliminary work has been done by developers to ensure sensor accessibility, but further effort is needed to stabilise the connection and integrate the code into the mainline.
- **Samsung A3 (2015)**. With pmOS, this smartphone is able to have access to the communication module as well as the embedded sensors (accelerometer and magnetometer). However the connection with a external microcontroller failed (USB OTG). This failure comes from the new OS. It could be device-specific or universal for all these models.
- **Samsung galaxy A6+ (2018)**. This phone provided access to WiFi and 3G/4G features, along with two essential sensors, the accelerometer and gyroscope. Nevertheless, despite these capabilities, we encountered hardware issues preventing communication between the phone and a microcontroller (USB OTG).

These specifications for each developed smartphones are summarised in Table 4.1.

	WiFi	3G/4G	Bluetooth	GPS	Acc	Gyro	Magn	Camera	USB OTG	GPIO Keys
FP2	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗
Sam A3 (2015)	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Sam A6+	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓

Table 4.1: Features accessibility of three different phone with postmarktetOS

An initial conclusion can be drawn here, concerning the software used. Although postmarketOS offers many advantages for our application, it is still in its early stages, involving various software compatibility problems. As future work, additional research need to be made, exploiting other smartphone model. As a result, the current pmOS version does not allow us to fully replace the flight computer of the quadcopter and our design need to be reviewed.

4.3 Prototype design second version (V2)

In order to replace the flight computer, the new design will focus on the reusing of a Fairphone 2 (the only tested phone with USB OTG available) along with external sensors and a microcontroller. These extra electronics compensate the unavailable sensor features of the Fairphone. Consequently, the tasks of the flight computer are distributed into three components as follows :

- **Fairphone 2**: which serves as communication interface between the remote control and the microcontroller. It will reuse the Wi-Fi module of the smartphone as communication support.
- **External sensors**: are used to determine the states of the drone and sending them to the microcontroller.
- **Microcontroller** controls the quadcopter's motors. This control results of control algorithms that take as input the received commands from the remote controller, along with the quadcopter's states estimated through the external sensors.

Furthermore, the Fairphone's design offers additional advantages specific to UAV development. Its easy dismantling allows us to isolate the main board while removing unnecessary components (screen, battery, camera etc) thereby reducing the quadcopter's weight and space, as shown in Figure 4.4.

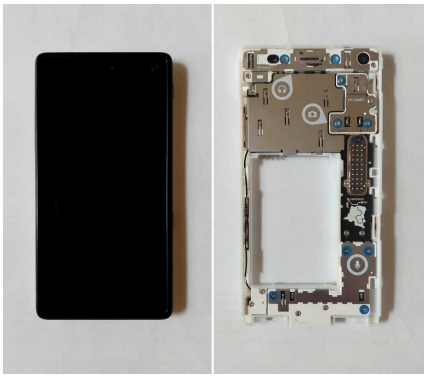


Figure 4.4: Fairphone 2 before and after dismantlement.

In this design, we will focus on the attitude control of the quadcopter coupled with the altitude control [h] (which is part of the position control, Section 2.1.2). As a result, the command inputs are resumed as pitch, roll, yaw and height.

Additionally, as we are able to have access to the sensors and connectivity options of the other two phones, the project will also explore using one of them as remote control command. This approach will help to determine whether internal sensors can be effectively utilised with postmarketOS, potentially validating their use as a futur part of a flight computer. Specifically, based on the phone's orientation we will try to use its internal sensors and determine a pitch, roll and yaw input control. In our case, the final selected smartphone is the Samsung A3 who will sent control input on Wi-Fi to the FP2, more detail in Chapter 6. This second quadcopter version is represented in the Figure 4.5

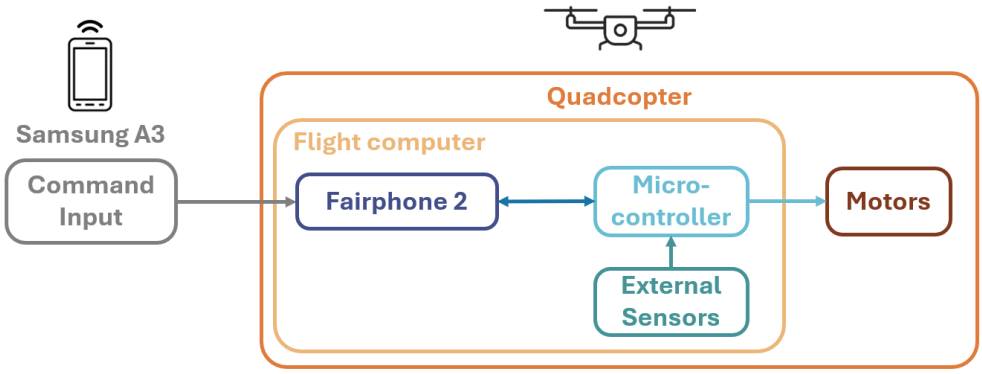


Figure 4.5: Prototype design v2 block diagram where the remote controller is replaced by a Samsung A3 and the flight computer is now composed of a Fairphone 2, external sensors and a microcontroller.

During the development of the original design, our inability to access certain smartphone features prevented us from using the discarded smartphones as intended. Consequently, these desired features were substituted with equivalent electronics. Figure 4.6 completes the evolution design diagram illustrated before (see Figure 4.1).

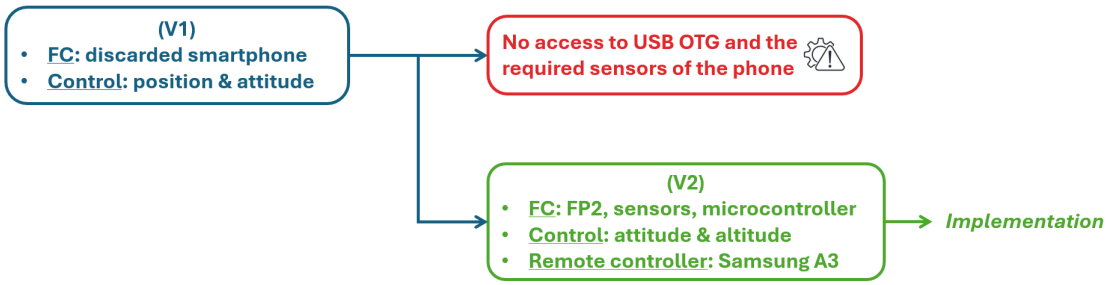


Figure 4.6: Evolution design with additional electronics, two discarded smartphones and simplified control.

This approach is motivated with the anticipation that if this design's implementation is successful, the implemented control could be easily be mapped into

the original design, assuming the access to the internal features solved. Moreover, the interaction with the sensors of the Samsung A3 for remote control command using pmOS will give us additional insights about the feasibility of the original design.

In the end, for the secondary design, two discarded smartphones are repurposed:

1. The FP2 (onboard the drone) which transmits wireless data via its Wi-Fi module is used.

2. The Samsung A3 which serves as remote controller. Its internal sensors along its Wi-Fi module are used.

4.3.1 Sensing electronics

As developed before (Section 4.3), the new design requires a microcontroller and additional sensors. The criteria for selecting these components focus on reducing the environmental impact of the produced drone and using cost-effective and popular components. Studies have shown that the carbon footprint of electronic components increases with their complexity and multi-functionality [52]. As result, the chosen microcontroller is an *Arduino Nano*, a simple popular and open source microcontroller. To increase the number of pin connections, an *extension board* is added.

For the additional sensors, the purpose is to use similar sensors as the main available in discarded smartphones (see Table 3.1). Since the quadcopter needs to only estimate it's attitude and altitude states, an Arduino *9 DOF IMU* is used which includes an accelerometer, a gyroscope, and a magnetometer. Additionally, for the altitude estimation, a *barometer* is included. Although this sensor is not commonly found in conventional smartphone (Table 3.1), its presence can help to estimate the drone's altitude. Unfortunately, these extra sensors differ from those typically found in smartphones but meet the same specifications and are compatible with the selected microcontroller.

These sensing electronics are represented as follows (Figure 4.7).

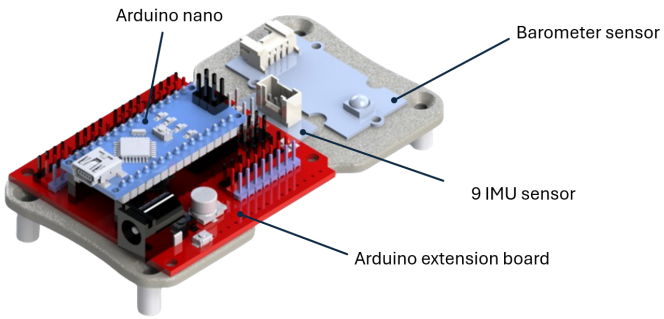


Figure 4.7: Onboard sensing electronics components, developed 3D CAD model.

4.3.2 Power electronics

Concerning the four motors of the drone, their selection involved careful consideration of various factors to ensure optimal performance and efficiency. Two key criteria were taken into account:

- **Thrust-to-weight ratio** is the ratio between the maximal thrust of the combined motors and the estimated weight of the drone. With an estimated weight of ±1000g, a minimum ratio of 2:1 is necessary to guaranty the drone to lift itself in the air. Therefore the required thrust should be at least 2 kg so 500g per individual motor.
- **KV rating** $[\frac{RPM}{V}]$ is the number of revolutions of the motor per minute when one volt is applied to the unloaded motor. The higher the KV, the faster is the motor but lower is the torque. In general, higher KV motors are coupled with smaller propeller and vice-versa.

Additionally the current of the ESC current rating must be higher than the maximum current drawn by the motor, and the drone's battery have to match the voltage [V] of the ESC. The battery should also provide sufficient flight time based on its capacity [mAh] and deliver adequate power as indicated by its C rating.

Considering all these requirements and promoting cost-effective and available electronics, the chosen components are detailed in the table 4.2. These components, such as the motors and ESCs, are acquired through purchase, while others like the battery, are obtained through recovery.

	Technical characteristics
Motor	1400 KV, max 20A
Propeller	8 inches diameter
Max thrust	2.2 kg
ESC	30 A
Battery	Li-Po, 11.1 V, 2200 mAh, 50 c
Size	45 mm x 45mm (without propellers)
Weight	1.150 kg

Table 4.2: Technical characteristics of the Power electronics components (motors, ESCs and battery) of the quadcopter

4.3.3 Schematics

The electronic circuit of the final design with the specified components is detailed in Figure 4.8. In this version a Samsung A3 sends command inputs to the Fairphone 2 via Wi-Fi. Then the Fairphone transfers these commands to the Arduino nano using USB (OTG) connection. In addition, the Arduino estimates the drone's state data from the 9 DOF IMU and the barometer. Based on the received data, the

microcontroller is able to command the ESCs (via PWM signals) and therefore the 4 motors of the drone. The motors are alimented with the battery through the ESCs. In this design, the Fairphone 2 retains its battery, allowing it to power the Arduino, which in turn powers the sensors.
A potential upgrade would be to aliment the Fairphone and the Arduino with the main battery. This way, the FP2's battery can be removed and the drone's weight is optimised leading to longer flight time.

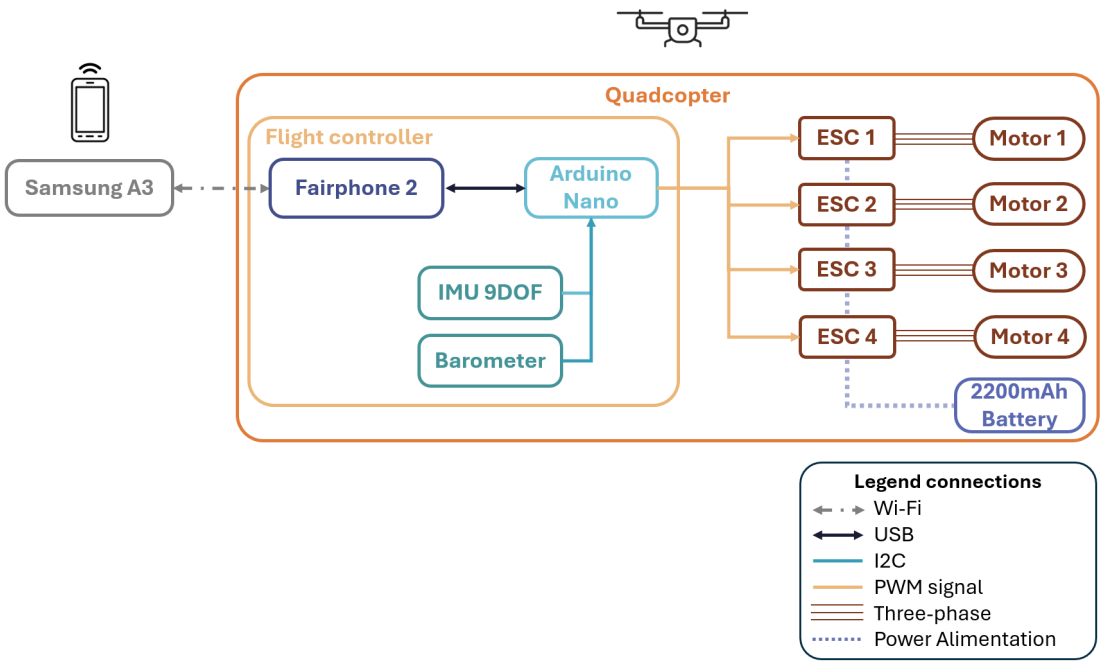


Figure 4.8: Final Prototype block diagram illustrating the connectivity between the selected components

4.3.4 Mounting of the quadcopter

Now that the components and their interconnection are established, the drone can be mounted, starting with the frame.

After some iterations, the frame of the drone is made inspired by the DJI F450 drone. This commercial model suited our application and its design made it easy to build by ourself. The four arms are 3D printed with 35% infill plastic PLA while the center parts are made of laser cut plywood. These different part are assembled using M3 nuts. 3D printing enables the production of complex, lightweight parts, while laser cutting provides easy creation of 2D file. Their use are also motivated by the goal of making the design cost-effective and accessible.

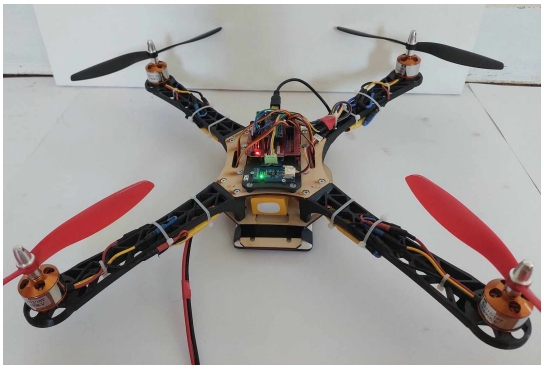
As design support for the integration of the different components, a 3D CAD (Computer-aided design) model is made as shown in Figure 4.9.

Figure 4.9: 3D developed CAD model of the quadcopter, expressing power electronics (motors, ESCs and battery), the sensing electronics on top of the drone, the FP2 below and the frame (4 arms maintained by 2 wood plate).

Specifically with this design:

- The sensing electronics are mounted at the centre of the drone. Aligning the body axis of the aircraft with the sensitive axis of IMU maximises their efficiency. Additionally, by centering the sensors, especially the accelerometer, the mechanical noise (vibration) created by the motors is minimised. To further reduce vibrations, rubber dampers are added between the frame and the sensing board.
- The Fairphone 2 (here not disassembled) is placed below the quadcopter, while the battery is positioned in the centre of the drone. As a result, the gravity point is maintain as low as possible maximising the stability of the drone.
- The four (yellow) ESCs are placed on the arms of the quadcopter (here on top to illustrate their presence). They are connecting the motors to the battery through the power distribution board and controlled by the Arduino. This power distribution board is made of a printed circuit board and attached between the top plate and the battery.
- The Fairphone 2 is connected to the Arduino with a USB OTG cable and this Arduino is also connected to the IMU sensors and the barometer board.

The final prototype is shown in the following Figures 4.10:



(a) Side view. Respectively to the CAD, the sensing electronics is disposed on top, the battery on it centre and the Fairphone2 is attached underneath.

(b) Top view, where the sensing electronics as well as the connection cables are visible

(c) Bottom view, exposing the ESCs as well as the Fairphone 2 (in all its part) powered with postmarketOS.

Figure 4.10: Designed quadcopter prototype, pictures from different point of view. [2]

[2]Here the red and black electric cable going out of the drone (visible in these pictures) is not part of the original design but will be used for the test procedure, more details in Section 7.2

Chapter 5

States estimation of the drone

In this chapter, we outline the necessary steps to estimate the orientation (states) of the drone which is crucial for autonomous control. An overview of the different sensors onboard is also made, leading to their implementation.

Now that the quadcopter is built (see Chapter 4), the next step consists of the implementation of the control dynamics using control loops. As represented in Figure 5.1, the implementation of a control loop (introduced in Figure 2.6) includes different key elements such as remote input command, sensors and the controller. This chapter will focus on the state's estimation of the quadcopter using the onboard sensors as well as the microcontroller.

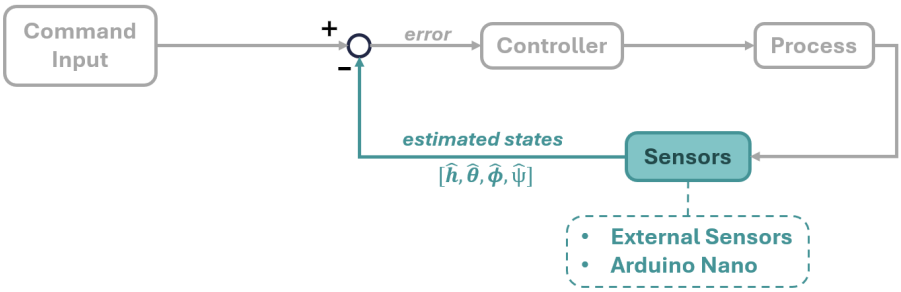


Figure 5.1: Control loop diagram of the implemented design, highlighting the states estimation process processed by the external sensors and the Arduino nano onboard.

In our design, four quadcopter's states are autonomously controlled on the pitch θ, roll ϕ, yaw ψ and height h. These states are responsible for the orientation and the altitude of the drone. They are measured using the selected onboard sensors of the V2 prototype (see Section 4.3), specifically a barometer with an IMU including an accelerometer, a gyroscope, and a magnetometer. These sensors will be used as illustrated in Figure 5.2 where they could be combined to improve the state's estimation as developed later in Section 5.1.3.

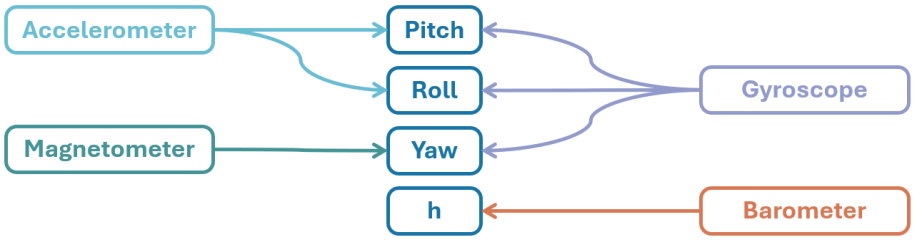


Figure 5.2: States estimation using the four onboard sensors.

The following sections will describe how the states of the quadcopter can be estimated using these sensors. It involves exploring the characteristics of each sensor as well as the application of additional filtering algorithms to enhance the accuracy of the state estimations.

5.1 Pitch and roll estimation

The pitch θ and roll ϕ of the quadcopter can be estimated with two different sensors, respectively the accelerometer and the gyroscope.

5.1.1 Accelerometer

The accelerometer is a device measuring the acceleration in the body frame of a vehicle $[m/s^2]$. The measured acceleration estimated is the total acceleration of the aircraft minus the gravity. Mathematically [53], this relationship is defined as :

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \underbrace{\frac{d\boldsymbol{v}}{dt}}_{\text{Linear acceleration}} + \underbrace{\boldsymbol{\omega_b} \times \boldsymbol{v}}_{\text{Rotational acceleration}} - \underbrace{R_i^b \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}}_{\text{Gravitational acceleration}} + \boldsymbol{\beta}(t) + \boldsymbol{\eta}(t) \quad (5.1)$$

Where :

- $\boldsymbol{v} = (u, v, w)^T$ represent the linear velocity of the vehicle $[m/s]$.
- $\boldsymbol{w_b} = (p, q, r)^T$ denotes the angular body rate $[rad/s]$.
- $R_i^b = R(\phi)R(\theta)R(\psi)$ is the rotation matrix from the inertial frame to the body frame determined by the Euler angles ϕ, θ, ψ.
- $\boldsymbol{\beta}(t)$ are sensors biases.
- $\boldsymbol{\eta}(t)$ represents additional noise measured by the sensor which can either be intrinsic noise or environmental noise.

Assuming no acceleration of the aircraft (only the gravitational remains) and resolving the Equation (5.1) along each body axis, we get :

$$\begin{aligned} a_x &= g\sin\theta + b_x + \eta_x \\ a_y &= g\cos\theta\sin\phi + b_y + \eta_y \\ a_z &= g\cos\theta\cos\phi + b_z + \eta_z \end{aligned} \tag{5.2}$$

From these equations we are now able to estimate the pitch (θ) and roll (ϕ). However, the yaw angle (ψ) cannot yet be estimated from the accelerometer data alone, as the accelerometer is primarily sensitive to gravitational forces and does not provide direct information about rotation around the vertical axis.
If the bias b_* are known, for example with a pre-flight calibration, then the roll and pitch angles can be approximated as follows :

$$\begin{aligned} \text{Estimated roll: } \hat{\phi}_{acc}(t) &= \tan^{-1}\left(\frac{a_y - b_y - \eta_y}{a_z - b_z - \eta_z}\right) \\ &= \phi(t) + \eta_\phi \end{aligned} \tag{5.3}$$

$$\begin{aligned} \text{Estimated pitch: } \hat{\theta}_{acc}(t) &= \sin^{-1}\left(\frac{a_x - b_x - \eta_x}{g}\right) \\ &= \theta(t) + \eta_\theta \end{aligned} \tag{5.4}$$

The implemented pre-flight calibration sequence consists estimating b_* by collecting the measured data from the sensor over certain time at rest (30 seconds in our application). The measured values are then averaged and subtracted from real time data.

To validate these equations, a sequence of different position is applied to the onboard accelerometer. The sequence can be described in terms of imposed angles where the drone is tilted forward by 45° for 10 seconds, followed by a +90° yaw rotation, and finally, another 45° roll inclination for 10 seconds. Figure 5.3 illustrates how the imposed 45° angles are carried out.

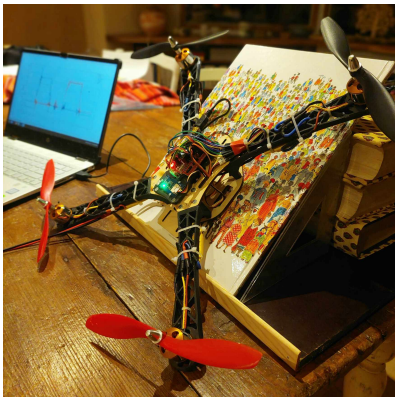


Figure 5.3: Picture of the experiment setup in order to applied 45° angle to the onboard sensor.

For the accelerometer, the results are displayed in Figure 5.4. In this experiment it can be observed that the onboard accelerometer correctly estimates the imposed angle of + 45° in pitch and roll, with a maximum error of 1 degree. Furthermore, some undesired oscillations can also be noticed. These oscillations results from a potential noisy external acceleration measured by the accelerometer due to the manipulation of the sensor, precisely at t = [20; 30; 35; 40; 50] seconds, exactly when the sensor changes position. We assume these oscillation as high-frequency environmental noise and can therefore be reduced with a low pass filter, as confirmed in Section 5.1.3.

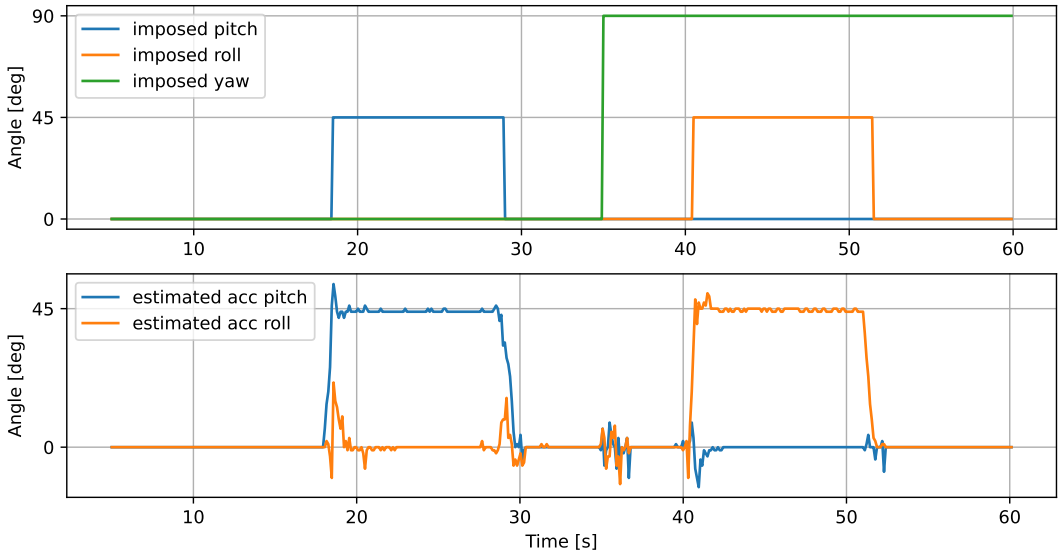


Figure 5.4: Comparison between the imposed angles (above) and the measured angles (below) using the onboard accelerometer, highlighting the good angles estimation by the accelerometer.

These results confirm that Equation (5.3) and Equation (5.4) accurately estimate the pitch and roll angle as long as the sensor is not influenced by external accelerations other than gravity. In the presence of such external accelerations, applying a low-pass filter is necessary to reduce high-frequency noise and improve the precision of the angle estimations.

5.1.2 Gyroscope

A gyroscope or also known as rate gyro, can detect changes in rotation angle per unit of time [rad/s] around its three axes. The rate gyroscope measurements of angular body rates p, q, and r can be modelled as [53]:

$$gyro_x = p + b_p + \eta_p \tag{5.5}$$

$$gyro_y = q + b_q + \eta_q \tag{5.6}$$

$$gyro_z = r + b_r + \eta_r \tag{5.7}$$

With b_* bias term and η an additional noise measured by the sensor which can either be intrinsic noise or environmental noise. By integrating the angular rate data, the angle covered over the time can be estimated : [1]

$$\hat{\phi}_{gyro}(t) = \int_{-\infty}^{t} gyro_x(\tau)\, d\tau = \phi(t) + \int_{-\infty}^{t} (b_p(\tau) + \eta_p(\tau))\, d\tau \tag{5.8}$$

$$= \phi(t) + \beta_\phi(t) \tag{5.9}$$

$$\hat{\theta}_{gyro}(t) = \int_{-\infty}^{t} gyro_y(\tau)\, d\tau = \phi(t) + \int_{-\infty}^{t} (b_q(\tau) + \eta_q(\tau)), d\tau \tag{5.10}$$

$$= \theta(t) + \beta_\theta(t) \tag{5.11}$$

$$\hat{\psi}_{gyro}(t) = \int_{-\infty}^{t} gyro_z(\tau)\, d\tau = \phi(t) + \int_{-\infty}^{t} (b_r(\tau) + \eta_r(\tau)), d\tau \tag{5.12}$$

$$= \psi(t) + \beta_\psi(t) \tag{5.13}$$

By integrating the bias b_* and the noise η an additional error signal $\beta_*(t)$ is introduced. Over time, this leads to a drift in the estimated values. If the bias terms are known, for example through pre-flight calibration the drift can be minimised. As $\beta_*(t)$ are assumed signal with low frequency content, a high pass filtered version of $[\hat{\phi}_{gyro}(t), \hat{\theta}_{gyro}(t), \hat{\psi}_{gyro}(t)]$ will respectively represent $[\phi(t), \theta(t), \psi(t)]$.

Furthermore, a similar experiment to the previous one (Figure 5.4) was carried out to check the validity of the estimates. From the results shown in Figure 5.5, it appears that the estimated angles do not accurately follow the imposed sequence. This can be explained by the integration process, where noise generated by the sensor's manipulation is integrated, creating offsets. While gyroscopes provide accurate estimations over short duration, they exhibit limitations over longer periods. Specifically, there is noticeable drift in the estimated yaw angle. This drift is attributable to integration errors caused by residual bias. Therefore, implementing an additional high-pass filter could effectively mitigate these errors and improve the estimation. This assumption is confirmed in Section 5.1.3.

[1] In the development we assumed $p, q, r = \dot{\phi}, \dot{\theta}, \dot{\psi}$

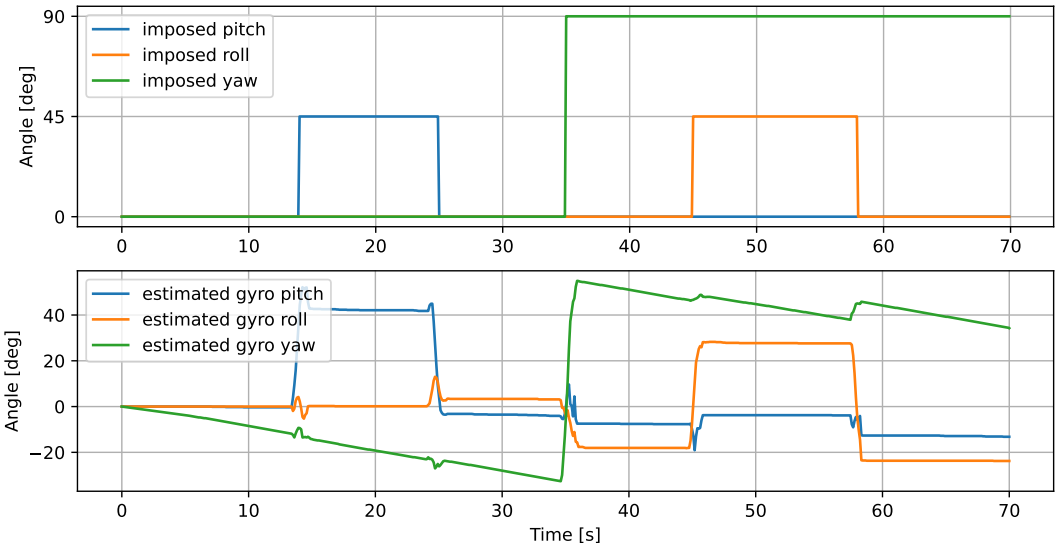


Figure 5.5: Comparison between the imposed angles (above) and the measured angles (below) using the onboard gyroscope, highlighting the drift obtained between the estimated angle and the imposed angle.

5.1.3 Complementary Filter

As described previously, the pitch and roll angles can be estimated using both the accelerometer and the gyroscope. Each sensor has its strengths and weaknesses: the accelerometer provides reliable long-term estimates but is susceptible to noise and external accelerations, while the gyroscope offers robust short-term estimates but is prone to drift over time due to bias [54].

To improve the overall estimation, sensors fusion algorithms may be applied. These algorithms are based on the idea that the errors from one sensor will be compensated by the other sensor, and vice versa. In our design, a complementary filter is implemented. This filter combines the accelerometer's low-pass filtered estimates with the gyroscope's high-pass filtered estimates. By combining these two estimations, the accuracy and reliability of the pitch and roll measurements can be improved [55]. Figure 5.6 and following developments illustrates the implementation for the roll angle ϕ but the same approach is applicable for the pitch estimation.

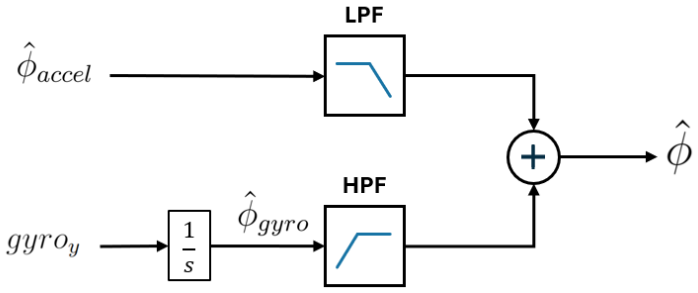


Figure 5.6: Complementary filter block diagram where the ϕ angle is estimated by combining a high pass of the accelerometer estimates and a low pass of the gyro estimates after integration ($[\frac{1}{s}]$ block).

The general equation for the complementary filter can be discretised and expressed as:

$$\hat{\phi}_{n+1} = \phi_{acc,n} \cdot \alpha + (1 - \alpha) \cdot \left[\hat{\phi}_n + T \cdot gyro_{y,n}\right] \tag{5.14}$$

In this equation

- $\hat{\phi}_{n+1}$ is the estimated angle [rad] at the next time step.
- $\hat{\phi}_n$ is the estimated angle [rad] at current time step.
- $\phi_{acc,n}$ represents pitch [rad] estimated by the accelerometer at the current time step.
- $gyro_{y,n}$ is the angular velocity [rad/s] measured by the gyroscope data at the current time step.
- α is the filter coefficient.
- T is the time interval between measurements [s].
- The term $\left[\hat{\phi}_n + T \cdot gyro_{y,n}\right]$ integrates the gyroscope's angular velocity to get the angle.

The factor α determines how much weight is given to the gyroscope data versus the accelerometer data. If α is close to 1, more weight is given to the accelerometer (lower frequency component), if α is close to 0, more weight is given to the gyroscope (higher frequency component). Its tuning is made experimentally but for such application a typical value will be around 0.02 [55, 56].

Figure 5.7 illustrates the applied complementary filter for pitch estimation with an α value of 0.01. This filter is running on the microcontroller onboard (arduino Nano). These results are obtained from the pre-calibrated onboard sensors of the quadcopter and are recorded while the motors are inactive. During the experiment, the drone is tilted forward and backward randomly. As shown, the final estimation

achieved with the complementary filter demonstrates significant improvement in angle estimation. The resulting angle is smoother and more precise than the individual estimation from the gyroscope or accelerometer alone. With this filter, the accelerometer eliminates drift created by the gyroscope, while the gyroscope effectively reduces undesired measured acceleration.

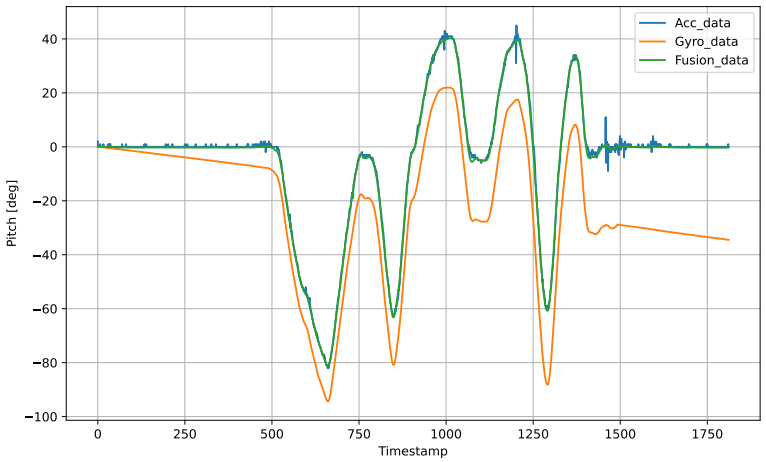
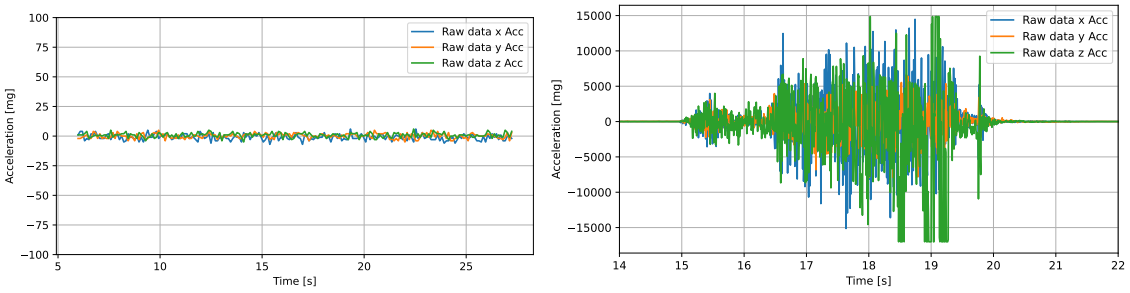


Figure 5.7: Quadcopter's pitch estimation comparison between the accelerometer, gyro and complementary filter estimation at rest, with pre-callibrated sensors and a non-determined moving sequence, showing that the angle estimation is improved when the sensors data are combined.

However, as shown in the experiment Figure 5.8, which compares the measured acceleration when the motors (coupled with propellers) are at rest and when they are running, the accelerometer measures significant noise when the motors are rotating. The measured acceleration increases with the power of the motors, reaching up to 16g (gravity of Earth) and causing the accelerometer to saturate. Due to this noise (probably due to the vibration or the electromagnetic interference's created by the motors), the implemented complementary filter needs to rely more on the gyroscope than the accelerometer, resulting in a low α.
As future work, conducting a noise analysis would allow for a more precise determination of the noise sources being measured, thereby improving the quality of the results.



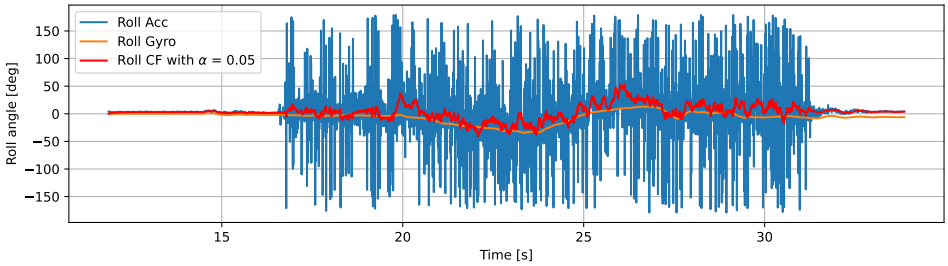
(a) Accelerometer raw measurement data when motors at rest.

(b) Accelerometer raw measurement data when motors with four motors powered, with increasing throttle.

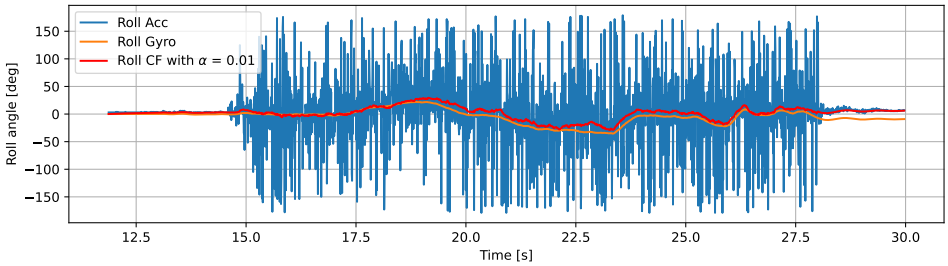
Figure 5.8: Raw accelerometer data compared between motors at rest and powered showing the saturation of the sensor when the motors are rotating.

To determine the optimal α value, a series of tests are conducted with different values. The values tested range from 0.05 to 0.001. For these tests, the four motors are powered, in order to simulate flight condition. The measured data, for three different values of α, are displayed in Figure 5.9. Based on qualitative appreciation of the results, an α value of 0.01 is selected.

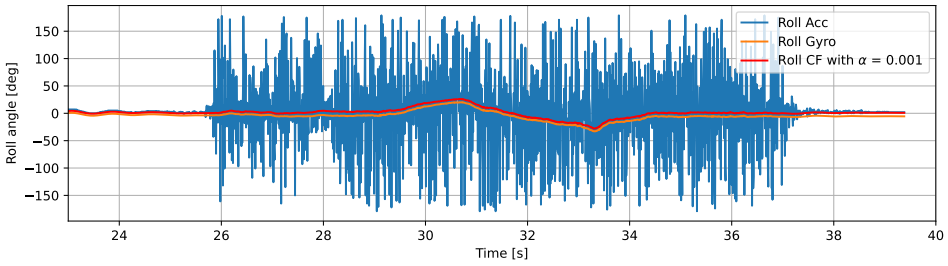
The complementary filter is motivated by its high simplicity of implementation. However, more complex fusing algorithms, such as the Kalman Filter, could further improve estimation accuracy. The Kalman Filter is a recursive algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies to produces optimal estimates that tend to be closer to the true values of the measurements. With this filter the α value is updated and optimised at each iteration. However, using such a filter is not only more complex to implement but also more computationally demanding [57]. Therefore it hasn't been done for this project but could be a good improvement for the future.



(a) CF with $\alpha = 0.05$. In this configuration the CP estimation is too noisy for an effective control (too much weight given to the accelerometer).



(b) CF with $\alpha = 0.01$. Here as α decreased, the CP estimation is closer to the gyro estimation.



(c) CF with $\alpha = 0.001$. Here, the CP estimation is only following the gyro estimation, loosing potential useful information of the accelerometer.

Figure 5.9: Quadcopter's roll estimation comparison between the accelerometer, gyroscope and complementary filter estimation with different α values. The best results are obtained with an α of 0.01.

5.2 Yaw estimation

The yaw estimation is used to control the rotation of the quadcopter around its z axis. As described before, the gyroscope measures angle around the 3 axes. Therefore, by integrating data over time, an estimation of the yaw can be made. However, alone, this value tends to drift due to the integration of noise (see Figure 5.5). Consequently, similar to the pitch and roll, the combination of the gyro with another sensor can improve the state's estimation. Another available sensor capable of measuring a yaw angle is the magnetometer.

5.2.1 Magnetometer

The magnetometer is a sensor used to measure the strength and direction of the local magnetic field surrounding a system [µTesla]. Since the earth is surrounded by a magnetic field the magnetometer can serve as digital compass to provide indication of heading relative to the magnetic North [ψ_m]. Magnetometers are commonly modelled using Equation (5.15), where $\boldsymbol{m}$ contains the magnetic readings as measured by the sensor [58].

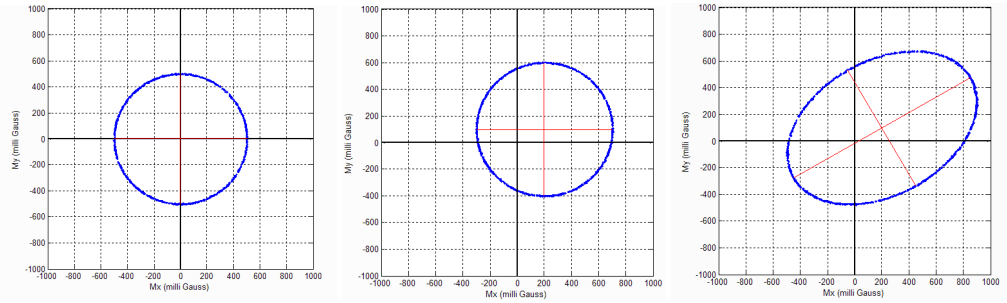
$$\boldsymbol{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = S_I^{-1}(m_E + m_{e(t)}) + b_{HI} + m_{i(t)} + \eta(t) \tag{5.15}$$

In an ideal magnetic environment, the measurement vector would only represent the Earth's magnetic field (m_E). However, in practical applications, the measured magnetic field typically includes contributions from both Earth's magnetic field and magnetic fields generated by nearby elements, commonly referred to as magnetic disturbances. These disturbances include both external objects in the surrounding environment ($m_e(t)$) and internal objects fixed relative to the sensor ($m_i(t)$). Additionally, hard iron distortions (b_{HI}) and soft iron distortions (S_I) can bias and distort Earth's magnetic field. Moreover, η represents additional magnetic noise (intrinsic or environmental) that are not represented in the previous terms.

Leaving aside the temporal interference ($m_{e(t)}$ and $m_{i(t)}$) which can poorly be compensated we will focus more on the hard and soft distortion. These interference's are deterministic and can significantly be reduced.

- **Hard Iron Distortions (b_{HI})** : These are caused by permanent magnetic fields from objects that are rigidly fixed to the magnetometer, such as electronics and structural components. They create a constant offset in the magnetic field measurements.
- **Soft Iron Distortions (S_I)** : These result from the presence of ferromagnetic materials near the magnetometer. They distort the magnetic field by altering its direction and magnitude depending on the sensor's orientation relative to these materials.This type of distortion is commonly caused by metals such as nickel and iron (like batteries).

These distortions can be visualised by fully rotating the magnetometer and plotting the measurements data as y axe vs x axe. In the absence of distortion, a perfectly centred circle will be observed, as depicted in Figure 5.10. Hard distortion, will cause an offset in the measurements, decentralising the circle. While soft-iron distortion will distort or stretch the magnetic field, making the circle of ideal measurements ellipsoid.
To note that these figures represent distortions in the 2D plane, but in reality, these distortions occur in 3D space hence in a sphere shape.



(a) No distortion (b) Hard iron distortion (c) Hard and soft distortion

Figure 5.10: 2D Visualisation of the soft and hard Magnetic distortions, theoretical examples [58].

To correct these distortion a pre-calibration is needed. This calibration needs to be performed in the same condition where the sensor is performing, here in the centre of the drone. By rearranging Equation (5.15) (with $m_{e(t)}$ and $m_{i(t)}$ removed), a compensation model (m_c) can be established :

$$m_c = \begin{pmatrix} m_{cx} \\ m_{cy} \\ m_{cz} \end{pmatrix} = S_I(\boldsymbol{m} - b_{HI}) + \eta \tag{5.16}$$

Heading

The heading (ψ_h) is the direction an object faces relative to true North, measured in degrees (0° for North, 90° for East, etc), see Figure 5.11. The calculated heading from the magnetometer is magnetic North, which may differ from true North due to magnetic declination (the angle difference between magnetic North and true North). To obtain the true heading, we need to add to the local magnetic declination δ to the magnetic heading ψ_m [2].

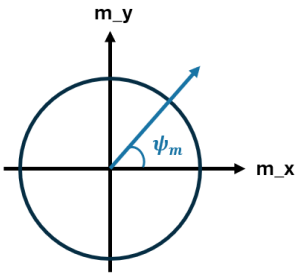
$$\psi_h = \psi_m + \delta \tag{5.17}$$

Using a magnetometer, the (magnetic) heading is calculated using the horizontal components (m_x, m_y) with the following formula :

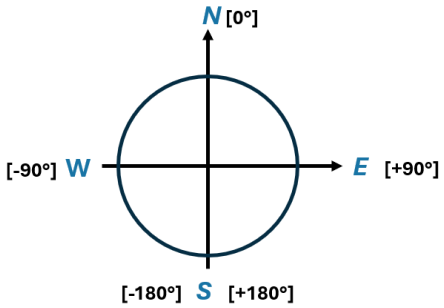
$$\psi_m = atan2\left(\frac{m_y}{m_x}\right) + \eta \tag{5.18}$$

Where the *atan2()* function returns the $atan()$ of m_x/m_y in the range of $[-\pi; \pi]$. This results can after be converted to degrees.

[2] In Bruxelles, the magnetic heading δ is +2° 23' (E)



(a) Magnetic heading calculation using the measurements of the magnetometer in the x and y axes.



(b) Mapping of the cardinal direction on the heading calculation resulting from the atan2() function.

Figure 5.11: Representation of the magnetometer's measurements which can be used to estimated the angle relative to the magnetic North (magnetic heading ψ_m).

Finally, in real-world applications, the magnetometer is often not perfectly horizontal. Therefore, tilt compensation is required. The magnetometer measurements are compensated using the estimation of the Euler angles derived from other sensors and rotation matrices $\boldsymbol{R}()$.

$$\boldsymbol{m}^I = \begin{pmatrix} m_x^I \\ m_y^I \\ m_z^I \end{pmatrix} = R(\theta)R(\phi)\boldsymbol{m} \tag{5.19}$$

With $\boldsymbol{m}^I$ is a vector of the projected measurements on the horizontal plane. Therefore the final heading is expressed as:

$$\psi_h = atan2\left(\frac{m_y^I}{m_x^I}\right) + \delta + \eta \tag{5.20}$$

Implementation

In our application of controlling a quadcopter, the heading measurement Equation (5.20) is used to determine the yaw angle of the aircraft. During the calibration procedure, an initial heading angle is measured and stored. Subsequently, the current heading is compared to this initial value to determine the yaw angle. This allows us to start with a heading $\psi_h = 0$ and evolving during flight time.

The calibration procedure involves collecting data from the onboard magnetometer while the drone is maintain still and then tilted along its x and y axes (with motors at rest). From this data, the composition model (m_c) introduced in Equation (5.16) is determined, along with the initial heading.

Finally, using the gyroscope and the magnetometer, the yaw estimation can also benefit from the complementary filter as shown in Figure 5.12. Again, these sensors corresponds to the IMU onboard and the filter is processed by the Arduino Nano.

In this experiment, the drone (with motors at rest) is moved around the z axe randomly. As observed, the magnetometer estimation presents some high frequency noise, while the gyroscope tend to drift over time. Combining these two estimations with the complementary filter effectively mitigates these issues, resulting in a more accurate yaw estimation. The value of α in the following experience is 0.02, giving more weight to the gyroscope measurements. This value is standard and has not been optimised.

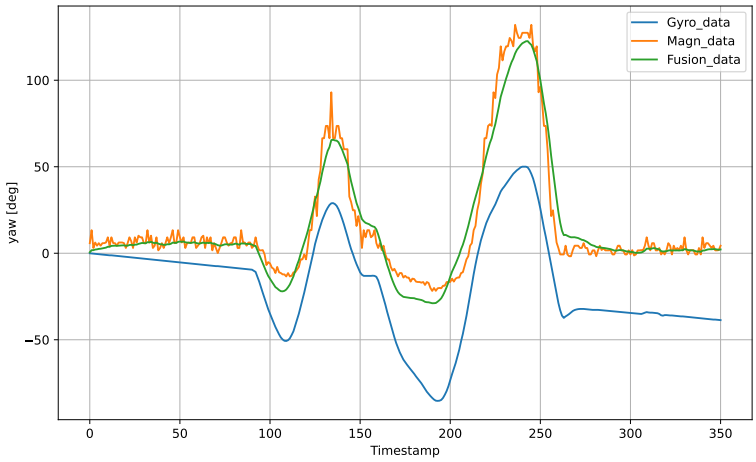


Figure 5.12: Quadcopter's yaw estimation comparison between the magnetometer, gyro and complementary filter estimation at rest, with pre-calibrated sensors, non determined imposed rotation. The angle estimation is improved when the sensors data are combined.

However, here the experiment is realised using only the microcontroller and the external sensors. In real condition, with the spinning motors battery and Fairphone 2 onboard, the yaw magnetometer estimation might suffer from external magnetic interference. Currently, no tests of this kind have been conducted, making it difficult to assess the degree of disturbance. This remains an area for future development.

5.3 Altitude estimation

For the actual version of the quadcopter, only the altitude is part of the quadcopter position control. This states can be measured using the barometer included in the prototype's design (see Section 4.3.1).

5.3.1 Barometer

Since atmospheric pressure changes with altitude, temperature, and humidity, it can be used to estimate altitude relative to a reference point. By keeping temperature

and humidity constant, the altitude can be approximated using the following formula [59]:

$$h_2 = h_1 + \frac{T}{L}\left[1 - \left(\frac{P_2}{P_1}\right)^{\frac{R \cdot L}{g_0 \cdot M}}\right] \tag{5.21}$$

where:

M is the molar mass of air (0.0289644 [kg/mol]).

R is the universal gas constant (8.31432 [N · m/(mol · K)]).

g_0 is the acceleration due to gravity (9.80665 [m/s^2]).

L is the temperature lapse rate (-0.0065 [K/m]).

P_1 and P_2 are the pressures [Pa] at altitudes h_1 and h_2 respectively.

T the constant measured temperature [K].

The barometer sensor used is characterised by a precision of 0.002 hPa (or ±0.02 m). To verify this precision value, an experiment is conducted:
During the calibration phase, an initial value h_1 is set to zero meters, and a certain altitude h_2 (equals to the ground altitude) is estimated using the previous formula (see Equation (5.21)). Subsequently, by replacing h_1 with the estimated h_2 from the calibration, the varying altitude can be measured. This setup theoretically establishes a starting point at zero meters. In the experiment (see Figure 5.13), the calibrated sensor is moved (by hand) vertically at t ≈ 570 [s] and then remains stationary for the remainder of the test. The motors of the drone are at rest and the algorithm is processed by the Arduino Nano.

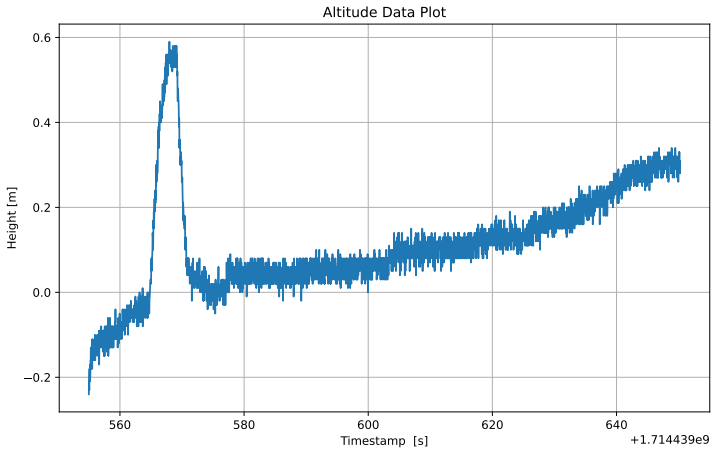


Figure 5.13: Estimated altitude test using the pre-calibrated onboard barometer. In this experiment the sensor is moved up and down at t = 570 [s] and then remains still for the rest of the experiment.

As shown in the Figure 5.13 while the sensor is able to measure the ≈ 0.6 meter variation in vertical distance, the measurements tend to drift over time by 30 [cm] (from t = 580s) when the sensor is maintained still. This drift makes the sensor quite limiting to control the drone's height, especially for landing and takeoff where an accurate control is required. This drift tends to result from measured noise of the sensor, limiting its precision.

As a result, this process is useful for detecting large altitude changes, but in our application where high precision is required, additional measurement systems or more advanced algorithmic estimations are necessary. Furthermore, the test was conducted at rest, suggesting that the observed drift could potentially be greater in actual flight conditions. These findings align with [12] where they tried to estimate the altitude with only a barometer and conclude that extra processor sensors are necessary to enhance the estimation.

5.4 Conclusion

In summary, this chapter covers the states estimation of the control loop. Based on this analysis, it can be concluded that, aside from altitude estimation, which requires further work, the three Euler angles (pitch, roll and yaw) can be accurately estimated using the available onboard sensors. As resumed in Figure 5.14, the pitch and roll are estimated using a complementary filter combining the gyro integration and the acceleration measurements. For the yaw, the angle is estimated combining the gyroscope integration and the magnetometer using also a complementary filter. These 3 sensors correspond to the 9 DOF IMU of the prototype design (V2) and the complementary filter are executed by the microcontroller.

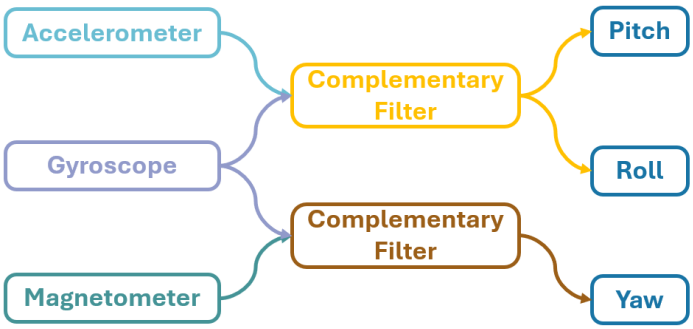


Figure 5.14: Attitude states estimation resulting from onboard sensors and digital processing algorithms.

Chapter 6

Remote control and wireless connection

In this section, we discuss how discarded smartphones are repurposed to control the quadcopter. Additionally, we explain how the wireless communication is established between the drone and the remote control.

This chapter focuses on the command input, a key element of the control loop illustrated in Figure 6.1. In previous Chapter 5, where the state's estimation was developed, only the pitch and roll were tested under flight conditions. Since the altitude and yaw estimations require additional development and testing, the implemented autonomous control will be limited to the pitch $[\theta]$, roll $[\phi]$. Additionally, without altitude control, an additional variable (thrust $[T]$), is necessary to manage the power of the quadcopter's motors, enabling it to change speed and height.

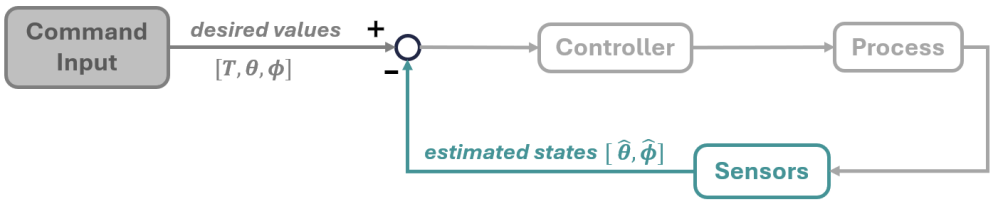


Figure 6.1: Control loop diagram of the quadcopter control highlighting the command input process and illustrating the states control θ and ϕ.

Like many UAV application, the quadcopter prototype is controlled from distance. In the implemented (V2) design (see Section 4.3), the remote control is made reusing a discarded smartphone, Samsung A3 (SA3), which communicates on Wi-Fi with a second discarded smartphone Fairphone 2 (FP2) onboard. The FP2 serves as interface between the command input from the SA3 and the microcontroller. The microcontroller, in our case an Arduino Nano, estimates the quadcopter's states (pitch and roll) from the measurements of the external sensors (accelerometer and gyro). Finally, based on the Samsung A3 inputs and the estimated states, the Arduino sends necessary outputs to control the motors as desired.

Figure 6.2 summarises the components of the designed prototype, their interaction and the type of signal used.

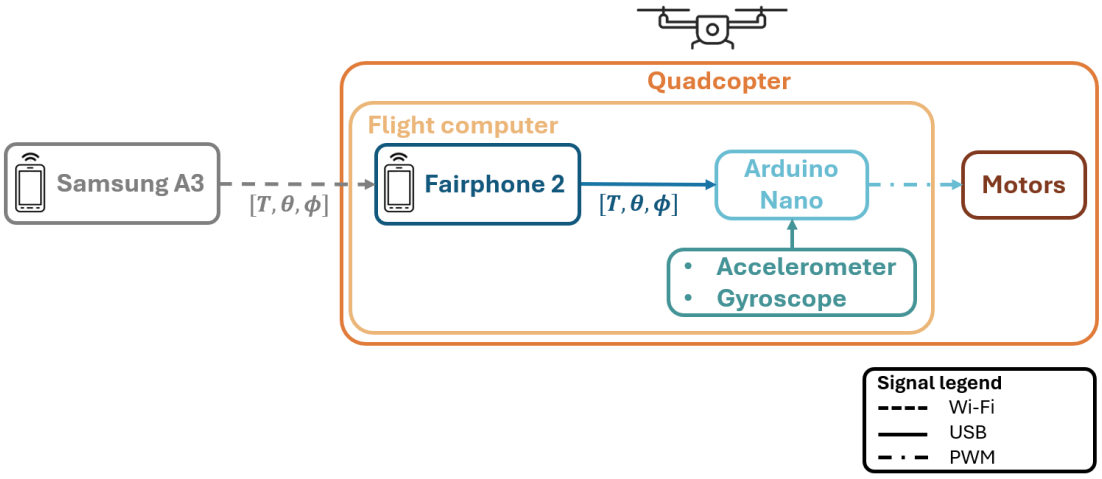


Figure 6.2: Adapted block diagram of the prototype where the inputs command are sent to the flight computer over Wi-Fi which after processing, adjust PWM outputs to control the motors.

In this design, the pitch and roll are determined using the sensors of the remote smartphone (SA3). These angles serve as input control for the drone and are represented in Figure 6.3. For example, if the phone is tilted on its y axe (resulting in a positive pitch command), the drone replicates this action by tilting forward as well.
The thrust variable (T) command will determined using the volume button (+ and -) of the phone. The power button will also be used as Start/Stop command.

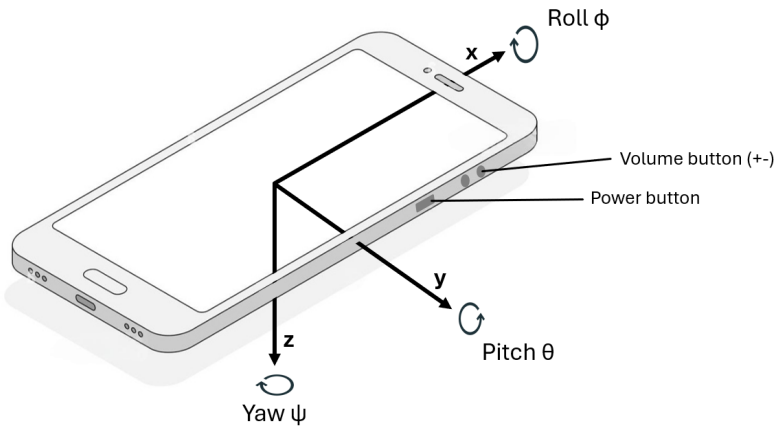


Figure 6.3: Smartphone's Euler angles and key buttons representation. If the phone is inclined on its x or y axe a roll and pitch are measured. The phone can also received user inputs via the key buttons.

The remote smartphone (SA3) is then exploited for two reasons :

(1) To control the drone in term of pitch, roll and thrust using its internal sensors and physic buttons, referenced as GPIO (General Purpose Input/Output) keys.

(2) To connect in Wi-Fi with the FP2 onboard of the quadcopter.

Additionally, at the early stage of the development, the Samsung A6+ (the third discarded smartphone exploited, see Figure 4.3) was first used as remote controller but due to restrictive issues the SA3 was used instead. The selection of this smartphone is the conclusion of several experiments as detailed later in Section 5.1.

In our design, all the discarded smartphones (SA6+, SA3, and FP2) are exploited using postmarketOS as phone Operating System. As developed in Section 3.4, this software allows us to compile and run code as well as access internal sensors and communication systems. However, since pmOS is not specifically designed for this type of application, particular try-error process is realised to explore the sensors and to acquire the desired data. Although this chapter is technical and experimental, it demonstrates new achievement: using the sensors of discarded smartphones with postmarketOS. Therefore it was important for us to develop in details the process.

Finally this chapter which aims to implement the command input component, is developed as follow:

1. Access the internal sensors and buttons of the discarded smartphone (Section 6.1).
2. Utilise the internal sensors and physical buttons to estimate pitch, roll, and thrust commands (Section 5.1).
3. Establish a Wi-Fi interface between the remote SA3 and the onboard FP2 to transfer the commands to the controller (Section 6.4).
4. Integrate all the steps to produce an effective control system (Section 6.5).

6.1 Access to the internal sensors

With postmarketOS installed on the phones, the architecture of the interface is similar to a Linux machine (since pmOS includes Alpine Linux). By connecting to the phone via the Secure Shell (SSH) protocol, the sensor measurements can be accessed through individual folders. These folders describe the internal sensors of the phone. For the SA6+, the accelerometer and gyroscope are available, while for the SA3 it is the the accelerometer and magnetometer. The listed directory below resumed the key elements of one of the folder (here the accelerometer of the SA6+).

```
samsung-a6plt:/sys/bus/iio/devices/iio:device2 ls
in_accel_scale
in_accel_scale_available
in_accel_x_raw
in_accel_y_raw
in_accel_z_raw
sampling_frequency
sampling_frequency_available
```

Listing 6.1: Terminal output listing the contents of the system directory.

Regarding the directory (Listing 6.1) :

- The files `[in_accel_*_raw]` contain the raw acceleration measurements from the sensor. These files can be directly accessed to retrieve the sensor's raw measurements.
- `in_accel_scale` typically represents the default scaling factor for the accelerometer readings, converting raw data into actual acceleration expressed in g (gravity of Earth). This scaling factor is listed among several values available in `in_accel_scale_available`.
- `sampling_frequency` determines how often the sensor data is updated and made available for processing. This frequency impacts the real-time capability of motion tracking. The desired sampling frequency can be adjusted and selected from the options listed in the `sampling_frequency_available` file. To simplify further development we will renamed them as **phone sampling frequency**.

By comparing these elements with the corresponding datasheet (which resumes the specifications of the sensors), it can be notice that certain values in the `in_accel_scale_available` and `sampling_frequency_available` differ or are missing from the specifications outlined in the datasheet. These defaults seem to be due to the implementation of the drivers, the specific program that interacts with the sensors.

In our application, we will focus on the *phone sampling frequency*, which will directly influence our control system. Consequently, a simple test code is developed to investigate the actual data acquisition speeds of the sensor. This program simply measured the elapsed time between two sensor's reading.
With this program we want to evaluate the sampling frequency of the sensors in a python script environment. Therefore, the *measured frequency* obtained do not correspond to the *phone frequency*. However, it will still provide valuable insights into the sensor's coding and performance characteristics. The pseudo code of the script is detailed as follows:

```
Set start_time to current time
while True :
    Read sensor's raw measurement
    dt = current time - start_time
    Measured frequency = 1/dt
```

Listing 6.2:]Pseudocode for measuring sensor data acquisition frequency [1/s]

This test was carried out with the different phone sampling frequency for the different sensors of the Samsung A3 and Samsung A6+. The Table 6.1 displays the *phone sampling frequency* along the *measured frequency* obtain via the the script detailed previously (see Listing 6.2). This table is then plot as shown in Figure 6.4.

Sensors	**Phone freq [Hz]**	**Measured freq [Hz]**
Accelerometer (SA3)	16, 32, 63, 125, 250, 500, 1000, 2000	930, 1084, 1077, 1082, 1084, 1000, 1020, 1060
Magnetometer (SA3)	2, 6, 8, 10, 15, 20, 25, 30	490, 534, 512, 466, 504, 514, 489, 557
Gyroscope (SA6+)	12.5, 52, 104, 208, 416	3.086, 12.5, 25, 43.47, 77
Accelerometer (SA6+)	12.5, 52, 104, 208, 416	3.086, 12.5, 25, 43.47, 77

Table 6.1: Phone sampling frequency vs measured frequency from test code of internal sensors of the discarded smartphones under pmOS compared. It shows that the measured frequency differs from the phone frequency, and these differences vary depending on the phone.

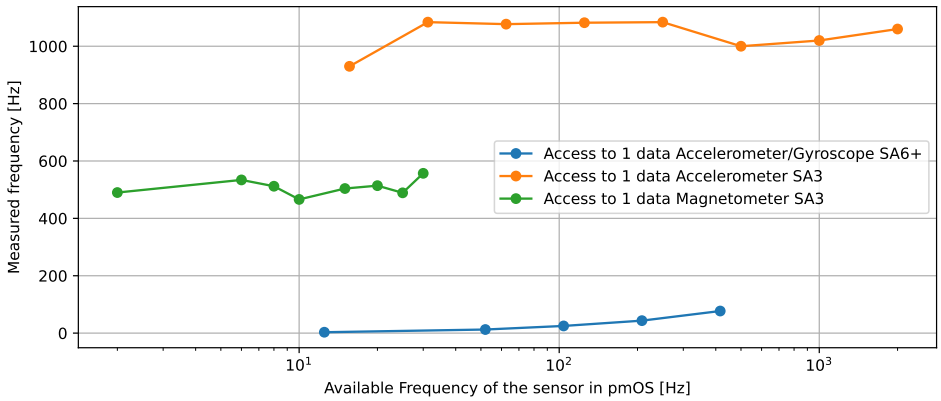


Figure 6.4: Lin/log plot of the acquisition data measured frequency from the tested program related to the available frequency, based on the Table 6.1. For the SA6+, the measured freq increases with the phone freq. While for SA3, the measured freq tends to remain constant, regarding the phone freq.

As observed, the results highly differs regarding the discarded phone exploited. For the Samsung A6+, as expected, a higher available (phone) frequency involves a higher measured frequency. While for the Samsung A3, the measured frequency tends to be always the same regarding the available frequency. These differences indicates a variant in the way drivers are coded. Moreover, we can also see that

the Samsung A3's sensors have a much higher acquisition frequency than those on the Samsung A6+. The consistently high acquisition frequency observed for the SA3 might be attributed to the configuration of its sensors, which may be set to non-filtered mode, allowing them to operate at their maximum sampling frequency (2000[Hz]).

Another test demonstrated that the time required to run the program increased with the number of sensor readings, at least for the Samsung SA6 (see Figure 6.5). This test consists of running the same program as before but here, instead of reading one value, we are reading three. Therefore, if the program reads only the acc_x value at a measured frequency of 75 Hz, then when acc_x, acc_y, and acc_z are read, the final measured reading frequency will be $\approx 75/3 = 25$ Hz. This difference is even more noticeable at higher frequency.
Here, this issue may be explained by how the measured values are acceeded, resulting in a sequential reading of the measurements and leading in a addition of time at each on each request. This hypothesis could not be verified and requires a more in-depth analysis of the driver implementation.

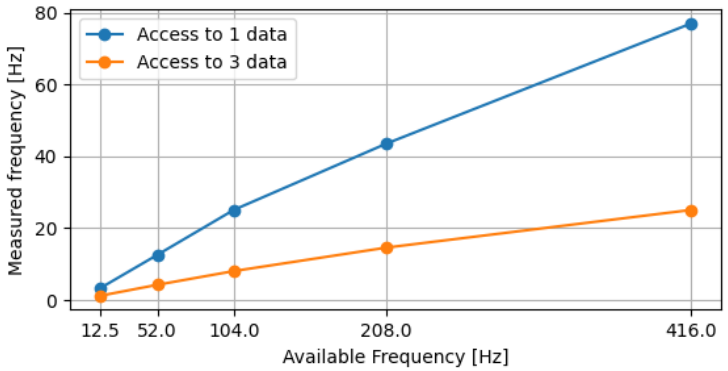


Figure 6.5: Plot of the experienced frequency to read three values of the accelerometer of the Samsung A6+ (acc_x, acc_y, and acc_z) compared with the frequency measured to access on value of the accelerometer (acc_x). The measured frequency increased with the number of data requests.

At this point, it emerges that for two discarded smartphones powered by postmarketOS, the results are totally different. This suggests that further discontinuities may arise with the investigating of other smartphones. Regarding the origin of these differences, we assumed that the drivers are coded differently for the two phones. Since these drivers are open source, we attempted to investigate these disparities but realised that it would required significant effort, time, and expertise. Therefore, no conclusion can been drawn up to now but this could be interesting to study in the future.

6.2 GPIO KEY

The GPIO keys refer to the physical interaction between the users and the phone. Typically : the power button, volume up and volume down (displayed in Figure 6.3). By pressing one of these keys, the user transmits some information to the phone. During the postmarketOS installation, we chose a minimalist graphic interface resulting in non-activation of the touchscreen. This choice is motivated by the high tendency of discarded smartphone to have deficient screen (see Section 3.1.2). Consequently, only these three buttons can be used as physical command with the phone.

Unlike the sensors, where the measured values are registered in specific files (see Listing 6.1), here the only way to detect a physical interaction with the user is by reading the corresponding GPIO files while pressing the key. This way, an output is displayed in the command terminal which can after be read and used by a program.

Currently, the quadcopter is controlled only in terms of pitch and roll (see Figure 6.1). However the thrust (T) need also to be controlled. This corresponds to a throttle command. This command controls the power of the motors, allowing the quadcopter to increase/decrease the speed of the motor and therefore to fly faster/slower or at higher/lower altitude. This can be achieved with a counter who increases when the phone's volume button up (+) is pressed and decreased when it is the volume button down (-). Additionally, the power button is also used to send ON/OFF commands.

6.3 Estimation of the pitch and roll command

To control the drone in terms of pitch and roll, the **Samsung A6+** is initially chosen. This phone contains both an accelerometer, a gyroscope accessible through postmarketOS which embeds the necessary drivers. Using a complementary filter as exploited in Section 5.1.3, these two sensors can be merged to provide an accurate estimation of the desired angle. We will therefore try to replicate the same implementation, in order to control the drone effectively. However, since the measured sampling frequency of the SA6+ varies regarding the phone sampling frequency (see Table 6.1), we need to select first the phone sampling frequency which best suit our application.

Different experiments are realised under a different phone sampling frequency of the SA6+. For these experiments, the phone is tilted on its x and y axes resulting in a sequence of [negative pitch - positive pitch - negative roll - positive roll]. The estimation are made with the internal gyroscope and accelerometer of the SA6 through the Equation (5.3) and Equation (5.4) established before (see Section 6.3). The results for a sampling frequency of 12.5 [Hz] and 208 and 416 [Hz] are displayed in Figure 6.6.

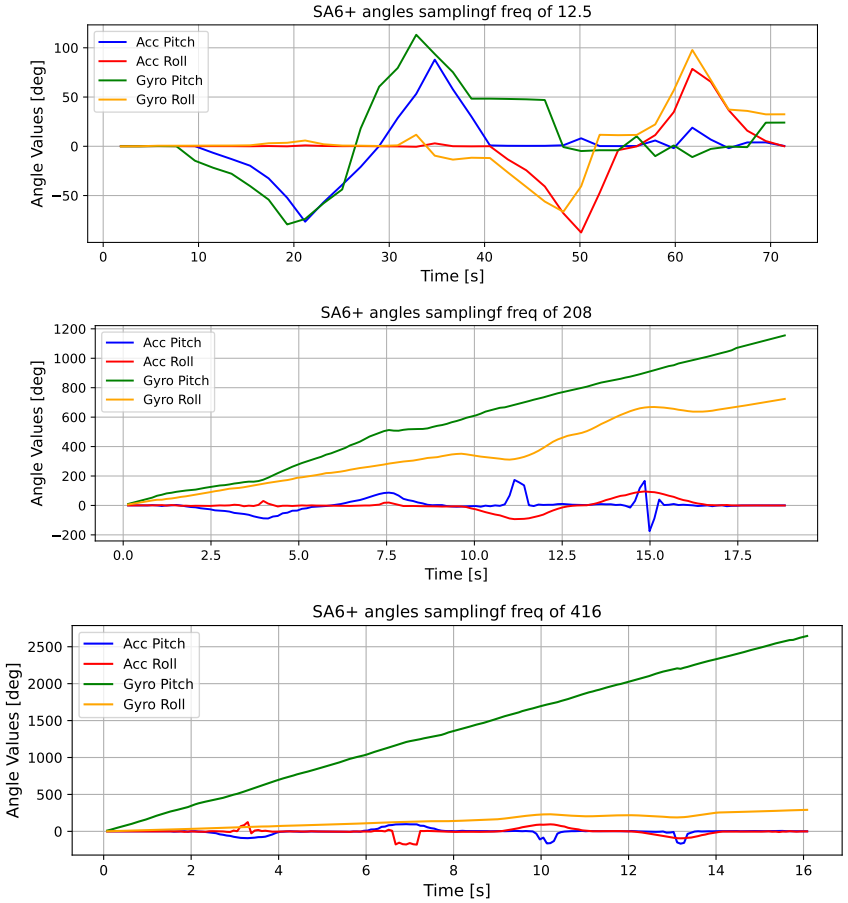
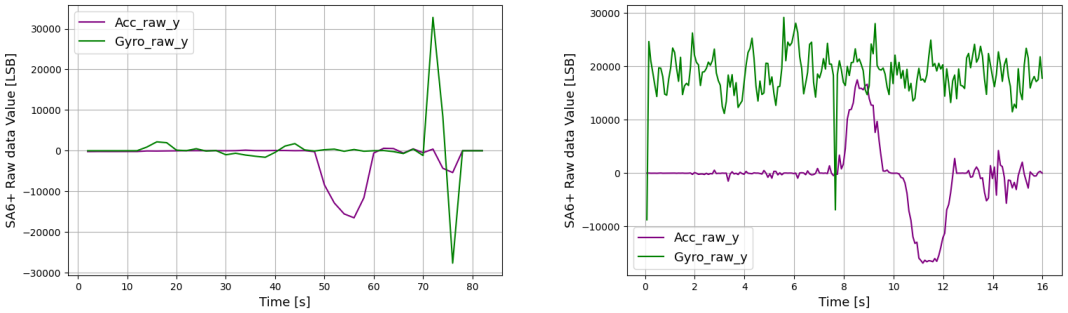


Figure 6.6: Pitch and roll estimated using the internal accelerometer and gyroscope of the SA6+ with different selected phone sampling frequency [12.5, 208, 416 Hz] and imposed sequence $[-\theta, +\theta, -\phi, +\phi]$. The angles estimation are smoother and experiment more drift with higher frequency. Using this phone, the best phone frequency is the 208 [Hz].

These experiments highlight the fact that a lower sampling frequency makes the data capture rougher. Therefore, a slow manipulation is required to obtain smooth control. Here the first test at 12.5 [Hz] (first sub-figure) is realised under 70 seconds and still provides high pics (t = 20, 30, 50, and 60 [s]) which can be limiting for the designed remote control application. Moreover, at higher frequency, the estimation is smoother, allowing finer control but as observed in the last sub-figure at 416 [Hz] the gyroscope estimation tends to drift consequently, reaching an angle of 2500 [°] in less than 20 seconds. Such a drift will impact the control.

With the aim of obtaining better control, a higher sampling frequency is privileged. However, to investigate deeper for the reason of this drift, we plot in Figure 6.7 the raw measurements of the accelerometer and gyroscope of the SA6+ with a selected phone sampling frequency of 12.5 [Hz] and 416 [Hz]. For this experiment the phone is maintain still and a slow tilt is applied follows by a faster rotation, creating an oscillation measured by the accelerometer then measured by the gyroscope.



(a) Sampling freq = 12.5 [Hz]

(b) Sampling freq = 416 [Hz]

Figure 6.7: Raw measurements of the internal accelerometer and gyroscope of the SA6+, low vs high sampling frequency compared. For higher frequency, more noise are measured.

These plots indicate that the sensors, specially the gyroscope is measuring a higher noise. This results suggest that the measurements with a lower frequency received additional filtering, reducing the measured noise. Additional investigation (e.g. drivers analysis) are required to confirm this hypothesis. Finally a sampling frequency of 208 [Hz] is selected which appears to be a good compromise between achieving smooth control inputs while minimising gyroscope noise (observable in Figure 6.6).

Now that the sampling frequency is selected, a complementary filter combining the gyro integration data and the accelerometer data can be applied to improve the pitch and roll estimation. This is applied in Figure 6.8 where the roll angle is estimated using the estimation of the accelerometer, the gyro integration and a complementary filter with an α of 0.5 is tested (giving the same weight to the gyroscope and to the accelerometer). The relative imposed sequence is positive angle roll followed by a negative roll of $\pm$ 40°.

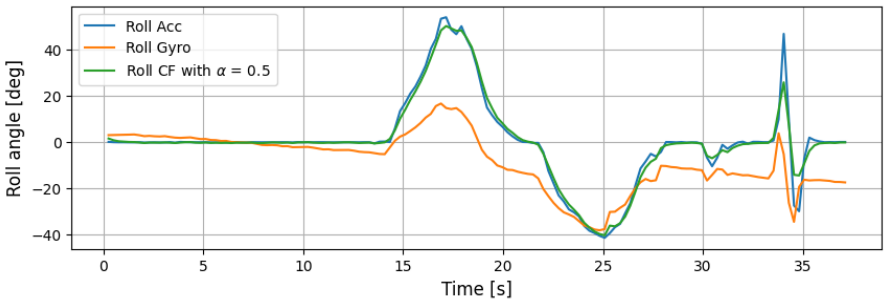


Figure 6.8: Comparison between the estimation of the roll angle using the internal sensors of the SA6+ and a CF with $\alpha = 0.5$ showing that the estimation could only be done using the accelerometer.

From these results, it can be observed that the angle estimation obtain with the complementary filter tends to be optimal when it closely matches the accelerometer's estimation (preferring higher α value over low α). This conclusion is in opposition with the configuration of the complementary for the onboard accelerometer and gyroscope observed in Section 5.1.3, which required a very low α (0.01). In fact, the accelerometer onboard the drone is highly affected by the noise created by the running motors (Figure 6.7).
In this configuration, where the phone is only controlled manually, less acceleration noise are measured which enable the internal accelerometer tends to correctly estimate the pitch and roll angles. We can therefore conclude that relying only on the phone's accelerometer for the pitch and roll estimation is feasible.

Consequently, by using only the accelerometer, the acquisition update frequency can be upgraded, resulting in smoother control since less measured values are required (see Figure 6.5). However, given that the accelerometer's measured sampling frequency on the Samsung A3 is 1 decade faster than the Samsung A6 (see Figure 6.4), it is logical to consider using the **Samsung A3** as remote controller instead of the Samsung A6. It will enhance the accuracy of the control and result in better performance. This can be visualised in the following Figure 6.9 where for 30 seconds recording,and phones at rest, the SA3 registered 10k points while the SA6 only 750.

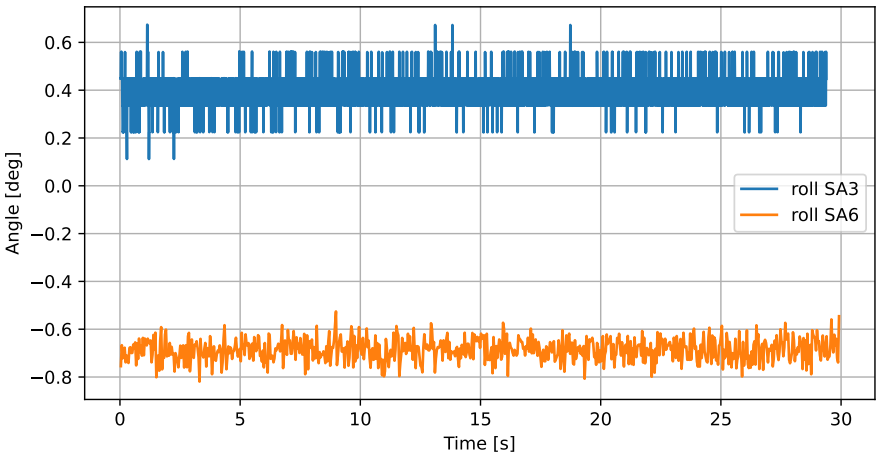


Figure 6.9: Roll estimation over time showing that the SA3 registers much more data over time than SA6. Here the sensors are non-callibrated which explain the difference measured of angles.

In the end, the investigation of the internal sensors was conducted step by step. Initially, we aimed to couple the internal gyroscope and accelerometer for better control inputs. Therefore all the analysis were made on the Samsung A6+ to investigate how to optimise the sensors measurements. In the end, we concluded that only using the accelerometer of the Samsung A3 would result in more reliable

control inputs. Nevertheless, this investigation process allowed us to create a global picture of how the internal drivers behave. Additionally, it demonstrated that postmarketOS allows us to run Python scripts and access internal sensor measurements. However, despite being open-source, the OS still contains certain *"black box"* elements, which complicate the process and need to be investigated.

These different issues faced during the process and their potential explanation are summarised Table 6.2.

	Faced issues/limitations
Samsung SA6+	Datasheet specification ≠ sensor pmOS folder, Listing 6.1
	Measured freq increases with number of request measures, Figure 6.5
	Measured noise increases with higher phone freq, Figure 6.7
Samsung A3	Datasheet specification ≠ sensor pmOS folder
	Measured freq constant regarding the selected phone freq, Figure 6.4

Table 6.2: Resumed limitation faced during the investigation process of internal sensors of the discarded smartphones associated. The SA3's sensors tends to be easier and more effective to exploit.

6.4 Wi-Fi communication

For wireless communication between the Samsung A3 and the Fairphone 2, a Wi-Fi connection is implemented. In this design, the Fairphone 2 serves as a Wi-Fi hotspot, and the Samsung A3 connects as a client. The communication protocol used between the two phones is a TCP server.

A TCP server is a program or process that runs on a computer and listens for incoming TCP connections from clients. When a client wants to communicate with the server, it initiates a TCP connection by specifying the server's IP address and port number. The server then accepts this connection request and establishes a dedicated communication channel with the client.

In our application, employing a TCP server provides several advantages. As example : once the TCP connection is established, both the server and the client can exchange data streams bidirectionally. This bidirectional communication is essential as it allows the control inputs to be sent to the quadcopter and the quadcopter's states to be received in return for performance analysis. Moreover, TCP ensures reliable delivery of data packets in the correct sequence and without errors which is also crucial for a quadcopter control.

6.5 Resumed implementation

Finally the communication setup is resumed in Figure 6.10. In this diagram, all the key communicating elements are represented. Their functions and implementation are summarised below :

- **Arduino Nano**: interfaces with the Fairphone 2 to control the drone's motors and apply commands received from the Samsung A3. On the same time, the Arduino received measurements from the external sensors and estimated the quadcopter's states via a complementary filter. Finally the Arduino calculates PWM outputs to control the motors as wanted.
- **Samsung A3** : This phone sends commands for thrust, start/stop, pitch, and roll to the Fairphone 2. A python multi-threaded program running on the Samsung A3 reads the phone's accelerometer, GPIO keys interaction and simultaneously establish a TCP connection with the Fairphone 2. Additionally, the Samsung A3 receives the drone's states (pitch, roll and motors inputs) in real time.
- **Fairphone 2** : Acting as an intermediary, the Fairphone 2 transmits control inputs from the Samsung A3 to the microcontroller and sends back the drone's status to the Samsung A3. It also generates a Wi-Fi hotspot and operates a TCP server as communication protocol between the devices. The Fairphone 2 runs a Python-based multi threaded program to manage these tasks efficiently.

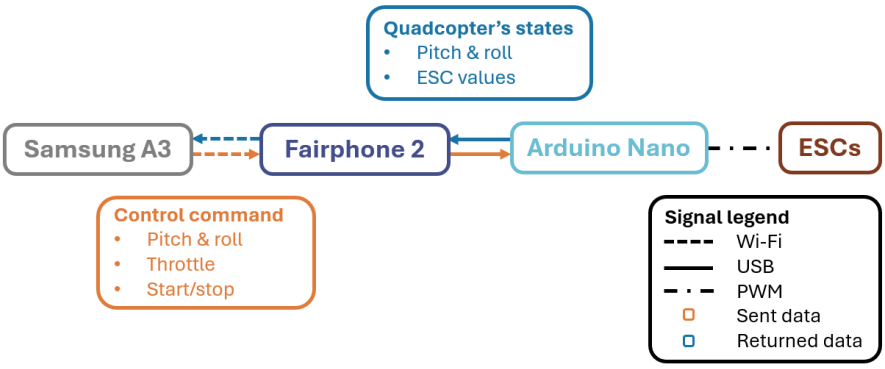


Figure 6.10: Diagram of the communication setup from the Samsung A3 to the ESCs.

Since time-delay can be a limiting factor for control (more details in Section 7.1), various algorithms are coded in such way that the transmission between the remote control and the drone is optimised. This is achieved by implementing multi-threading and increasing the transmission speed, typically by using a highest available Baud rate communication between the Arduino and the Fairphone 2. The implemented code for each of these devices (SA3, FP2 and arduino Nano) are detailed in Appendix A.

Chapter 7

Motors control algorithms and performance testing

In this section, we develop the final step necessary for controlling the quadcopter. First, we provide an overview of the control system, followed by the implementation of a PID control loop. Lastly, a test enclosing all the design process is performed an analysed.

Now that the state's estimation (Chapter 5) as well as the remote control (Chapter 6) are established. The controller remains the last component to be implemented in the control loop illustrates in Figure 7.1. The controller is made of algorithms which based on the estimated states and the control input will control the motors as desired. These algorithms, partially composed of PID controllers are processed by the microcontroller onboard.

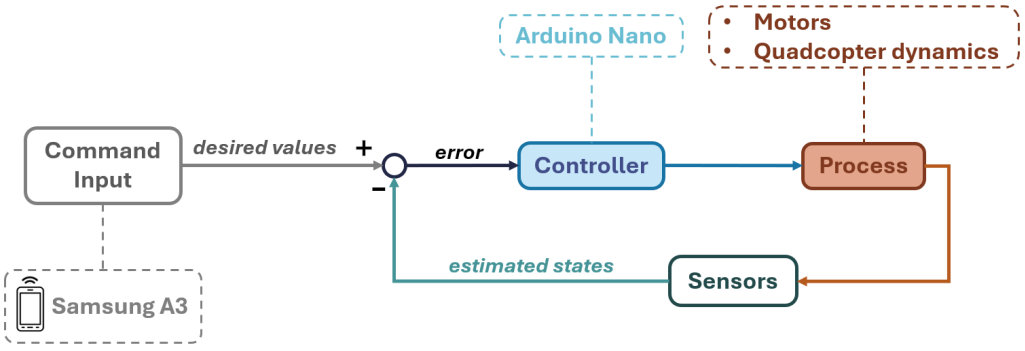


Figure 7.1: Control loop diagram of the quadcopter control highlighting the controller which operates on the Arduino nano and command the motors.

In this chapter, we first introduce a model of the dynamics of the quadcopter and the designed control system (see Section 7.1). Then, we review the PID control methodology and its implementation (see Section 7.2).

7.1 Control model and implementation

The quadcopter consists of four motors: two rotate clockwise (M2 & M4), and two rotate counterclockwise (M1 & M3) as represented in Figure 7.2). This configuration allows the drone to maintain stability in flight.

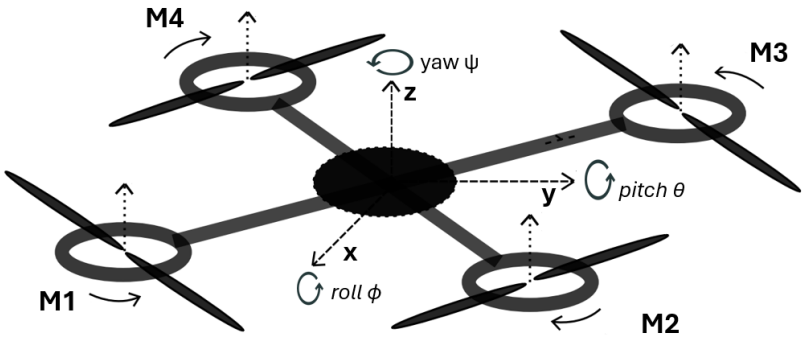


Figure 7.2: Quadcopter model labelling the motors and illustrating the Euler angle as well as the rotation direction of the motors, this configuration is the same used on the prototype. Adapted from [16]

For a quadcopter, the motors control is achieved through the control matrix Equation (7.1) shown below. This matrix takes as input the three Euler angles pitch (θ), roll (ϕ), and yaw (ψ) along with the thrust (T) and transforms these inputs into PWM signal, necessary to control the motors. In our application, as described in Chapter 6, the control of the yaw angle is not included. The last column vector will then be resumed to a 0 vector. This model has the advantage to be simple and easy to implement. However it results from a simplification of more advanced model available in [60] [61]. Therefore this simplification could be one limiting factor in the future performances.

$$\begin{bmatrix} Motor\_1 \\ Motor\_2 \\ Motor\_3 \\ Motor\_4 \end{bmatrix} = \begin{bmatrix} T_d & - & \theta_d & - & \phi_d & + & \psi_d \\ T_d & - & \theta_d & + & \phi_d & - & \psi_d \\ T_d & + & \theta_d & + & \phi_d & + & \psi_d \\ T_d & + & \theta_d & - & \phi_d & - & \psi_d \end{bmatrix} \tag{7.1}$$

Analysing this matrix, the following control logic can be observed :

- **Thrust (T):** Directly impacts all motors equally, increasing or decreasing their speed.
- **Pitch (θ):** For a positive pitch, the drone needs to tilt forward. This is achieved by reducing the speed of the front motors (M1 and M2) and increasing the speed of the rear motors (M3 and M4). For a negative pitch, the opposite adjustments are made.

- **Roll (ϕ):** For a positive roll, the drone needs to tilt to the right. This is done by reducing the speed of the right-side motors (M1 and M4) and increasing the speed of the left-side motors (M2 and M3). For a negative roll, the opposite adjustments are made.
- **Yaw (ψ):** For a positive yaw, the clockwise motors (M2 and M4) are reduced in speed, while the counterclockwise motors (M1 and M3) are increased. For a negative yaw, the opposite adjustments are made.

These scenarios are illustrated in Figure 7.3 :

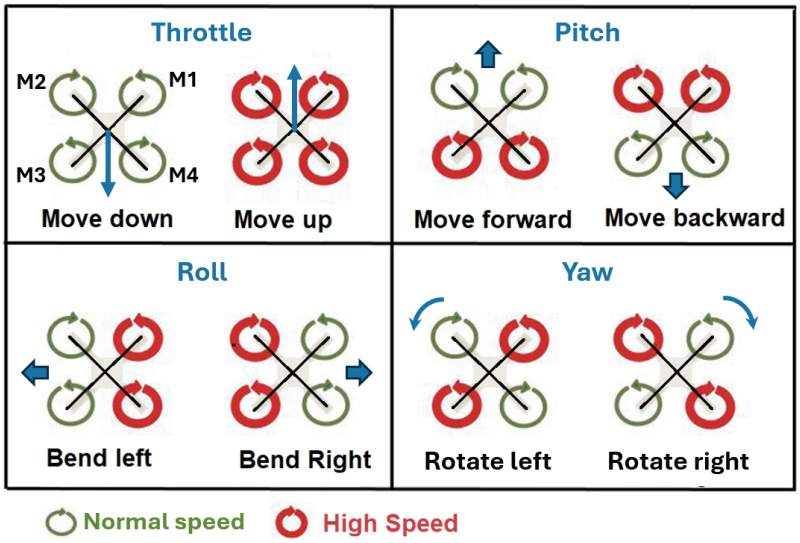


Figure 7.3: Desired dynamics (throttle, pitch, roll and yaw) of the quadcopter regarding the motor's speed. The configuration of the motors (M1,M2,M3,M4) is the same on each quadcopter diagram, adapted from [62].

The roll (ϕ_d), pitch (θ_d), and yaw (ψ_d) values displayed in the control matrix result from PID controllers outputs and stand as *desired values*. These desired values will influence the PWM values of the motors and therefore control the quadcopter as desired.

In our design, two PID controller are implemented, one controller for the pitch angle and another for the roll angle of drone. As illustrated in 7.4. The PID controller received the error $[*_e]$ between the input angles $[*_i]$ (send by the Samsung A3) and the actual orientation angles $[\hat{*}]$ (estimated by the onboard sensors). Based on these errors, the PID directly commands the motors to compensate and adjust the orientation of the drone. For effective and reactive control, this control loop must be updated as fast as possible. This required a fast acquisition data as well as fast process. As suggest by the literature, a control loop operating between 250 and 500 [Hz] should be optimal [63, 64].

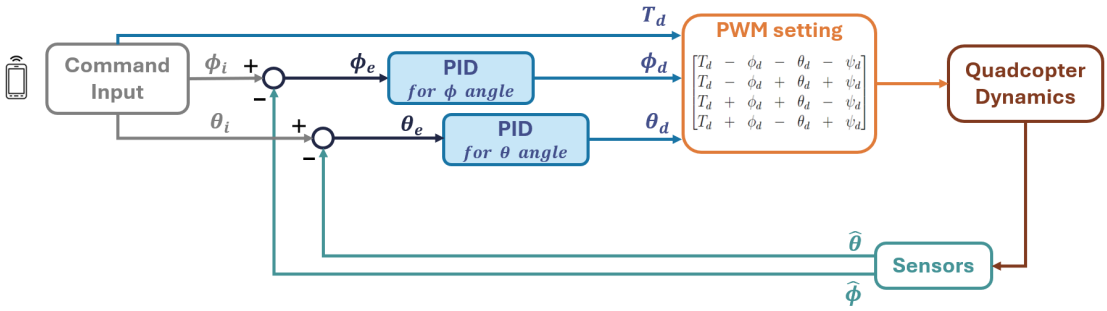


Figure 7.4: Quadcopter control based on the control matrix powering the motors (PWM setting). This matrix is alimented by two PID controllers which adjust the pitch and roll angle of the drone according to the remote command.

In our application, the main control loop is updated on the microcontroller with a maximum frequency of around 100 [Hz]. This frequency allows for consistent control but is not optimal for achieving the highest performance. The limitation in this implementation are mainly due to the limited computational power of the microcontroller. This constraint results in slower sensor data processing, slower control command readings, and a slower execution of the main control loop. By upgrading the microcontroller, the frequency can be improved, reducing time delays and enhancing the accuracy of the control system. Unfortunately, this has not yet been investigated and remains as future work.

7.2 PID Control

The PID controller is a crucial component of the control loop. This popular controller is made of three integrated controllers, each affecting the control system differently. They refer as **Kp, Ki, Kd** gain and need to be tuned accurately to obtain the desired control response.

- **Proportional Controller (P):** This component produces an output that is proportional to the current error value. The proportional controller aims to reduce the present error by adjusting the control input in direct relation to the error magnitude. The larger the error, the stronger the corrective action.
- **Integral Controller (I):** This component considers the cumulative sum of past errors. By integrating the error over time, the integrative controller addresses accumulated offset that the proportional controller alone cannot eliminate. It helps to correct any persistent, small errors by adding a component to the control input that grows with time.
- **Derivative Controller (D):** This component reacts to the rate of change of the error. By predicting future error based on its rate of change, the derivative controller introduces a damping effect, reducing the tendency of the

system to overshoot the desired position. It provides a stabilising influence by considering how quickly the error is changing.

As an example the *pitch θ PID controller* is represented in Figure 7.5 but the design is the same for the *roll PID controller* and *yaw PID controller*.

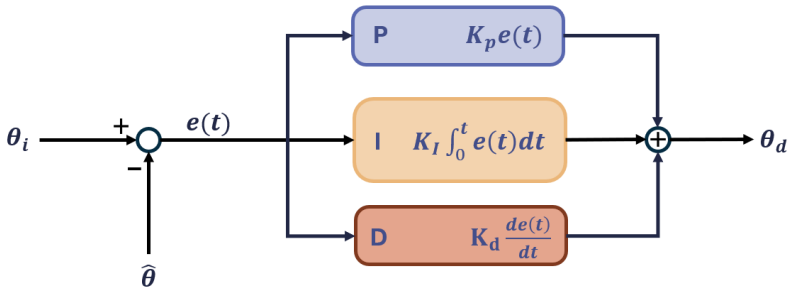


Figure 7.5: Pitch θ PID controller bloc diagram highlighting the role of the 3 gains : K_p, K_i, K_d in the control process.

Different methods can be applied to tune the PID gains. The optimal approach would be to determine the gains through simulation (typically using Matlab) [65]. However, obtaining a simulation that accurately reflects the physics of the prototype requires considerable time and effort. In our case, the PID gains were tuned manually by adjusting the K_P, K_I, and K_D components based on the drone's behaviour.

In order to adjust the drone's control in a safe environment, an experiment setup is built, as represented in Figure 7.6. This setup is made of a 3D printed plastic piece that can moved freely around a fixed metal bar using ball bearings. The bottom of the quadcopter is then clipped to this piece.

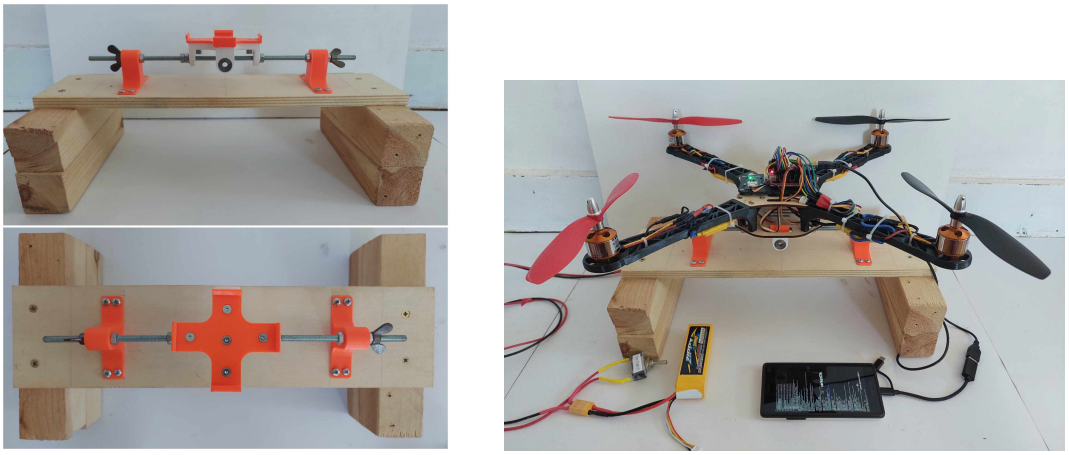


Figure 7.6: Pictures of the home made experiment setup where the prototype is fixed on one of its horizontal axe (y or x) and then remotely controlled. For safety reason, the battery is apart and a switch ON/OFF is installed.

To tune the *PID roll (ϕ) controller*, the drone is attached along its x-axis, allowing it to rotate freely around this axis. Hover mode is then established, causing the motors to spin at a constant speed. Based on the roll (ϕ) command input and response, the gains are tuned accordingly. For tuning the *PID pitch θ controller*, the drone is attached along its y-axis, and the same process is followed.

In these controlled conditions, the PID gains were manually adjusted to achieve the desired stability and responsiveness. First all the gain are set to 0, then the Proportional gain (K_p) is tuned, follow by the Integral gain (K_i) and finally the Derivative gain (K_d). This approach allowed for iterative fine-tuning in a safe environment. As preliminary work, the PID gain of the roll controller are adjusted using the experimental setup, resulting with the following gain Table 7.1. The experimental procedure to tune the PID gains is similar to the test presented in Section 7.3

Kp	**Ki**	**Kd**
2.5	0.5	0.01

Table 7.1: Adjusted PID gain for the roll control in the control test environment.

7.3 Analysis of the implemented control

The following described experiment aims to validate the implemented roll PID control while also demonstrating the effectiveness of the entire implementation process of the control loop summarised in Figure 7.7. This process includes :

- **Remote control.** Estimation of the pitch and roll inputs command using the accelerometer of the Samsung A3 and transmitted to the drone. Additionally, two commands are also transmitted: the ON/OFF command and the thrust command using the GPIO key buttons of the phone.
- **Transmission data.** Bidirectional transmission data between the SA3 (remote controller) and the FP2 (onboard) on Wi-Fi using a TCP server. Additionally, the FP2 communicates with the Arduino Nano via USB OTG.
- **Drone's states estimation.** Estimation of the orientation of the drone (pitch and roll) combining external sensors (accelerometer and gyroscope) through complementary filters executed by the microcontroller onboard.
- **Control command.** Implementation of roll PID controller and [PWM] algorithms (Equation (7.1)) to control and adjust the motors based on the error received between the measured states and the desired states.

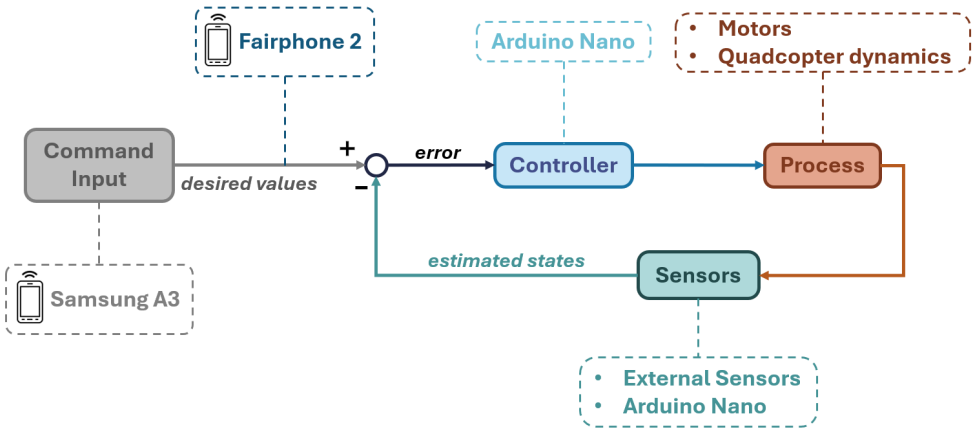


Figure 7.7: Block diagram of the final implemented control design of the prototype, highlighting the main components.

In the experiment, only the roll angle is controlled. Therefore, the prototype is attached on its x axe on the experimental setup as represented on the right of Figure 7.6. The PID gain of the roll control correspond to the adjusted gain determined in Table 7.1. The imposed inputs corresponds to the orientation of the remote controller (Samsung A3) which is rotated by hand on its x-axe, imposing a roll oscillation sequence. The results are displayed in Figure 7.8 where the imposed roll angle (in blue) is compared to the measured roll angle of the drone (in orange). Figure 7.9 illustrates the behaviour of the quadcopter during the experiment at certain time t.

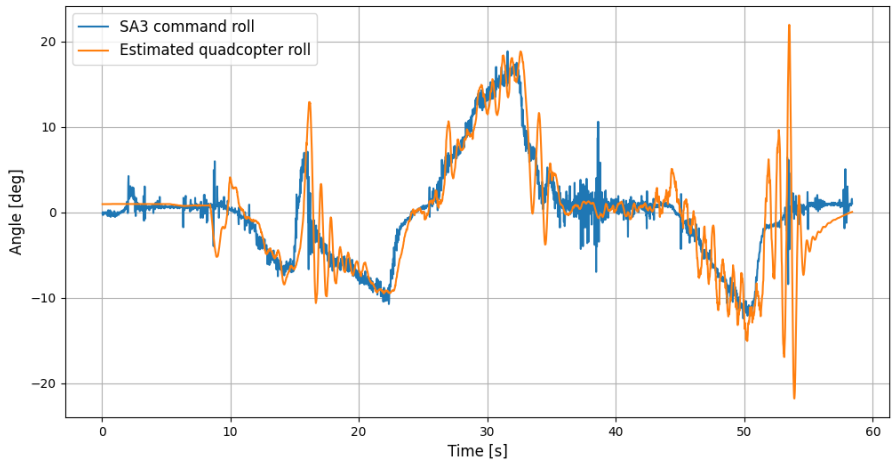


Figure 7.8: PID roll controller with adjusted PID gains. Despite some oscillations, the prototype tends to adequately follow the imposed roll angle by the SA3. The motors start at t = 9 [s] and are stop at t = 55 [s] due to control lost.

(a) t = 25 [s] the drone is inclined with a negative roll [∼10°]

(b) t = 33 [s] the drone is inclined with a positive roll [∼30°]

(c) t = 40 [s] the drone is maintain still.

(d) t = 55 [s] due to a step command, the drone is overreacting and losing control.

Figure 7.9: Pictures of the prototype at specific time during the experiment showing the behaviour of the drone respectively to the imposed roll angle.

As it can be observed in Figure 7.9, the estimated roll is closely following the commanded roll without any noticeable delay. However the measured roll exhibits significant oscillations and deviations, particularly around the 50-second mark. At t = 55 [s], the motors are stopped, due to the overreacting response of the system leading to a loss of control.

To improve the performance in term of control, specific PID tuning adjustments are necessary. Specifically, decreasing the proportional gain (K_p) will help reduce the aggressive response, and increasing the derivative gain (K_d) will add damping to mitigate oscillations. Moreover the complementary filter tends to work as attended as the roll estimation tends to be relatively accurate.

Adequately, the PID gains are tested regarding the system's response to a step input. However, testing the prototype was not the easiest part in our application. Despite the enhanced safety of the built system, various parameters impact motor control, resulting sometimes in uncontrolled behaviour which limited the number of attempted tests. Therefore, with this experiment, in addition to the PID gain tuning we aim to illustrate the good working of the all implemented system.

7.4 Limitations

Although this example shows a good integration of various components to obtain effective control of the UAV, certain limitations of this experiment should be noted.

The experiment setup does not accurately replicate the flight conditions of the drone, necessitating additional PID tuning in real conditions. Several factors contribute to this divergence:

- **Fixed Support** : the drone is attached to a fixed support, which can constrain its reactions and possibly increase acceleration measurements due to the vibration of the entire structure.
- **Ground Effect** : The test bench is positioned on the ground, introducing additional ground effect that can disturb the system.
- **Stabilisation Measure** : A small piece of foam is placed under the drone to provide a stable starting position for pre-flight calibration and enhance safety. However, this setup is altering the drone's natural behaviour and can be one of the reasons of the oscillations observed in Figure 7.8.
- **Modified weight and inertia**: For safety reasons, the battery is placed outside the drone and equipped with an ON/OFF switch. This allows the drone to be turned OFF remotely in case of uncontrolled behaviour. However, this results in a lack of weight and change of inertia for the quadcopter. This result is accentuated as the Fairphone 2 is also placed outside the setup, since on the test bench there is no place to attach the Fairphone 2 in the desired position.

These factors highlight the limitations of the conducted experiment, making it essential to perform additional PID tuning under real flight conditions to achieve optimal performance. Overall, the test shown in Figure 7.8 demonstrates that the final prototype appears to react as desired which marks this experiment (Figure 5.3) as a success.

Chapter 8

Discussion

In this chapter, a summary of the achieved work is presented. The final prototype is reviewed in terms of design and performances. Moreover, a focus is given to the different limitation met during the development process and how they could be bypass.

Along this project, many findings were realised. First, an analysis was made regarding the discarded smartphones and their available features (see Chapter 3). The studied smartphones indicate a high percentage of available components that could be reused, see Section 3.1.1. Moreover, we estimated that the main components present in UAV electronic flight computer (such as processor, IMU sensors, wireless communication module, GPS) are available in most of surveyed smartphones (see Table 3.1). Furthermore, these features tend to still be functional even after designed lifespan of the phones, see Section 3.1.2.

We also demonstrated in this same Chapter 3 that repurposing smartphones is one of the best solution to recycle a discarded smartphone. However this process could suffer from rebound effect and in some circumstances, mainly, when the electronic production is not reduced (e.g. when additional electronics is required or the cost-effective of the product makes it even more manufactured). Therefore a certain precaution must be taken regarding these terms.

Additionally, an open-source phone operating system, postmarketOS, was selected to exploit discarded smartphones in Section 3.4. Although this OS is not designed for flight computer functions, we successfully accessed the sensor measurements through trial and error, and after changing phone (Section 5.1). We concluded that postmarketOS contains certain "black box" elements concerning access to internal sensor measurements, resumed in Table 6.2. Regarding these obstacles, certain assumptions have been made about the implementation drivers under postmarketOS described in Section 6.1, but additional efforts regarding the drivers and the way the sensors's measurements under pmOS are required to confirm them.

Throughout this thesis, a prototype of quadcopter is implemented. The purpose behind is to study the feasibility of reusing discarded smartphones as flight computer using postmarketOS. The prototype results from a design process consisting of hardware components selection and software programming (Chapter 4).

As expressed in Figure 8.1. The final implemented design differs from the original design due to different limitations encountered during development.

- First the original design V1 attempted (Section 4.1), with position control, and advanced reuse of discarded smartphone could not be implemented due to various technical reasons (e.g. hardware compatibility of OS : non-access to the required smartphone's features with pmOS). The design needed to be adapted, resulting in a secondary design V2.
- This second design V2 (Section 4.3) which involves a simpler control than V1 and the integration of additional sensing electronics. However, due to time constraints and technical limitations (e.g. altitude and yaw angle estimation) the second version could not be fully implemented.
- The final prototype is a restricted version in terms of control and used sensors of the second design (V2). This final prototype, that we will rename V2.1 design corresponds to the final implemented and tested design in Section 7.3.

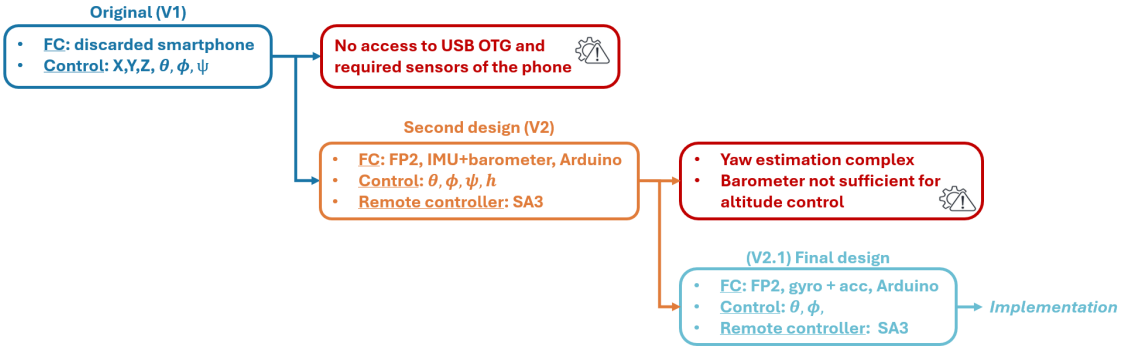


Figure 8.1: Prototype design evolution regarding the encountered issues, simplification of the control, variation in the electronics components.

In this section, we will review the design process. Each design evolution of the prototype will be individually assessed in terms of implementation and performance, along with the limitations encountered and potential ways to address them. Following the evolution of the design, we will start with the original design (V1), Section 8.1 , then proceed to the secondary design (V2), Section 8.2, and conclude by analysing the final implemented prototype (V2.1), Section 8.3. Additionally a global review of the software postmarketOS is realised, Section 8.4.

8.1 Original Version (V1)

This version concerns the first prototype developed in Chapter 4 but was not carried out for certain technical reasons that will be reviewed in this section. This model is summarised in Table 8.1. In this version (V1) a consistent amount of features available in discarded smartphones are repurposed (see Table 8.1). Therefore, we minimise the need for external sensing electronics which increases the potential of reducing electronic waste (Chapter 3).

Control	**Reused smartphone's components**	**Additional electronics**
• altitude controller (ϕ, θ, ψ) • position controller (X,Y,Z)	• Wi-Fi /4G network module • accelerometer • gyroscope • magnetometer • barometer • GPS • processor	microcontroller

Table 8.1: Characteristics of the first version prototype resumed in term of control, reused components of discarded smartphone and required additional electronics.

In this design, the remote control is ignored, assuming that input command can be correctly sent and received. The desired control is a position and attitude of the drone (see Section 2.1.2). This complex control made of two control loops required an access to a great number of sensors such as the IMU, GPS and the barometer as well as an effective execution of the program. According to this design, the states estimation, the control and the wireless communication are processed by the repurposed discarded smartphone. However, as reviewed in Section 2.2 a microcontroller is still required to power the motors.

The research made in Chapter 3 showed that most of the discarded smartphones are equipped with such sensors (except for the barometer which is more specific, see Table 3.1). Nevertheless, with postmarketOS installed, none of the studied smartphone had all these sensors available. As example, the internal sensors of the Fairphone 2 as well as the magnetometer of the Samsung SA6+ were not available due to a lack of specific drivers. Moreover, only the FP2 could communicate with a microcontroller. These limitations forced us to change of design.

This software limitation can be explained by the fact that postmarketOS is in its early development and that the developers first focused on the main telephony features (calling, SMS, Wi-Fi) leaving sensing features aside.
We tried to develop the missing lines for the Fairphone's sensors drivers but after more advanced research, we realised that it would required much more knowledge and experiences in term of Soc development. A future project involving computer engineers could facilitate the development of these drivers.

Furthermore, assuming that the necessary sensor drivers are available. The remain issues are related to the speed of data acquisition. Specifically, the acquisition time in a program environment established in Section 6.1 for the internal sensors of the Samsung SA6+ and the Samsung A3 are currently too slow to consider developing an effective attitude control system using these sensors [64]. We suppose that this issue is due to the software and the way the data are accessed. Therefore, bypassing these limitations required consequent development of the pmOS or using another OS. Finally, given that the sensor drivers are implemented in different ways, we also assume that there could be a smartphone running postmarketOS with sufficient acquisition speed. However, we cannot confirm this assumption until the OS is implemented on the phone.

Additionally, for effective control, the attitude and position loop needs to be executed at a sufficient frequency as developed in Section 7.1. In the final implemented design (V2.1), the (reduced) control algorithms are processed by an Arduino Nano, barely reaching a frequency of 100 Hz. For the original design (V1), the processing is carried out by the processor of the phone. Since the processors of smartphones are much more powerful than the Arduino Nano [66], we can assume that the main loop can be executed without failure, and the limitation for repurposing the smartphone will only concern the sensor's data acquisition.

In conclusion, by using the postmarketOS as OS to exploit the discarded smartphone, we restrained our-self with the selection of hardware (Section 4.2). Even though the three selected smartphones (SA3, SA6+ and FP2 see Section 4.2) resulted from advanced research's in term of potential accessibility of the internal features. After the implementation of the OS, all the desired features such as USB OTG and sensors drivers were not available with the selected discarded smartphones. Furthermore, development on advanced design (V2 and V2.1) allow us to conclude that even if the sensors are available, their frequency data acquisition are too slow for efficient control.
Related work based on Android development applications did not show the same limitations (Section 2.2), suggesting that a different development system might be more suitable for this type of application. Therefore, using the same discarded smartphones, two options are considered, as resumed in Figure 8.2 :

1. Improving postmarketOS and developing the necessary drivers to access the internal sensors, as well as upgrading the frequency of acquisition measurements.

2. Changing the operating software: regarding the alternative projects (see Section 2.2), keeping the Android system tends to not suffer from hardware compatibility and could be therefore an option to overcome the limitations. However as discussed in Section 3.3, Android is limited in terms of update support over time, which could be a drawback for older discarded smartphones.

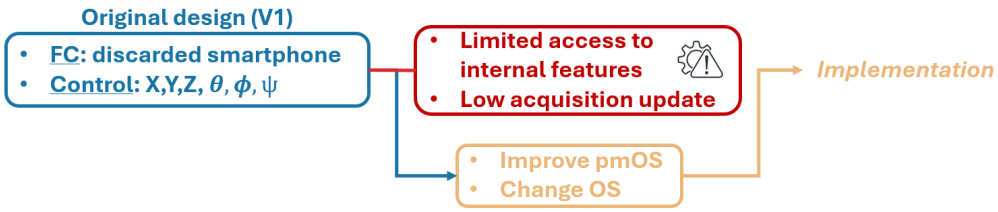


Figure 8.2: Further development to overcome the gathered limitations of accessing the internal features of discarded smartphone with pmOS.

8.2 Secondary design (V2) review

This second design presented in Section 4.3 results from adaptation due to the restriction meet during the development of the first design, while exploiting the discarded smartphone (Section 8.1).
For this second design, additional electronics are required such as barometer, gyroscope, accelerometer and magnetometer. In terms of control, the drone is autonomously controlled in terms of pitch, roll, yaw and altitude. The main loop control is now processed by the microcontroller (and not the onboard phone like the previous design V1). In this design (V2), the onboard phone (FP2) is only reused for its Wi-Fi module. Another discarded smartphone (SA3) is also reused as remote controller. The different characteristics of this version are displayed in Table 8.2.

Control	**Reused smartphone's components**	**Additional electronics**
• pitch controller • roll controller • yaw controller • altitude controller	• Quadcopter : Wi-Fi module, GPS • Remote controller : accelerometer, magnetometer, key buttons, Wi-Fi	• Arduino microcontroller • accelerometer • gyroscope • magnetometer • barometer

Table 8.2: Characteristics of the second version (V2). compared to the first design (V1), here the implemented control is restricted and additional sensors are required to compensate the non-access of internal features of discarded smartphone.

As developed in Chapter 5 and Chapter 6 in this design (V2), we have succeeded to correctly estimate the pitch and roll angle of the drone. We also have succeeded to access the internal sensors of discarded smartphones under pmOS and concluded that only using the accelerometer is more efficient to estimate the roll and pitch angle of the phone. Therefore by inclining the remote controller we are able to send input control command to the drone. For these transmission we reused the hotspot Wi-Fi of the FP2 and implemented a TCP server between the FP2 onboard and the SA3 as developed in Section 6.4. Since these developed elements are also integrated in the latest version V2.1, they will be review in the following Section 8.3.

However, during the development of this design (V2), we faced some limitations

to correctly estimate the yaw and altitude of the drone. As developed in Section 5.2 and Section 5.3, the yaw control requires complex calibration and due to a lack of time we didn't implemented it in real flight condition. Similarly for the altitude control, we conclude that the barometer sensor is not enough to estimate precisely the altitude of the drone and that additional processing or sensors are required. In this section, potential solutions are proposed in order to bypass these limitations. These solutions only stand as suggestions for further development and do not guaranty to be the optimal option.

8.2.1 Yaw control

For the yaw control, a first version is presented in Section 5.2 where the angle is estimated using a complementary filter combining the onboard magnetometer and the gyroscope integration. The test (Figure 5.12) demonstrated good performances but was not realised in real flight condition. In the experimental conditions described in Section 5.2.1, due to the high sensitivity of the sensor, perturbations of the estimation are expected. Therefore, to address this issue, an advanced calibrated magnetometer is designed consisting of two calibration steps.

1. The first calibration concerns the powered electronics onboard. This calibration removes potential bias created by the onboard electronics, performed once and saved.

2. On top of that a second calibration is performed to adjust the magnetometer for environmental magnetic perturbation factors.

Additionally, since the Fairphone 2 includes a GPS module which is available through postmarketOS adding the GPS data will enhance the heading estimation. The integration of magnetometer and geo-localisation data could then be achieved using a fusing algorithm like a complementary filter or a Kalman filter for better performances [67]. Moreover, the acquisition data of the GPS need to be fast enough in order to have an effective control of the drone [68]. Unfortunately, no tests have been yet conducted on regarding the specification of the GPS. Future developments on the exploited smartphones under postmarketOS could focus on testing and optimising the GPS module in a program environment, as we did for the internal sensors in Section 6.1.

Finally, similarly as described in Section 5.2.1, the yaw is estimated by merging the heading estimation and the gyroscope integration. This fusion is achieved through advanced filtering algorithms like the Kalman Filter, resulting in improved precision and reliability in yaw estimation.

8.2.2 Altitude

For the altitude controller, as noticed in Section 5.3, the barometer is not accurate enough for the application. This results converges with related work where they used

additional electronics like ultrasound distances sensors to improve the precision[12].

Following the objectives of limiting the addition of new electronics, the incorporation of new sensors is not considered. However, other solutions can be developed using existing sensors. For example :

- **GPS Integration:** The GPS also estimated the position in the z-axis. By fusioning the data from the pressure sensor with the GPS data, the estimation of the altitude can be improved. The Fairphone 2 includes a GPS module, which is accessible through postmarketOS. Again, this access need to be at efficient speed to provide a accurate control of the drone. Unfortunately, no tests have yet been carried out on the GPS specifications.
- **Vertical Velocity Control:** Instead of directly controlling the altitude, a control can be established on the vertical velocity. This state can be estimated by coupling the accelerometer data from the drone with the readings from the pressure sensor. The accelerometer on the drone measures the acceleration in three axes, including the vertical axis (z-axis). By integrating the vertical acceleration over time, an estimate of the vertical velocity can be made. Similarly, by differentiating the altitude measurements from the pressure sensor with respect to time, we can estimate the vertical velocity. Additionally, These estimations from the two sensors require additional complex filtering processes, such as the Kalman Filter. This control method is implemented in [59] [69] demonstrating conclusive results.

8.2.3 Remote controller's estimation

If the yaw angle and the altitude of the quadcopter can be estimated, these states need to be commanded from the remote controller. In this design (V2), this function is achieved by the Samsung A3.

- **Yaw angle**: The SA3 includes a magnetometer and a GPS module that are already accessible under pmOS (see Table 4.1). Therefore, similarly as for the yaw estimation onboard, by pre-calibrating the magnetometer and combining it with the GPS data, a yaw angle can be estimated and serve as input control. Several tests still need to be conducted, such as integrating the magnetometer's data and GPS data of the SA3 in a simple a program environment.
- **Altitude**: For this state control, using the GPIO keys as exploited in Section 6.2 can still be envisaged and adapted regarding the altitude control chosen. For example, a counter can be implemented to command the height in meters. The counter increases by one meter when the volume up (+) button is pressed and decreases by the same height when the volume down (-) button is pressed.

8.2.4 Resumed solutions and conditions

In conclusion the limitation encountered can potentially be overcome by implementing the proposed version and resumed in the Figure 8.3.

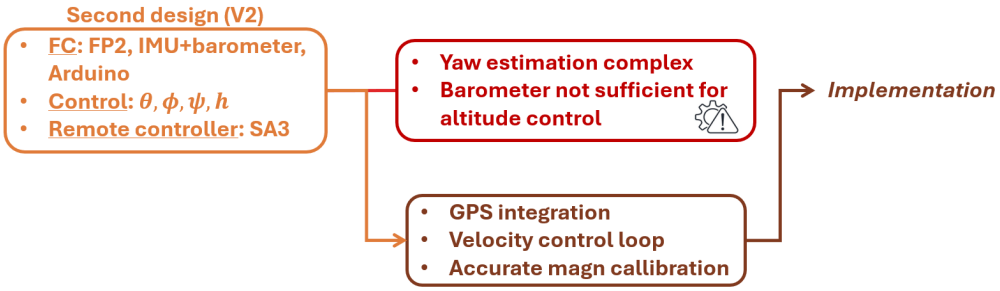


Figure 8.3: Further development to overcome the limitation of yaw and altitude control of the second design, using the barometer, magnetometer and GPS.

However these solutions imply certain conditions, such as:

- Good integration of the GPS data on the Samsung A3 and the Fairphone 2 as well as a efficient data frequency acquisition.
- Modifying the control system with an additional feedback loop for vertical velocity control.
- We also mentioned the use of Kalman Filter, we assume that they are also well integrated in the control process and do not affect the execution speed of the control loop.

8.3 Final prototype (V2.1) review

The final implemented design result from adaptation due to the limitation encountered in state's estimation yaw and altitude in the secondary design (Section 8.2). This final design is similar to the secondary design (Table 8.2), except that for this final version, only the pitch and roll control are implemented which therefore, limit the use of additional sensors (magnetometer and barometer) Table 8.3 resumed the main characteristics of this final design.

Control	Reused smartphone's components	Additional electronics
• pitch controller • roll controller	• Quadcopter : Wi-Fi module • Remote controller : accelerometer GPIO buttons, Wi-Fi	• Arduino microcontroller • accelerometer • gyroscope

Table 8.3: Characteristics third version prototype resumed in term of control, reused components of discarded smartphones and required additional electronics.

The implemented control loop of the prototype is summarised and tested in Section 7.3. In this following section, an overall review is provided, highlighting potential improvements in terms of control and design that could be considered for future work.

8.3.1 Control loop

States estimation

As developed in Section 5.1, the states estimation (pitch and roll) using the developed equations showed good estimation performances of the drone's orientation at rest (Figure 5.7). However, in flight condition, as showed in Figure 5.8, the noise created by the motors is saturating the accelerometer, affecting the angle estimation. Moreover, the test realised in Section 7.3 to control the roll angle shows that the prototype oscillates. These oscillations may be partly due to an incorrect estimate of the angle (to be comfirmed in more advanced test). Therefore more developments are required for finer control.

In our design, these angle estimations are obtained through the combining of the gyroscope and the accelerometer using a complementary filter (Section 5.1.3). This filter enhances the angle estimation minimising noisy acceleration of the accelerometer and drift of the gyroscope integration. This estimation can be improved by using more advanced and complex algorithms such as the Kalman Filter. This filter is a sophisticated version of the complementary filter, where the weight assigned to each sensor (α) is dynamically chosen at each iteration. While the Kalman Filter is significantly more efficient than the complementary filter, it also demands more computational resources, which will impact the execution loop time [57].

Remote controller

Regarding the inputs control command (pitch, roll, thrust, start, stop) sent by the remote controller (the Samsung A3), they are adequately using the different hardware resources of the discarded smartphone (see Section 6.2 and Section 6.1). The pitch and roll angle estimated by the internal accelerometer are following correctly the imposed angle as resumed in Section 6.3. Nevertheless, pressing these keys buttons introduces perturbations in the measured acceleration which can disrupt the control commands. [1].
This can be improved by combining the accelerometer with a gyroscope using a fusing filter like CP or Kalman filter as developed in Section 5.1.3. However, this approach necessitates access to a discarded smartphone with a gyroscope sensor that has a high sampling frequency for efficient control, which was not available in the studied devices under pmOS (Figure 6.5).

[1] Indeed, when the key buttons are pressed, the smartphone undergoes involuntary movements.

Finally, the action of pressing the buttons might be too slow for quick adjustments for special manoeuvres like takeoff. To overcome this issues, adjustment need to be made regarding the incrementing of the thrust counter implemented in Section 6.2, having smaller step for take-off and landing and bigger steps in flight.

PID control

In terms of quadcopter control, the final control loop is composed of a pitch and roll control (see Figure 7.4). This control is achieved through PID controllers, which are tuned using different gains via an experimental setup described in Section 7.2. Unfortunately, only the gains of the roll controller were manually optimised and tested following a trial-and-error process (Section 7.2). Since the quadcopter is symmetric, the implementation for the pitch angle is the same, with slightly different PID gains.

As described in Section 7.3, the controller showed good performance but along the experiment, some oscillations occurs, leading to a lost of control (see Figure 7.8). In this same section, we concluded that in order to achieve optimal control in real flight condition more effort and time are required.
Indeed, we demonstrated that the PID gains are not optimal and should be upgraded. Since the experimental setup does not fully reflect flight performance (Section 7.4), optimal tuning should ideally result from simulation [65]. However, to achieve realistic simulations of the prototype, an advanced mathematical model and a good characterisation of the final prototype are required.

Additionally, the control is made on the pitch and roll through PID controller. This type of control is simple but as developed in [16] an internal control loop could be inserted, controlling the derivative of the states and providing better performances. However, this control is more complex and required particular attention in the execution speed loop (since we have two loops interconnected). Alternatively, the PWM matrix Equation (7.1) results from a simplification of the physical model of the quadcopter therefore, using a more sophisticated model as developed in [70, 71] will increases control performances.

Data transmission

The bidirectional communication established between the Samsung A3 and the Fairphone 2 seems to be effective and does not suffer from any time delay. This communication is made under a TCP server on Wi-Fi signal, see 5.1.3. However, as proposed in [12], using the 4G network is more suitable for transmitting information over distances greater than 150 meters.

Furthermore, in the communication design (summarised in Section 6.5), the limiting factor appears to be the USB connection between the Arduino and the Fairphone 2, identified during the implementation phase of Section 6.4. Although a

detailed analysis of the communication protocol implemented on the Arduino has not been conducted, the current configuration relies on "printing" and "reading" data via the USB channel. This approach could be improved by adopting a more efficient communication protocol, such as encoding the transmission or considering the use of an alternative microcontroller.

8.3.2 Design

Regarding the built prototype developed in Section 4.3.4 and the final test realised in Figure 7.6, several conclusions can be drawn regarding the frame design and the choice of hardware.

Frame

The frame is constructed using a combination of 3D printed plastic PLA, laser-cut plywood. As desired in Chapter 1, this design is cost-effective, straightforward to manufacture and assemble, providing a significant advantage for easy replacement of broken parts. However, the materials used are not the lightest, resulting in a relatively heavy drone and consequently, reducing the flight time. Unfortunately, this design is particularly sensitive to the vibrations coming from the spinning motors, which affect overall stability and performance. This can be observed in Figure 5.8 where we assume that a part of the noise measured by the accelerometer is created by the rotation of the motors (and propellers).

To enhance the design, several improvements can be implemented:

- Utilising lighter and smaller screws, possibly incorporating plastic inserts, can effectively reduce weight without compromising structural integrity.
- Substituting laser-cut wood with carbon fibre parts offers a significant weight reduction while maintaining strength and rigidity.
- Replacing PLA with a lighter and more advanced material, such as a lightweight composite could reduce the overall weight.
- Enhancing the mounting of sensors with better damping structures can significantly reduce vibrations, improving sensor accuracy and overall stability.

Hardware

The choice of different hardware components was driven by affordability and availability, which introduced several limitations during the design process:

- **Electronics Fragility:** Some components proved fragile and occasionally malfunctioned without clear reasons, necessitating the replacement of 2 ESCs and one motor.

- **Motor Synchronisation Issues:** There were times where motors did not synchronise perfectly, requiring software calibration to address synchronisation issues which poses significant challenges for quadcopter stability and control.
- **Arduino Nano Limitations:** The Arduino Nano was essential because the phone cannot directly control ESCs. In our application (design V2 and V2.1), it also handled tasks such as reading external sensors and executing control algorithms.
  However, the actual attitude control showed some limitation in term of time delay. As described in Section 7.1, the main loop executed on the Arduino Nano is characterised by a frequency of 100 [Hz], which is very poor for an accurate quadcopter control. The limitation processes tend to be the capture and transfer of onboard sensor's measurements, the USB reading of the command from the FP2, as well as the computational demand of the main code. Alternative microcontrollers with more powerful processors and memory could be more suitable for the application like Arduino Nano, Teensy 4.0, ESP32 Mini. Further research and additional delay tests are required to confirm these hypotheses.

Additionally, in the sensors selection in Section 4.3.1 where external sensors are required to compensate the non-access to the specific internal features of the discarded smartphone (Section 4.2). It would have been more appropriate to use exactly the same sensors present inside the studied smartphones. This approach could have facilitated a direct comparison and better identification of limitations encountered in the investigation of internal sensors of discarded smartphones in Section 6.1.

8.3.3 Reused electronics

One of the purposes of this project was to reuse electronics from discarded smartphones. This way, we tend to limit e-waste by avoiding the production of new electronics and at the same time the expected life of the smartphone is extended, reducing its ecological impact.

With this final prototype (V2.1), a discarded smartphone is repurposed as a remote controller, reusing its sensing and communication capabilities. The research made in Chapter 3 showed that most of discarded smartphones are equipped with these same components and are still functional over time, suggesting that similar applications (e.g. sending remote control command) could be implemented with other discarded smartphones. The main limitation would be the battery, which is often defective in discarded smartphones. However, for short-term use and low-demanding programs (like the one implemented), this is manageable.

For the remote controller (Samsung A3), the Wi-Fi module, the buttons, the internal sensors and the processor are considered as reused components. Additionally, assuming the touch screen to be still functional (Section 3.1.2), an upgraded

version could be developed to reuse it as a user interface, providing a graphical experience and additional control commands. Considering that a device like the repurposed Samsung A3 could replace a new remote controller, we can consider that repurposing the smartphone is beneficial to reduce e-waste.

Regarding the quadcopter, the number of repurposed smartphone's components is less convincing. In this design, the presence of the microcontroller is inevitable but the external sensors weighted down our efforts. Furthermore, only the Wi-Fi capabilities of the Fairphone 2 is reused. And, using an entire discarded smartphone only for its Wi-Fi module might be overkill and unnecessary as it is more costly, has bigger ecological impact and heavier compared to a simple Wi-Fi board [72]. In the end, due to the additional electronics, it cannot be concluded if this designed quadcopter is better in terms of environmental impact. As future work, developing LCA of the different design prototype is a good option to enhance this conclusion.

Finally, we can conclude that given the diversity of smartphones and the work accomplished so far, it becomes clear that developing this project on a larger scale presents considerable challenges. Consequently, this approach is more suited for individual projects or research initiatives. While this limitation helps reduce the rebound effect, it also restricts the true potential of reusing discarded smartphones as flight computers in broader applications.

However, alternative UAV applications, such as Delta flying wing which demands less precision and responsiveness, present promising opportunities for repurposing discarded smartphones with the implemented design (V2.1). This type of UAV, introduced in Chapter 2, features fewer actuators and a simpler control system. Additionally, its tendency to fly at higher altitudes and over longer distances would significantly benefit from the 4G and GPS capabilities of discarded smartphones.

8.3.4 Comparison with related work

Since the final version of the quadcopter is not flying yet (Figure 7.6), comparing performance results with related work is challenging. However, some observations can still be made.

Similarly to our design, related projects also implemented PID controllers. In terms of performances, the final prototype is barely controlled on its roll angle which is a poor control related to original design (V1) and others reviewed project (summarised in Table 2.1). Most of the other implemented control system are more advanced and benefit from more accurate tuning. Many projects include cascaded control loops, where trajectory control is implemented [17]. Others, implement additional loops controlling the velocity and acceleration of the states or incorporate external perturbations in their control [16]. Moreover, for state estimation, many developed quadcopters use a Kalman filter to fuse sensor data, providing more

accurate measurements [12, 13, 16, 17]

Additionally, all the compared models use a microcontroller, which is essential for interfacing with the smartphone. However, several projects opt for a more powerful microcontroller like the Arduino Mega ADK, which is suitable for Android phone interaction compared to the Arduino Nano used in our project. Some models also incorporate additional electronics such as sensors and GPS to enhance control capabilities [11]. Overall, the related "phone-drone" projects (Table 2.1) tend to reuse more components from smartphone (processor, Wi-fi, sensors, GPS and camera) than the final implemented design V2.1 who only reuses sensors and Wi-Fi module. These differences result from the operating software use.
Most of the developed UAVs, repurposing smartphones are based on the Android platform Table 2.1. This underscores our project as a pioneering effort in exploring postmarketOS operating system for UAV development.

Finally, our conclusion on the designed quadcopter aligns with others by affirming that smartphones are particularly suited for performing tasks such as drone control but to obtain great flight performances significant efforts in testing are required.
Regarding the potential of repurposing smartphone components, [15] confirms that discarded smartphone capabilities closely matches with the requirements of quadcopters. The others related projects don't specifically analysed this point of view which limits our ability to compare the findings in this area.

8.4 Postmarket review

To exploit the discarded smartphones, the postmarketOS was installed. Using this software was a key element during the design process. Therefore in this section we resumed the pro's and con's of using this OS to repurposed discarded smartphone for future projects.

PostmarketOS offers several advantages like :

+ Its installation is relatively simple on the selected discarded smartphones, including older models like the Samsung A3 from 2015.

+ The Linux environment offers a robust platform for developing necessary programs and applications as well as accessing certain features like Wi-Fi.

+ PostmarketOS is an open-source project encouraging the sustainability [51].

However, during the development of the different version, several limitations have been encountered, which can be summarised as follows:

- PostmarketOS can only be installed on Android smartphones, excluding iPhones. This significantly restricts the potential for repurposing powerful

devices, particularly since iPhones often offer a more extensive set of features compared to typical Android smartphones. Additionally, only a small portion of Android phones are actually compatible with PostmarketOS, which severely constrained the hardware selection for our designs, even more if the objective is to repurpose discarded smartphone on large scale.

- The availability of features with pmOS varies between smartphone models, leading to discontinuities in the features available across different devices.

- Development of postmarketOS prioritises features such as Wi-Fi, SMS, and phone calls, often neglecting access to sensors. This limitation restricts its suitability for applications that require sensor integration, such as quadcopter.

- Despite being open-source, postmarketOS is primarily designed for developers, resulting in a relatively complex utilisation and comprehension process. As example, using the sensors drivers required try-error to understand how are the data encoded.

In summary, the current development state of postmarketOS is too limited for practical repurposing applications. The complexity, lack of drivers, and limited compatibility with smartphones have significantly constrained the design. Although repurposing smartphones is not the primary goal of postmarketOS, this OS holds potential for such applications. Therefore, depending on the direction in which postmarketOS evolves, this OS could become a highly advantageous software for repurposing smartphones. Nevertheless, for now, the best approach to implement the original design (V1) is to retain the original OS (Android or iOS) and develop the necessary applications on the platform.

Chapter 9

Conclusion

In conclusion, developing and building a quadcopter is a significant project. It encompass a large spectrum of engineering. This includes 3D modeling, control theory, programming, data analysis, exploiting Linux systems, prototyping, testing, and more. After different design iteration, we finally managed to produce a working prototype and to attempt some great pre-flight performances. In order to make the drone fly, the implemented controller requires more tests, advanced control algorithms and simulations.

With this project, we demonstrated that old smartphones still represent a great interest in terms of internal features. Indeed, even when these phones are classified as discarded, for most of them, their internal electronics are still reusable. Moreover, the great diversity of the available features offer a wide range of repurposing possibilities.

For our application, we decided to repurpose discarded smartphones into a flight computer and optionally a remote controller in order to design and control a quadcopter. The main objective behind this repurposing application was to reduce the production of new electronics and on the same way extend the life of smartphone. The final prototype successfully exploits the sensing and wireless communication features of two discarded smartphones. However, due to restrictive issues of software development, the final prototype included additional electronics. This highlighted that reducing e-waste through repurposing smartphones is challenging.
Discarded smartphones still offer significant advantages due to the accessibility of their features. This is particularly relevant given the widespread availability making them an abundant and cost-effective resource. Future developments should focus on making these features more exploitable.

Different operating system can be considered to exploit the smartphones features. In our design the postmarkeOS was opted as it showed great interest for providing support for older smartphones. However, the development proved that this OS is not ready yet and not adapted for such control application. To repurpose smartphones with this OS, small projects with lower demands for effectiveness should be considered.

Appendix A

Implemented code

In this appendix the different implemented codes of the prototype are detailed. These codes offer additional information regarding the reading of internal sensor's measurements and GPIO keys interaction on the Samsung A3, the TCP server established by the Fairphone 2, and finally, the main control loop executed on the Arduino Nano.

A.1 Samsung A3 - Sensor and command control

Drone Sensor and Control Script

```
#!/usr/bin/env python3

import threading
import math
import time
import socket
import csv

# File paths for sensor data
ACC_SENSOR_DATA_PATH_X = "/sys/bus/iio/devices/iio:device2/
    in_accel_x_raw"
ACC_SENSOR_DATA_PATH_Y = "/sys/bus/iio/devices/iio:device2/
    in_accel_y_raw"
ACC_SENSOR_DATA_PATH_Z = "/sys/bus/iio/devices/iio:device2/
    in_accel_z_raw"

# Conversion factors
dtr = math.pi / 180  # radians to degrees
rtd = 180 / math.pi  # degrees to radians

# CSV file configuration
save = True
csv_file = "test_roll30.csv"

```

```
# Global variables for sensor data
acc_x_data = 0
acc_y_data = 0
acc_z_data = 0

clg_x = 0
clg_y = 0
clg_z = 0

# Global variables for pitch, roll, and yaw
pitch = 0
roll = 0
yaw = 0

# Global variables for power, up, and down
power = 0
up = 0
down = 0
count = 0

# Data received from the server
received_data = "init"

# Lock for synchronization
lock = threading.Lock()

# Timestamp variables
start_time = time.time()
last_timestamp = time.time()
T = 0

# Function to read sensor data from file
def read_sensor_data_file(path):
    with open(path, "r") as sensor_file:
        return sensor_file.read().strip()

# Input device configurations
input_devices = [
    ("/dev/input/event2", "up"),
    ("/dev/input/event0", "power"),
    ("/dev/input/event1", "down")
]

# Function to read sensor data and calculate pitch and roll
def read_sensor_data():
    global T, acc_x_data, acc_y_data, acc_z_data, pitch, yaw, roll
        , last_timestamp, start_time

    while True:
        # Read sensor data
        with lock:
            acc_x_data = -float(read_sensor_data_file(
                ACC_SENSOR_DATA_PATH_X))
```

```
73              acc_y_data = -float(read_sensor_data_file(
                    ACC_SENSOR_DATA_PATH_Y))
74              acc_z_data = -float(read_sensor_data_file(
                    ACC_SENSOR_DATA_PATH_Z))
75
76          # Calculate pitch and roll
77          pitch = math.atan2(acc_x_data, acc_z_data) * 180 / math.pi
78          roll = math.atan2(-acc_y_data, acc_z_data) * 180 / math.pi
79
80          current_timestamp = time.time()
81          dt = current_timestamp - last_timestamp
82          last_timestamp = current_timestamp
83          T = current_timestamp - start_time
84
85  # Function to read input events
86  def read_input_events(device_file, device_name):
87      global power, up, down, count
88
89      with open(device_file, 'rb') as f:
90          while True:
91              event_data = f.read(16)
92              if event_data:
93                  if device_name == "up":
94                      count += 1
95                  elif device_name == "down":
96                      count -= 1
97                  else:
98                      power += 1
99              else:
100                 break
101
102 # Function to establish TCP client connection and send message
103 def tcp_client(host, port, message):
104     global received_data
105     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
            client_socket:
106         try:
107             client_socket.connect((host, port))
108             client_socket.sendall(str(message).encode('utf-8'))
109             data = client_socket.recv(1024)
110             received_data = data.decode('utf-8')
111             print(f"Received: {received_data}")
112         except Exception as e:
113             print("No connection established")
114             time.sleep(1)
115
116
117 if __name__ == "__main__":
118     host = '10.42.0.1'
119     port = 8534
120
121     if save:
122         write_to_csv()
```

```
123
124     # Start sensor data reading thread
125     sensor_thread = threading.Thread(target=read_sensor_data)
126     sensor_thread.start()
127
128     # Start input event reading threads
129     event_threads = []
130     for device_file, device_name in input_devices:
131         thread = threading.Thread(target=read_input_events, args=(
                device_file, device_name))
132         thread.start()
133         event_threads.append(thread)
134
135     try:
136         while True:
137             P = (power / 6) % 2
138             if P == 0:
139                 count = 0
140                 s = 0
141                 pitch = 0
142                 roll = 0
143             elif P > 0:
144                 s = T
145             message = [round(pitch, 2), round(roll, 2), count / 6,
                    P]
146             tcp_thread = threading.Thread(target=tcp_client, args
                =(host, port, message))
147             tcp_thread.start()
148             tcp_thread.join()
149     finally:
150         for thread in event_threads:
151             thread.join()
152         sensor_thread.join()
```

A.2 Fairphone 2 - Wi-Fi communication & data transfer code

```
1  #!/usr/bin/env python3
2
3  import serial
4  import os
5  import time
6  import socket
7  import logging
8  import queue
9  import threading
10
11 # Initialize state variables
```

```
12 init_states = [0, 0, 0, 0]
13
14 # Create queues for commands and states
15 command_queue = queue.Queue()
16 states_queue = queue.Queue()
17
18 # Configure logger
19 logger = logging.getLogger(__name__)
20 logger.critical('Hello World')
21
22 # Define the path for serial communication
23 PATH = "/dev/ttyUSB0"
24
25 # TCP server function
26 def tcp_server(host, port):
27     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
        server_socket:
28         server_socket.bind((host, port))
29         server_socket.listen()
30
31         while True:
32             client_socket, client_address = server_socket.accept()
33             data = client_socket.recv(1024)
34             if data:
35                 command_queue.put(data.decode('utf-8'))
36                 client_socket.sendall(states_queue.get().encode('
                    utf-8'))
37             client_socket.close()
38
39 # Arduino read and write function
40 def arduino_read_and_write():
41     print("Thread initiating")
42     while True:
43         if not os.path.exists(PATH):
44             print("Waiting for USB connection")
45             time.sleep(1)
46         else:
47             try:
48                 ser = serial.Serial(PATH, 250000, timeout=10)
49                 ser.flush()
50                 if not command_queue.empty():
51                     data = str(command_queue.get())
52                     print(f"Data encoded in the Arduino: {data}")
53                     data = data + "\n"
54                     ser.write(data.encode('utf-8'))
55                     line = ser.readline().decode('utf-8').rstrip()
                           # Read Arduino data
56                     states_queue.put(line)
57             except serial.SerialException as e:
58                 print(f"Error opening serial port: {e}")
59                 time.sleep(1)
60
61 # Main function
```

```
if __name__ == "__main__":
    host = '10.42.0.1'
    port = 8534

    # Initialize queues with initial states
    states_queue.put(str(init_states))
    command_queue.put(str(init_states))

    # Start threads for Arduino communication and TCP server
    arduino_thread = threading.Thread(target=
        arduino_read_and_write)
    arduino_thread.start()

    tcp_thread = threading.Thread(target=tcp_server, args=(host,
        port))
    tcp_thread.start()

    # Wait for threads to complete
    arduino_thread.join()
    tcp_thread.join()
```

A.3 Arduino Nano - quadcopter controller code

Initialisation

```
// Include necessary libraries
#include <Servo.h>
#include <Wire.h>
#include "ICM20600.h"
#include "AK09918.h"
#include <math.h>

// Define pins
#define PITCH_PIN 2
#define THROTTLE_PIN 3
#define YAW_PIN 4
#define START_PIN 5
#define ROLL_PIN 7

// Initialize variables
uint32_t LoopTimer;
unsigned long lastLoopTime = 0;
unsigned long loopCount = 0;

// Sensor register
AK09918_err_type_t err;
AK09918 ak09918;
ICM20600 icm20600(true);

```

```
25 // Sensor data arrays
26 int16_t acc[3];
27 int32_t magn[3];
28 float angle[3];
29 float angleGyro[3];
30 int16_t angleAcc[2];
31 float angleMagn;
32 int16_t rawAcc[3];
33 int16_t rawGyro[3];
34 int32_t rawMagn[3];
35 int16_t offsetGyro[3] = {0};
36 int16_t offsetAcc[3] = {0};
37 int32_t offsetMagn[3] = {0};
38 float scaleMagn[3] = {0};
39 float initialHeading = 0;
40 float heading = 0;
41 int nbrsampl = 100;
42
43 int32_t magn_offset[3];
44 int32_t magn_scale[3];
45 int32_t magn_data[3];
46 int32_t magn_avg[3];
47
48 float magnetic_declination = +2.20; // Brussels
49
50 unsigned long previousMillis = 0;
51 float dt = 0;
52
53 // Create servo objects
54 Servo esc_0;
55 Servo esc_1;
56 Servo esc_2;
57 Servo esc_3;
58
59 // Command for communication
60 String input;
61
62 // Limit values
63 const int pid_min = -200;
64 const int pid_max = 200;
65
66 // PID gains
67 const float Kp_pitch = 0;
68 const float Ki_pitch = 0;
69 const float Kd_pitch = 0;
70
71 //here, only the roll PID is implemented
72 const float Kp_roll = 2.5;
73 const float Ki_roll = 0.0;
74 const float Kd_roll = 0.5;
75
76 const float Kp_yaw = 0;
77 const float Ki_yaw = 0;
```

```
78 const float Kd_yaw = 0;
79
80 const int anti_windup_lim = 150;
81
82 // PID variables initalisation
83 float est_pitch, est_roll, est_yaw;
84 float pid_output_pitch, pid_output_roll, pid_output_yaw;
85 float error_pitch, error_roll, error_yaw;
86 float lastError_pitch = 0, lastError_roll = 0, lastError_yaw = 0;
87 float sumError_pitch = 0, sumError_roll = 0, sumError_yaw = 0;
88 float desired_pitch = 0, desired_roll = 0, desired_yaw = 0;
89 unsigned long lastTime_pitch, lastTime_roll, lastTime_yaw;
90 unsigned long deltaTime = 100; // Time interval for PID
       calculation
91
92 // Motor input init
93 int16_t esc[4]; // ESC command
94
95 // Receiver Input
96 int rcValue[6] = {1000, 1500, 1500, 1000, 1500, 1000};
97 int throttle = 1000;
98 int start = 0;
99
100 float DT = 0;
```

Setup Function

```
1  void setup() {
2    // Connect the ESC signal wire to digital pins
3    esc_0.attach(6); // Motor front-right - CCW
4    esc_1.attach(9); // Motor rear-right - CW
5    esc_2.attach(10); // Motor rear-left - CCW
6    esc_3.attach(11); // Motor front-left - CW
7
8    // Initialize time for the integration for the PID
9    lastTime_pitch = millis();
10   lastTime_roll = millis();
11   lastTime_yaw = millis();
12
13   Wire.begin();
14   Serial.begin(250000);
15
16   // Connection with the sensors
17   icm20600.setGyroScaleRange(RANGE_500_DPS);
18   icm20600.setAccScaleRange(RANGE_8G);
19   err = ak09918.initialize();
20   icm20600.initialize();
21   ak09918.switchMode(AK09918_POWER_DOWN);
22   ak09918.switchMode(AK09918_CONTINUOUS_100HZ);
23
```

```
  err = ak09918.isDataReady();
  while (err != AK09918_ERR_OK) {
      Serial.println("Waiting Sensor");
      delay(100);
      err = ak09918.isDataReady();
  }
  callibrateAcc();
  delay();
  callibrateGyro();
  delay();
  LoopTimer = micros();
  Serial.println("init complete");
}
```

Main Loop Function

```
void loop() {
  unsigned long currentMillis = micros();
  dt = (currentMillis - previousMillis) / 1000000.0;
  previousMillis = currentMillis;

  getAcc_angle();
  getGyro_angle(dt);
  getMagn_angle();
  getFusion_angle(dt);

  est_pitch = angle[0];
  est_roll = angle[1];
  est_yaw = angle[2];

  // reading inputs from serial
  if (Serial.available()) {
    input = Serial.readStringUntil('\n');
    input.trim();
    int startCommaIndex = input.indexOf('[');
    int firstCommaIndex = input.indexOf(',', startCommaIndex + 1);
    int secondCommaIndex = input.indexOf(',', firstCommaIndex + 1)
       ;
    int thirdCommaIndex = input.indexOf(',', secondCommaIndex + 1)
       ;
    int endCommaIndex = input.indexOf(']', thirdCommaIndex + 1);

    if (firstCommaIndex > 0 && secondCommaIndex > 0 &&
       thirdCommaIndex > 0) {
      String pitchStr = input.substring(startCommaIndex + 1,
         firstCommaIndex);
      String rollStr = input.substring(firstCommaIndex + 1,
         secondCommaIndex);
      String throttleStr = input.substring(secondCommaIndex + 1,
         thirdCommaIndex);
```

```
29        String startStr = input.substring(thirdCommaIndex + 1,
              endCommaIndex);
30
31        // map the inputs into desired states
32        desired_pitch = pitchStr.toFloat();
33        desired_roll = rollStr.toFloat();
34        start = startStr.toInt();
35        throttle = 1000 + 50 * throttleStr.toInt();
36      }
37    }
38
39    esc[0] = 1000;
40    esc[1] = 1000;
41    esc[2] = 1000;
42    esc[3] = 1000;
43
44    if (start) {
45      int pwm_min = 1050;
46      int pwm_max = 1800;
47
48      computePID_pitch();
49      computePID_roll();
50      computePID_yaw();
51
52      throttle = constrain(throttle, 1100, 1500);
53      \\PMW matrix control
54      esc[0] = throttle - pid_output_pitch - pid_output_roll -
            pid_output_yaw;
55      esc[1] = throttle - pid_output_pitch + pid_output_roll +
            pid_output_yaw;
56      esc[2] = throttle + pid_output_pitch + pid_output_roll -
            pid_output_yaw;
57      esc[3] = throttle + pid_output_pitch - pid_output_roll +
            pid_output_yaw;
58
59      \\ safety constrain
60      esc[0] = constrain(esc[0], pwm_min, pwm_max);
61      esc[1] = constrain(esc[1], pwm_min, pwm_max);
62      esc[2] = constrain(esc[2], pwm_min, pwm_max);
63      esc[3] = constrain(esc[3], pwm_min, pwm_max);
64    }
65
66    \\write the states so the we can follow the behaviour of the
          drone
67    esc_0.write(esc[0]);
68    esc_1.write(esc[1]);
69    esc_2.write(esc[2]);
70    esc_3.write(esc[3]);
71
72    Serial.print(start);
73    Serial.print(throttle);
74    Serial.print(est_pitch);
75    Serial.print(est_roll);
```

```
  Serial.print(esc[0]);
  Serial.print(esc[1]);
  Serial.print(esc[2]);
  Serial.println(esc[3]);
}
```

PID Computation Functions

```
void computePID_roll(){
  unsigned long currentTime = millis();
  unsigned long elapsedTime = (currentTime - lastTime_roll);

  if (elapsedTime >= deltaTime) {
    error_roll = desired_roll - est_roll;
    float dInput = (est_roll - lastError_roll) / elapsedTime;

    //anti-winding
    if (abs(pid_output_roll) < anti_windup_lim){
      sumError_roll += (error_roll * elapsedTime);
    }
    if (start>1500){
      sumError_roll = 0;
    }
    pid_output_roll = Kp_roll * error_roll + Ki_roll *
       sumError_roll + Kd_roll * dInput;

    pid_output_roll = constrain(pid_output_roll, pid_min, pid_max)
       ;

    lastError_roll = est_roll;
    lastTime_roll = currentTime;
  }
}

void computePID_pitch() {} //same logic as for PID roll control

void computePID_yaw() {} //same logic as for the PID roll and
   pitch control
```

Sensor Functions

```
void getAcc_angle() {
  icm20600.getAcceleration(&rawAcc[0], &rawAcc[1], &rawAcc[2]);
  angleAcc[0] = atan2(-rawAcc[0], rawAcc[2]) * 180 / PI;
  angleAcc[1] = atan2(rawAcc[1], rawAcc[2]) * 180 / PI;
}
```

```
6
7  void getGyro_angle(float dt) {
8    icm20600.getGyroscope(&rawGyro[0], &rawGyro[1], &rawGyro[2]);
9    rawGyro[0] -= offsetGyro[0];
10   rawGyro[1] -= offsetGyro[1];
11   rawGyro[2] -= offsetGyro[2];
12   angleGyro[0] += rawGyro[1] * dt;
13   angleGyro[1] += rawGyro[0] * dt;
14   angleGyro[2] += rawGyro[2] * dt;
15 }
16
17 void getMagn_angle() {
18   ak09918.getData(&rawMagn[0], &rawMagn[1], &rawMagn[2]);
19   rawMagn[0] = (rawMagn[0] - offsetMagn[0]) * scaleMagn[0];
20   rawMagn[1] = (rawMagn[1] - offsetMagn[1]) * scaleMagn[1];
21   rawMagn[2] = (rawMagn[2] - offsetMagn[2]) * scaleMagn[2];
22   heading = -(atan2(rawMagn[1], rawMagn[0]) * 180 / PI +
        magnetic_declination);
23   angleMagn = heading - initialHeading;
24 }
25
26 void getFusion_angle(float dt) {
27   // Complementary Filter
28   float alpha = 0.01;
29   float beta = 0.1;
30   angle[0] = (angle[0] + rawGyro[1] * dt) * (1 - alpha) + angleAcc
        [0] * alpha;
31   angle[1] = (angle[1] + rawGyro[0] * dt) * (1 - alpha) + angleAcc
        [1] * alpha;
32   angle[2] = (angle[2] + rawGyro[2] * dt) * (1 - beta) + angleMagn
         * beta;
33 }
```

Calibration Functions

```
1  void callibrateAcc() {
2    for (int i = 0; i < 1000; i++) {
3      icm20600.getAcceleration(&rawAcc[0], &rawAcc[1], &rawAcc[2]);
4      offsetAcc[0] += rawAcc[0];
5      offsetAcc[1] += rawAcc[1];
6      offsetAcc[2] += rawAcc[2] - 1000;
7    }
8    offsetAcc[0] /= 1000;
9    offsetAcc[1] /= 1000;
10   offsetAcc[2] /= 1000;
11 }
12
13 void callibrateGyro() {
14
15   for (int i = 0; i < 1000; i++) {
```

```
    icm20600.getGyroscope(&rawGyro[0], &rawGyro[1], &rawGyro[2]);
    offsetGyro[0] += rawGyro[0];
    offsetGyro[1] += rawGyro[1];
    offsetGyro[2] += rawGyro[2];
  }
  offsetGyro[0] /= 1000;
  offsetGyro[1] /= 1000;
  offsetGyro[2] /= 1000;
}

void callibrateCompass() {
  float x = 0;
  float y = 0;
  int32_t magn_min[3] = {0};
  int32_t magn_max[3] = {0};
  float avg_delta[3];
  uint32_t timeStart = 0;
  uint32_t timeout = 20000;
  ak09918.getData(&rawMagn[0], &rawMagn[1], &rawMagn[2]);
  magn_min[0] = magn_max[0] = rawMagn[0];
  magn_min[1] = magn_max[1] = rawMagn[1];
  magn_min[2] = magn_max[2] = rawMagn[2];
  delay(3000);
  timeStart = millis();

  while ((millis() - timeStart) < timeout) {
    ak09918.getData(&rawMagn[0], &rawMagn[1], &rawMagn[2]);
    for (int i = 0; i < 3; i++) {
      if (magn_min[i] > rawMagn[i]) {
          magn_min[i] = rawMagn[i];
      }
      if (magn_max[i] < rawMagn[i]) {
          magn_max[i] = rawMagn[i];
      }
    }
  }
  avg_delta[0] = (magn_max[0] - magn_min[0]) / 2;
  avg_delta[1] = (magn_max[1] - magn_min[1]) / 2;
  avg_delta[2] = (magn_max[2] - magn_min[2]) / 2;
  float Delta = (avg_delta[0] + avg_delta[1] + avg_delta[2]) / 3;

  for (int i = 0; i < 3; i++) {
    offsetMagn[i] = (magn_max[i] + magn_min[i]) / 2;
    scaleMagn[i] = Delta / avg_delta[i];
  }

  for (int i = 0; i < nbrsampl; i++) {
    ak09918.getData(&rawMagn[0], &rawMagn[1], &rawMagn[2]);
    x += (rawMagn[0] - offsetMagn[0]) * scaleMagn[0];
    y += (rawMagn[1] - offsetMagn[1]) * scaleMagn[1];
  }
  x /= nbrsampl;
  y /= nbrsampl;
```

```
69
70    initialHeading = -(atan2(y, x) * 180 / PI + magnetic_declination
          );
71  }
```

Bibliography

[1] "International e-waste day: Of ~16 billion mobile phones possessed worldwide, ~5.3 billion will become waste in 2022 | WEEE forum." (), [Online]. Available: https://weee-forum.org/ws_news/of-16-billion-mobile-phones-possessed-worldwide-5-3-billion-will-become-waste-in-2022/ (visited on 07/29/2024).

[2] R. Ahirwar and A. K. Tripathi, "E-waste management: A review of recycling process, environmental and occupational health hazards, and potential solutions," *Environmental Nanotechnology, Monitoring & Management*, vol. 15, p. 100 409, May 1, 2021, ISSN: 2215-1532. DOI: 10.1016/j.enmm.2020.100409. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215153220303925 (visited on 07/29/2024).

[3] "How many people own smartphones? (2024-2029)," Exploding Topics. (Nov. 19, 2021), [Online]. Available: https://explodingtopics.com/blog/smartphone-stats (visited on 04/14/2024).

[4] K. Shetty and S. Bhat, "Advancement and contribution of mobile smartphones to the consumer," *International Journal of Case Studies in Business, IT, and Education*, pp. 699–713, Dec. 15, 2022. DOI: 10.47992/IJCSBE.2581.6942.0227.

[5] S. Knight. "How often do people upgrade their phone? (2023 statistics)," SellCell.com Blog. (Nov. 22, 2023), [Online]. Available: https://www.sellcell.com/blog/how-often-do-people-upgrade-their-phone-2023-statistics/ (visited on 04/15/2024).

[6] M. Ayamga, S. Akaba, and A. A. Nyaaba, "Multifaceted applicability of drones: A review," *Technological Forecasting and Social Change*, vol. 167, p. 120 677, Jun. 1, 2021, ISSN: 0040-1625. DOI: 10.1016/j.techfore.2021.120677. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162521001098 (visited on 04/14/2024).

[7] F. Nex *et al.*, "UAV in the advent of the twenties: Where we stand and what is next," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 184, pp. 215–242, Feb. 1, 2022, ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2021.12.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271621003282 (visited on 07/17/2024).

[8] "Unmanned aerial vehicle [UAV] market growth & share, 2030." (), [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/unmanned-aerial-vehicle-uav-market-101603 (visited on 07/17/2024).

[9] M. P. Stewart and S. T. Martin, "Unmanned aerial vehicles: Fundamentals, components, mechanics, and regulations," in *Unmanned Aerial Vehicles*, N. Barrera, Ed., Chapter 1, New York, NY: Nova Science Publishers, Inc., 2021, ISBN: 978-1-53618-900-1. [Online]. Available: https://www.researchgate.net/publication/351037538_In_Unmanned_Aerial_Vehicles_UNMANNED_AERIAL_VEHICLES_FUNDAMENTALS_COMPONENTS_MECHANICS_AND_REGULATIONS.

[10] P. Degond, A. Diez, and M. Na, *Bulk topological states in a new collective dynamics model*. Jan. 22, 2021.

[11] S. Erhard, K. E. Wenzel, and A. Zell, "Flyphone: Visual self-localisation using a mobile phone as onboard image processor on a quadrocopter," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 451–465, Jan. 1, 2010, ISSN: 1573-0409. DOI: 10.1007/s10846-009-9360-8. [Online]. Available: https://doi.org/10.1007/s10846-009-9360-8 (visited on 05/19/2024).

[12] P. Wong, D. Nguyen, A. Abukmail, R. Brown, R. Ryan, and M. Pagnutti, "Low cost unmanned aerial vehicle monitoring using smart phone technology," pp. 286–291, Apr. 2015. DOI: 10.1109/ITNG.2015.52. [Online]. Available: https://ieeexplore.ieee.org/document/7113487 (visited on 05/19/2024).

[13] G. Loianno *et al.*, "Smartphones power flying robots," pp. 1256–1263, Sep. 2015. DOI: 10.1109/IROS.2015.7353530. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7353530 (visited on 12/05/2023).

[14] M. Leichtfried, C. Kaltenriner, A. Mossel, and H. Kaufmann, *Autonomous Flight using a Smartphone as On-Board Processing Unit in GPS-Denied Environments*. Dec. 2, 2013, Journal Abbreviation: ACM International Conference Proceeding Series Publication Title: ACM International Conference Proceeding Series, ISBN: 978-1-4503-2106-8. DOI: 10.1145/2536853.2536898.

[15] V. A. Isuru, M. A. I. M. Dharmadasa, D. V. B. C. Jayasinghe, and C. I. Keppitiyagama, "Reusing discarded-smartphone capabilities on quadcopters: The rationale, benefits and issues," pp. 229–236, Sep. 2016, ISSN: 2472-7598. DOI: 10.1109/ICTER.2016.7829923. [Online]. Available: https://ieeexplore.ieee.org/document/7829923/authors#authors (visited on 02/27/2024).

[16] A. Astudillo, P. Muñoz, F. Álvarez, and E. Rosero, "Altitude and attitude cascade controller for a smartphone-based quadcopter," pp. 1447–1454, Jun. 2017. DOI: 10.1109/ICUAS.2017.7991400. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7991400 (visited on 12/04/2023).

[17] A. Astudillo, B. Bacca, and E. Rosero, "Optimal and robust controllers design for a smartphone-based quadrotor," pp. 1–6, Oct. 2017. DOI: 10.1109/CCAC.2017.8276392. [Online]. Available: https://ieeexplore.ieee.org/document/8276392 (visited on 12/04/2023).

[18] "PhoneDrone ethos - a whole new dimension for your smartphone by xCraft — kickstarter." (), [Online]. Available: https://www.kickstarter.com/projects/137596013/phonedrone-ethos-a-whole-new-dimension-for-your-sm?lang=fr (visited on 05/19/2024).

[19] S. Prabhu N and R. Majhi, "Disposal of obsolete mobile phones: A review on replacement, disposal methods, in-use lifespan, reuse and recycling," *Waste Management & Research*, vol. 41, no. 1, pp. 18–36, Jan. 1, 2023, Publisher: SAGE Publications Ltd STM, ISSN: 0734-242X. DOI: 10.1177/0734242X221105429. [Online]. Available: https://doi.org/10.1177/0734242X221105429 (visited on 07/29/2024).

[20] J. Switzer, E. Siu, S. Ramesh, R. Hu, E. Zadorian, and R. Kastner, "Renée: New life for old phones," *IEEE Embedded Systems Letters*, vol. 14, no. 3, pp. 135–138, 2022.

[21] R. Khattak. "The environmental impact of e-waste," Earth.Org. (Mar. 13, 2023), [Online]. Available: https://earth.org/environmental-impact-of-e-waste/ (visited on 12/18/2023).

[22] M. Ercan, J. Malmodin, P. Bergmark, E. Kimfalk, and E. Nilsson, *Life Cycle Assessment of a Smartphone*. Jan. 1, 2016. DOI: 10.2991/ict4s-16.2016.15.

[23] P. Singhal, "Life cycle environmental issues of mobile phones," Espoo, Finland, 2005.

[24] T. Zink and R. Amirtharajah, "Life cycle aware computing: Reusing silicon technology," in *Proceedings of the International Symposium on Electronics and the Environment*, 2014, pp. 99–104. DOI: 10.1145/2565585.2565598.

[25] A. Unknown, "Circular economy practices in the waste electrical and electronic equipment industry," *Unknown Journal*, vol. Unknown, Unknown, 2022. DOI: Unknown.

[26] R. Seif, F. Z. Salem, and N. K. Allam, "E-waste recycled materials as efficient catalysts for renewable energy technologies and better environmental sustainability," *Environment, Development and Sustainability*, Jan. 18, 2023, ISSN: 1573-2975. DOI: 10.1007/s10668-023-02925-7. [Online]. Available: https://doi.org/10.1007/s10668-023-02925-7 (visited on 11/30/2023).

[27] T. Coughlan, C. Fitzpatrick, and M. McMahon, "Repurposing end of life notebook computers from consumer weee as thin client computers – a hybrid end of life strategy for the circular economy in electronics," 2018.

[28] J. Y. Oliver, R. Amirtharajah, V. Akella, R. Geyer, and F. T. Chong, "Life cycle aware computing: Reusing silicon technology," *IEEE Computer Society*, 2007.

[29] D. R. Cooper and T. G. Gutowski, "The environmental impacts of reuse: A review," *Journal of Industrial Ecology*, vol. 00, no. 0, pp. 1–18, 2015. DOI: 10.1111/jiec.12388.

[30] E. Rodriguez, O. Carrasquillo, C. Lee, J. Lee, and A. Zhou, "iGo green: A life cycle assessment of apple's iPhone," *iConference 2015 Proceedings*, Mar. 15, 2015, Publisher: iSchools. [Online]. Available: https://hdl.handle.net/2142/73760 (visited on 11/30/2023).

[31] A. Joshi, A. Gupta, S. Verma, A. R. Paul, A. Jain, and N. Haque, "Life cycle based greenhouse gas footprint assessment of a smartphone," *IOP Conference Series: Earth and Environmental Science*, vol. 795, no. 1, p. 012 028, Jun. 2021, Publisher: IOP Publishing, ISSN: 1755-1315. DOI: 10.1088/1755-1315/795/1/012028. [Online]. Available: https://dx.doi.org/10.1088/1755-1315/795/1/012028 (visited on 11/30/2023).

[32] I. M. S. K. Ilankoon, Y. Ghorbani, M. N. Chong, G. Herath, T. Moyo, and J. Petersen, "E-waste in the international context - a review of trade flows, regulations, hazards, waste management strategies and technologies for value recovery," *Waste Management (New York, N.Y.)*, vol. 82, pp. 258–275, Dec. 2018, ISSN: 1879-2456. DOI: 10.1016/j.wasman.2018.10.018.

[33] T. Zink, F. Maker, R. Geyer, R. Amirtharajah, and V. Akella, "Comparative life cycle assessment of smartphone reuse: Repurposing vs. refurbishment," *The International Journal of Life Cycle Assessment*, vol. 19, pp. 1099–1109, 2014. DOI: 10.1007/s11367-014-0720-7.

[34] F. Maker and T. Zink, "Repurposing end of life notebook computers," in *Proceedings of the International Symposium on Electronics and the Environment*, 2014, pp. 99–104. DOI: 10.1109/ISEE.2014.6870347.

[35] G. Challen *et al.*, "The mote is dead. long live the discarded smartphone!" In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile '14)*, ACM, New York, NY, USA: ACM, 2014, pp. 1–6, ISBN: 978-1-4503-2742-8. DOI: 10.1145/2565585.2565598. [Online]. Available: http://dx.doi.org/10.1145/2565585.2565598.

[36] "GSMArena.com - mobile phone reviews, news, specifications and more..." (), [Online]. Available: https://www.gsmarena.com/ (visited on 08/03/2024).

[37] A. Jhaveri. "Why do most smartphones not have FM radio anymore?" Mashable India. (Feb. 13, 2019), [Online]. Available: https://in.mashable.com/tech/2154/why-do-most-smartphones-not-have-fm-radio-anymore (visited on 07/29/2024).

[38] M. Proske, E. Poppe, and M. Jaeger-Erben, *The smartphone evolution - an analysis of the design evolution and environmental impact of smartphones*. Sep. 1, 2020.

[39] E. Jardim, "From smart to senseless: The global impact of 10 years of smartphones," Greenpeace Inc., Washington, D.C., Feb. 2017, Edited by Maria Elena De Matteo. [Online]. Available: https://www.greenpeace.org/usa/reports/from-smart-to-senseless/.

[40] M. Brannon, P. Graeter, D. Schwartz, and J. R. Santos, "Reducing electronic waste through the development of an adaptable mobile device," in *2014 Systems and Information Engineering Design Symposium (SIEDS)*, Apr. 2014, pp. 57–62. DOI: 10.1109/SIEDS.2014.6829871. [Online]. Available: https://ieeexplore.ieee.org/document/6829871 (visited on 06/18/2024).

[41] H. Huang, X. Tong, Y. Cai, and H. Tian, "Gap between discarding and recycling: Estimate lifespan of electronic products by survey in formal recycling plants in china," *Resources, Conservation and Recycling*, vol. 156, p. 104 700, May 1, 2020, ISSN: 0921-3449. DOI: 10.1016/j.resconrec.2020.104700. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921344920300227 (visited on 07/29/2024).

[42] "Smartphones." (), [Online]. Available: https://www.backmarket.be/fr-be/l/smartphones/6c290010-c0c2-47a4-b68a-ac2ec2b64dca?msclkid=a100c81693f1175c07e0aebe81c34cf5&utm_source=bing&utm_medium=cpc&utm_campaign=BE_SA_SEARCH_M_GEN_Android_Others_New_2023&utm_term=smartphone&utm_content=Smartphones+-+All+Keywords (visited on 07/29/2024).

[43] M. Cordella, F. Alfieri, C. Clemm, and A. Berwald, "Durability of smartphones: A technical analysis of reliability and repairability aspects," *Journal of Cleaner Production*, vol. 286, p. 125 388, Mar. 1, 2021, ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2020.125388. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652620354342 (visited on 04/15/2024).

[44] T. Makov and D. Font Vivanco, "Does the circular economy grow the pie? the case of rebound effects from smartphone reuse," *Frontiers in Energy Research*, vol. 6, 2018, ISSN: 2296-598X. DOI: 10.3389/fenrg.2018.00039. [Online]. Available: https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2018.00039.

[45] "Mobile operating system market share worldwide," StatCounter Global Stats. (), [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide (visited on 06/16/2024).

[46] J. Fedewa. "How long will my android phone be supported with updates?" How-To Geek. Section: Android. (May 2, 2022), [Online]. Available: https://www.howtogeek.com/797200/how-long-will-my-android-phone-be-supported-with-updates/ (visited on 07/29/2024).

[47] "Here are the phone update policies from every major android manufacturer," Android Authority. (Jun. 17, 2024), [Online]. Available: https://www.androidauthority.com/phone-update-policies-1658633/ (visited on 07/29/2024).

[48] "How long does apple support iPhones? | RefurbMe blog." (Jun. 11, 2024), [Online]. Available: https://www.refurb.me/blog/how-long-does-apple-support-iphones (visited on 06/17/2024).

[49] LineageOS. "LineageOS – LineageOS android distribution." (), [Online]. Available: https://lineageos.org/ (visited on 05/08/2024).

[50] "Home | Ubuntu Touch," UBports Foundation. (), [Online]. Available: https://ubuntu-touch.io/fr/-3 (visited on 05/08/2024).

[51] "postmarketOS // real linux distribution for phones," postmarketOS. (), [Online]. Available: https://postmarketos.org/ (visited on 05/08/2024).

[52] T. Pirson and D. Bol, "Assessing the embodied carbon footprint of iot edge devices with a bottom-up life-cycle approach," *Journal of Cleaner Production*, vol. 322, p. 128 966, 2021. DOI: 10.1016/j.jclepro.2021.128966. [Online]. Available: https://doi.org/10.1016/j.jclepro.2021.128966.

[53] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. USA: Princeton University Press, 2012, ISBN: 0691149216.

[54] S. P. Tseng, W.-L. Li, C.-Y. Sheng, J.-W. Hsu, and C.-S. Chen, "Motion and attitude estimation using inertial measurements with complementary filter," in *Proceedings of the 8th Asian Control Conference (ASCC)*, Kaohsiung, Taiwan: IEEE, May 2011, pp. 863–868. DOI: 10.1109/ASCC.2011.5975261. [Online]. Available: https://ieeexplore.ieee.org/document/5975261.

[55] A. Noordin, M. A. M. Basri, and Z. Mohamed, "Sensor fusion algorithm by complementary filter for attitude estimation of quadrotor with low-cost imu," *TELKOMNIKA*, vol. 16, no. 2, pp. 868–875, Apr. 2018, ISSN: 1693-6930. DOI: 10.12928/TELKOMNIKA.v16i2.9020.

[56] "Complementary filter and relative orientation with MPU6050," HiBit. (Feb. 20, 2023), [Online]. Available: https://www.hibit.dev/posts/92/complementary-filter-and-relative-orientation-with-mpu6050 (visited on 07/31/2024).

[57] "Kalman filter - wikipedia." (), [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter (visited on 07/30/2024).

[58] "Learn more about magnetometer models and HSI calibration · VectorNav." (), [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-hsicalibration (visited on 05/12/2024).

[59] C. Z. Roman Szewczyk and M. Kaliczyńska, *Recent Advances in Automation, Robotics and Measuring Techniques* (Advances in Intelligent Systems and Computing). Cham, Heidelberg, New York, Dordrecht, London: Springer International Publishing, 2014, vol. 267, ISBN: 978-3-319-05352-3. DOI: 10.1007/978-3-319-05353-0. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-05353-0.

[60] K. Patel and J. Barve, "Modeling, simulation and control study for the quad-copter uav," *International Journal of Dynamics and Control*, vol. 11, pp. 689–700, 2023. DOI: 10.1007/s40435-022-01016-1.

[61] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Thèse No. 3727 (2007), Doctoral Thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2007. [Online]. Available: https://infoscience.epfl.ch/record/82850.

[62] P. Hemanth *et al.*, *Fabrication Of Quadcopter with Face Recognition for Surveillance*. Aug. 16, 2022. DOI: 10.46254/IN02.20220633.

[63] I. Sa and P. Corke, "System identification, estimation and control for a cost effective open-source quadcopter," in *2012 IEEE International Conference on Robotics and Automation*, ISSN: 1050-4729, May 2012, pp. 2202–2209. DOI: 10.1109/ICRA.2012.6224896. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6224896?casa_token=3IUHQxtHElkAAAAA:TLZlfF_cUSM4fd4hXzFIrZAzA7scdin-YM7fhPcgEfYB8Zk6c4CI9tkAo8gJQR3t9535oxHFlAhyxi (visited on 07/30/2024).

[64] Z. Huang, R. Bauer, and Y.-J. Pan, "Closed-loop identification and real-time control of a micro quadcopter," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 3, pp. 2855–2863, Mar. 2022, Conference Name: IEEE Transactions on Industrial Electronics, ISSN: 1557-9948. DOI: 10.1109/TIE.2021.3065613. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9380974?casa_token=x4fV6WxUXjQAAAAA:yi52ljoStEw1ZjAbss2g-VG1R8h8eTXNqyN8AY5XNrbVuezKkso (visited on 07/30/2024).

[65] K. Patel and J. Barve, "Modeling, simulation and control study for the quad-copter uav," *IEEE Xplore*, 2024, Downloaded on April 11, 2024, from IEEE Xplore.

[66] "Arduino nano," Arduino Official Store. (), [Online]. Available: https://store.arduino.cc/products/arduino-nano (visited on 08/09/2024).

[67] H.-Q.-T. Ngo, T.-P. Nguyen, V.-N.-S. Huynh, T.-S. Le, and C.-T. Nguyen, "Experimental comparison of complementary filter and kalman filter design for low-cost sensor in quadcopter," in *2017 International Conference on System Science and Engineering (ICSSE)*, ISSN: 2325-0925, Jul. 2017, pp. 488–493. DOI: 10.1109/ICSSE.2017.8030922. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8030922?casa_token=qb1SLOzT3KUAAAAA:cnpyeWQ7DwNtLYg1ZGddOAwMxx_czUJmXx4fMj-NhkXb68ouPLE8e5JWqBc4dNnQUaEvDbdCGOIm4w (visited on 08/09/2024).

[68] Z. Wu, M. Yao, H. Ma, and W. Jia, "Improving accuracy of the vehicle attitude estimation for low-cost INS/GPS integration aided by the GPS-measured course angle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 553–564, Jun. 2013, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2012.2224343. [Online]. Available: https://ieeexplore.

ieee.org/abstract/document/6340344?casa_token=lzbluF5kv7QAAAAA:2Eg1JvLr7kzQraBjiOKIduuECiGZUSktS4D40XcjBNEVx1qrrI_4W-_-GFZYXaxaPFX7ZMamj-ZHdQ (visited on 08/09/2024).

[69] M. Mirtaba, M. Jeddi, A. Nikoofard, and Z. Shirmohammadi, "Design and implementation of a low-complexity flight controller for a quadrotor uav," *International Journal of Dynamics and Control*, vol. 11, pp. 689–700, 2023. DOI: 10.1007/s40435-022-01016-1.

[70] A. T. Bayisa, "Controlling quadcopter altitude using pid-control system," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 12, pp. 195–199, 2019. DOI: 10.17577/IJERTV8IS120118. [Online]. Available: http://www.ijert.org.

[71] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," Master's thesis, Lund University, Lund, Sweden, 2008. [Online]. Available: http://www.control.lth.se/publications/.

[72] B. Sikdar, "A study of the environmental impact of wired and wireless local area network access," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 85–92, Feb. 2013, Conference Name: IEEE Transactions on Consumer Electronics, ISSN: 1558-4127. DOI: 10.1109/TCE.2013.6490245. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6490245?casa_token=BRT4WMccE_sAAAAA:Tq6_0bNRfHFciiNnXNBlfAwEun59Gqcdyja7ILaPeNKzTxuw0d (visited on 07/30/2024).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**