

Universidad
Rey Juan Carlos

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

CURSO ACADÉMICO 2014/2015

PROYECTO DE FIN DE CARRERA

An UAV prototype

Autor: Livio Calvo Ramos

Tutor: Jose María Cañas Plaza

Acknowledges

I wish to thank Jose María Cañas Plaza for his guidance in the development of this project, to Oscar García for his advice on electronics and aeromodelism, and to María Teresa González de Lena, José Centeno González, Julián Nora and many other teachers whose teachings were especially helpful to me.

Summary

The Unmanned Aerial Vehicles (UAVs), also known as "drones", are flying robots that can be controlled remotely by a human operator or can function completely autonomous. They are a booming technology and we are now seeing them in the news, movies and video games. They can patrol borders, monitor power lines, transport light payloads and even fight in war scenarios.

Implementing a real UAV can be very expensive and time consuming. This final degree project has been oriented to implement an UAV prototype with cheap hardware components to simulate the basic operations of a real UAV like obtaining information from hardware sensors, calculating the flying route and sending the corresponding electric currents to the motors. A tele-operation mode has also been implemented through a web interface to allow a human operator to monitor and control the UAV prototype remotely.

Finally, several tests have been run to check that all the simulated capabilities of the UAV prototype worked as expected. All of them were successfully completed.

Contents

1	Introduction	5
1.1	Unmanned Aerial Vehicles	5
1.1.1	Aeronautical Design	5
1.1.2	Sensors and Actuators	6
1.1.3	Radio-controlled aircraft	9
1.2	UAV applications	9
1.2.1	Surveillance and Reconnaissance	9
1.2.2	Combat Operations	16
1.3	UAVs in Spain	17
1.3.1	Universidad de Sevilla	17
1.3.2	Alfa Bravo Servicios Aeronáuticos	18
1.3.3	Atlante	18
1.3.4	Universidad Politécnica de Madrid (UPM)	20
1.3.5	FUVE: Future Vehicles and Entrepreneurs	21
2	Objectives	23
2.1	Description of the problem	23
2.2	Requirements	23
2.3	Development Process	24
3	Infrastructure	26
3.1	Android Mobile	26
3.2	Raspberry Pi	28
3.3	Arduino	30
3.4	Web Interface Technologies	32
3.4.1	Apache	32
3.4.2	MySQL	33
3.4.3	PHP	33

3.4.4	Motion	34
3.5	Flying platform	34
3.5.1	DC Motors	34
3.5.2	Servos	35
3.5.3	Propellers	36
3.5.4	Phantom FPV Flying Wing	37
4	UAV System Description	39
4.1	UAV Design	39
4.2	Navigation	45
4.2.1	Android application	46
4.2.2	Java application	53
4.2.3	Arduino application	58
4.3	Web Interface	59
4.3.1	UAV Manager web application	60
4.3.2	Apache Web Server	63
4.3.3	Motion	64
4.3.4	MySQL server	64
4.4	Sensors-Actuators Infrastructure	64
4.4.1	Sensors-to-Navigation	66
4.4.2	Navigation-to-Motors	66
4.4.3	Web Interface-Motors	68
5	Experiments	70
5.1	UAV test board assembly	70
5.2	UAV platform teleoperation	71
5.3	Assembly of the "Phantom" RC airplane	79
6	Conclusions	81
6.1	Conclusions	81
6.2	Future developments	82

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV), commonly known as "drone", is an aircraft without a human pilot on board. It's controlled by computers on board or by a human operator through a remote control. They are usually deployed in missions that are too simple or dangerous for a human pilot.

The use of drones has grown quickly in recent years because unlike manned aircrafts they can stay aloft for many hours (a British drone called Zephyr holds the official endurance record, flying more than 336 hours). They are much cheaper than conventional aircrafts and they are controlled remotely so there is no danger to the flight crew.

1.1 Unmanned Aerial Vehicles

1.1.1 Aeronautical Design

The UAVs can have several shapes and sizes. They can fly by gaining support from the atmosphere. This support counters the force of gravity by using either static lift, as airships do, or by using the dynamic lift of an airfoil, like airplanes.

The human activity that surrounds aircraft is called aviation. Crewed aircrafts are flown by an onboard pilot, but UAVs may be remotely controlled or self-controlled by onboard computers. Aircrafts may be classified by different criteria, such as lift type, propulsion, usage and others. They can be as small as radiocontrol planes or as big as an Airbus 380. A general classification follows:

- Airplanes: they are powered fixed-wing aircrafts (as seen in figure 1.1) that are propelled forward by thrust from a jet engine or propeller. If the airplane has no engines it's called a glider and it's supported in flight only by the dynamic reaction of the air against its lifting surfaces.



Figure 1.1: IAI Heron airplane

- VTOLs (Vertical Take-Off and Landing): they are powered aircrafts that can hover, take off, and land vertically. All helicopters are VTOLs, like other aircrafts such as the Harrier, the V-22 Osprey or the Parrot AR.Drone.
- Airships: they are lighter-than-air aircrafts (as seen in figure 1.2) that can be steered and propelled through the air using rudders and propellers or other thrust mechanisms. Airships stay aloft by having a large "envelope" filled with a lifting gas that is less dense than the surrounding air.

1.1.2 Sensors and Actuators

The UAVs, as other robots, need sensors to perceive the surrounding world and actuators to interact with their environment. Usually UAVs have these technologies onboard:

- Camera: it allows the human operator to get a video feed, so he can see where the UAV is flying. It allows to take pictures and to record video.



Figure 1.2: Airship

In some cases, the UAV can analyze the video to assist in navigation, pursue targets and other tasks.

- Magnetic sensors: they sense earth magnetic field and can be used as a compass so they provide the heading of the UAV. They can be used to get pitch and roll values too.
- Altimeter: it measures the atmospheric pressure. When the altitude is higher, the pressure is lower. Using this rule the altimeter can calculate the altitude of the UAV.
- GPS receiver: the Global Positioning System (GPS) is composed of several satellites in geostationary orbit around the Earth. Each satellite continually transmits electromagnetic signals with information about the time the message was transmitted and the satellite position at the time of message transmission. With this information the GPS receiver calculates the 3D position of the UAV.
- Inertial Measurement Unit (IMU): it measures the aircraft velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. The data collected from the IMU's sensors allow a computer to track the aircraft's position, using a method known as dead reckoning. An IMU works even when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interferences are present.

- **Flight Control Surfaces:** the moving parts attached to the airframe of the aircraft (as seen in figure 1.3). The flight control surfaces can deflect the air stream passing over them. This redirection of the air stream generates an unbalanced force to rotate the aircraft about the associated axis, so the pilot (or computer) can control the aircraft's flight attitude.

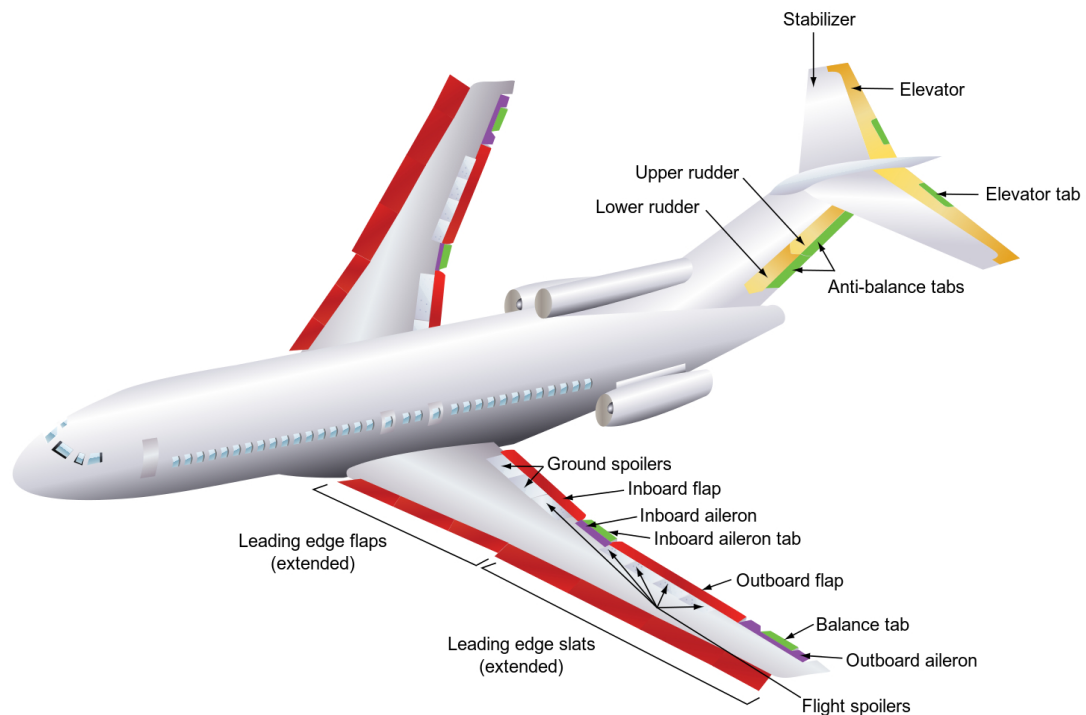


Figure 1.3: Flight control surfaces of an airplane

- **Motors:** they are machines designed to convert energy into useful mechanical motion. They are used when circular motions are required, for example, to move a propeller. Aircrafts can use heat motors or electric motors: Heat motors burn a fuel to create heat, which then creates motion, while electric motors convert electrical energy into mechanical motion. Electric DC motors are powered from direct current and they are the most used in small aircrafts because they use rechargeable bat-

teries instead of fuel and they are safer and cheaper than heat motors. Heat engines provide much more energy than electric ones, so big aircrafts still use this technology. Aircrafts use motors to provide forward and/or upward thrust.

- Servos: they are actuators which move or control a mechanism or system. They are operated by electric current and convert that energy into motion. Servos are used to control the aircraft, doing tasks like move the control surfaces or down the landing gear.

1.1.3 Radio-controlled aircraft

A radio-controlled aircraft (RC aircraft) is a small flying machine that is controlled remotely by an operator on the ground using a radio transmitter. The flying machine can be an airplane (as seen in figure 1.4), an helicopter, a multicopter or even an airship. The transmitter communicates with a receiver within the flying machine that sends signals to servomechanisms which move the control surfaces and change the motor speed based on the position of joysticks on the transmitter. In this way the flying machine can be flied remotely, so a radio-controlled aircraft is an UAV that only has a manual control mode.

1.2 UAV applications

UAVs are predominantly used in military applications but also in a growing number of civil applications, which basically fall into two categories:

- Surveillance and Reconnaissance: the UAV gets any type of information of a designed flying route using its internal sensors
- Combat Operations: the UAV is armed with missiles and/or bombs, and uses its internal sensors to locate and destroy enemy targets

1.2.1 Surveillance and Reconnaissance

UAVs can be used for surveillance operations, which raise significant issues for privacy. Surveillance drones are already in use by law enforcement and can carry various types of equipment including live-feed video cameras, infrared



Figure 1.4: RC airplane

cameras, heat sensors, and radar. They can even carry wifi crackers and fake cell phone towers that can determine your location or intercept your texts and phone calls[1].

A fleet of miniature helicopter drones (as seen in figure 1.5) mounted with thermal imaging cameras will be deployed to combat graffiti-spraying gangs on the german railway network [2]. Each drone will be able to record videos which will be used as evidence in trials. These drones are manufactured by german firm Microdrones. The use of drones against vandals is the latest indication of the growing civilian market for unmanned aerial reconnaissance. Over 400 new drone systems are being developed by firms based in Europe.

UK police is using the "spy drone" [3], fitted with CCTV cameras, mainly for tackling anti-social behaviour and public disorder, but also for monitoring traffic congestions. These drones are manufactured by Microdrones (UK). The machines, which are flown by remote control or using pre-programmed GPS navigation systems, are silent and can be fitted with night-vision cameras (as seen in figure 1.6). The images they record are sent back to a police support vehicle or control room.

UAVs like "InView Unmanned Aircraft System" (as seen in figure 1.7)



Figure 1.5: Deutsche Bahn drones

can be used to perform aeromagnetic surveys[4] where the processed measurements of the Earth's magnetic field strength are used to calculate the nature of the underlying magnetic rock structure, which helps trained geophysicists to predict the location of mineral deposits. This drone is manufactured by Barnard Microsystems in the UK.

The "Air Mule" UAV (as seen in figure 1.8) is being designed for cargo transport, medical evacuation and troop supply missions[5]. It can be operated in remote areas where helicopters and traditional rotorcraft cannot function. It's scheduled to enter service in 2014 and it's being developed by Urban Aeronautics in Israel.

The U.S. Customs and Border Protection is using a computer system called VADER[6] (Vehicle Dismount and Exploitation Radar) developed by Northrop Grumman and operated from Predator drones (as seen in figure 1.9) which patrol Mexico border to locate illegal immigrants and smugglers. This system was developed to track Taliban fighters in Afghanistan.

French electricity distribution network operator ERDF and Cassidian[7] are developing an UAV (as seen in figure 1.10) to inspect high voltage power lines. This drone is an helicopter which can fly in unfavourable weather



Figure 1.6: UK Police Drone



Figure 1.7: InView Unmanned Aircraft System



Figure 1.8: AirMule



Figure 1.9: Predator drone used in border control

conditions, carry large numbers of various types of sensors and provide data post-processing.



Figure 1.10: ERDF/Cassidian drone for power lines inspection

Amazon, the world's largest online retailer, is testing unmanned drones to deliver goods to customers[8]. The drones (as seen in figure 1.11) could deliver packages weighing up to 2.3 kg to customers within 30 minutes of flying time from drone base to customer home. This service is called Amazon Prime Air and could take up to five years to start. The US Federal Aviation Administration is yet to approve the use of unmanned drones for civilian purposes.

Brazilian police have bought israeli surveillance drones (as seen in figure 1.12) to monitor the areas around the soccer stadiums during the 2014 World Cup soccer tournament[9]. There are thousands of fans from across the world expected to attend this event, and this and other measures are expected to increase street security.

Parrot AR.Drone is a radio controlled quadcopter (as seen in figure 1.13) built by the French company Parrot. The drone is designed to be controlled by mobile or tablet operating systems such as Android or iOS, but it can be controlled from other operating systems through unofficial software. It takes off automatically, hovers by itself and is very easy to fly thanks to its autopilot and sensors. It also incorporates a 720p camera located in the nose.



Figure 1.11: Amazon drone



Figure 1.12: Brazilian Police drone

Organizations from around the world such as the Stanford Robotics Club or Students at NASA Program are now using the drone[10].



Figure 1.13: Parrot AR.Drone

1.2.2 Combat Operations

While the military drones are physically in the war zone (i.e.: Afghanistan or Iraq), they are usually controlled via satellite from the country to which they belong. Ground crews launch drones from the conflict zone, then operation is handed over to controllers at video screens in specially designed facilities in their country. One person flies the drone, another operates and monitors the cameras and sensors, while a third person is in contact with ground troops and commanders in the war zone.

Some military drones are very popular today as they have appeared in the news, movies and video games. Such is the case of the Predator (as seen in figure 1.14): initially conceived for reconnaissance and forward observation roles, it carries cameras and other sensors but has been modified and upgraded to carry and fire two AGM-114 Hellfire missiles or other munitions. The drone, in use since 1995, has seen combat over Afghanistan, Pakistan, Bosnia, Serbia, Iraq, Yemen, Libya, and Somalia.

Following 2001, the Predator became the primary unmanned aircraft used for offensive operations by the USAF and the CIA in Afghanistan and the Pakistani tribal areas; it has also been deployed elsewhere. Civilian applications have included border enforcement and scientific studies.



Figure 1.14: Predator

1.3 UAVs in Spain

There are several spanish universities and firms developing and manufacturing UAVs. The following list describes the main projects:

1.3.1 Universidad de Sevilla

They are working in several UAV related projects like MUAC-IREN[11] (Multi-UAV Cooperation for long endurance applications). This project is trying to develop long endurance UAVs (as seen in figure 1.15) with a very high range¹. They are exploring new ideas like autonomous soaring exploiting favourable wind conditions to extend flight duration. UAVs which are

¹Range is the distance an aircraft can fly between takeoff and landing, as limited by fuel or battery capacity in powered aircraft, or environmental conditions in unpowered aircraft

able to fly without need to land for recharge batteries or fill fuel tanks can be very useful for many applications such as surveillance, traffic control, coast control, etc. They are also exploring ways to integrate weather estimation and control algorithms that enable UAVs to fly and perform their mission successfully even if very adverse weather conditions are encountered.



Figure 1.15: MUAC-IREN aerial robot

1.3.2 Alfa Bravo Servicios Aeronáuticos

Alfa Bravo Servicios Aeronáuticos SL is a young firm, founded in 2008, focused in the production of electronic systems and unmanned robots for Spanish Armed Forces, including military UAVs. In 2009 they signed a contract with Spanish Armed Forces[12] for 1,96 million euros. The UAV model was not disclosed but the internet portal "El Confidencial Digital" assured that the model chosen by Spanish Armed Forces was a RQ-11 Raven (as seen in figure 1.16), supplied by the american firm AeroVironment. The RQ-11 Raven is a small hand-launched unmanned aerial vehicle developed for the U.S. military, but now adopted by the military forces of many other countries. The Raven can be either remotely controlled from the ground station or fly autonomous missions using GPS waypoint navigation.

1.3.3 Atlante

Atlante (as seen in figure 1.17)[13] is a tactical long-endurance unmanned air vehicle (UAV) system designed to perform ISTAR² operations. Other operations carried out by the UAV include identifying targets, day and night

²Intelligence, Surveillance, Target Acquisition and Reconnaissance



Figure 1.16: RQ-11 Raven



Figure 1.17: Atlante

surveillance, over-the-hill reconnaissance, battle-damage assessment, troop and convoy protection, border surveillance, and search and rescue.

The UAV will also meet the requirements of Guardia Civil and other Spanish emergency agencies. The UAV system is manufactured by EADS CASA and made its first flight in February 28th, 2013 in Lugo (Spain).

Atlante offers fully automated take-off and landing capabilities, even in adverse weather conditions. It uses a pneumatic catapult to take off and is recovered with parachutes or nets. It has a simple landing gear to take off and land in conventional airstrips.

1.3.4 Universidad Politécnica de Madrid (UPM)



Figure 1.18: UPM autonomous drone flying indoors

A team of five members of the UPM research group "Visión por Computador del Centro de Automática y Robótica" has won the first position in the IMAV competition 2013. The category of this award was Indoor

Autonomy[14], their best innovations were UAV autonomy and simultaneous coordination of several UAVs. The Indoor Autonomy category requires the UAV to fly autonomously in an environment only partially known and without use any GPS sensor. The research group presented a multi-robot autonomous system using as platform the Parrot AR Drone 2.0 drone (as seen in figure 1.18). Each drone communicates with a computer via wifi in which the navigation algorithms run, and each computer communicates with other computers through a LAN network.

1.3.5 FUVe: Future Vehicles and Entrepreneurs

FUVe[15] is a team of students from different universities and various disciplines such as marine engineering, aerospace, electronics, software and telecommunications. They are working together in Politechnical University of Madrid (UPM) Naval School to design and construct remotely operated underwater vehicles (ROVs) and Unmanned Aerial Vehicles. Their UAV is an Autogyro (as seen in figure 1.19) which was created by Spanish engineer Juan de la Cierva in 1923. This type of flying machine uses an unpowered rotor in autorotation to develop lift, and an engine-powered propeller to provide thrust.



Figure 1.19: FUVe Autogyro

The objective of this project is the design of a basic UAV prototype, including onboard computer, sensors, actuators, and a user interface for remote control.

The rest of the document is organized in 5 chapters. The second chapter describes the objectives of the project. The third chapter offers a description of the hardware and software technologies that have been used to develop the UAV. The fourth chapter explains the implementation of our UAV system. The fifth chapter describes the experiments made to test the prototype. Finally the sixth chapter explains the conclusions of the project and proposes some ideas for future developments.

Chapter 2

Objectives

The purpose of this work is to build a prototype simulating the basic components of an Unmanned Aerial Vehicle.

2.1 Description of the problem

We have divided the global goal into several subgoals:

- Design of an UAV prototype including hardware and software architecture
- Development of an API¹ to read the sensors and control the actuators of the UAV prototype
- Development of a web interface to monitor and control the UAV prototype remotely

2.2 Requirements

The UAV system should have the following features:

- It must have an onboard computer to run the software that controls the system

¹An Application Programming Interface (API) is a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.

- Free software tools: The UAV must use only free software tools which are developed under terms that guarantee the users freedom to run it, adapt it to their needs, and redistribute it with or without changes
- License: The source code of the project applications are free software, under GPLv3 License ²
- The onboard computer must be connected to sensors to detect signals from the UAV environment
- The UAV must have actuators to control flight control surfaces and/or motors
- Real time: The UAV hardware and software must be fast enough to control an hypothetical airship like flight
- Robustness: The UAV hardware and software never should freeze or behave in an unsafe way

2.3 Development Process

The development model has been "Code and fix" ³. Due to this is a small project and there was no previous knowledge about aeromodeling in the robotics group at URJC, the system was developed from scratch without following any previous design.

During the course of the project several stages had been established so different parts of the development have been addressed progressively, always considering cost as a critical factor. The cost of the hardware has been more than 200€, and the cost of the software has been 0€ as only have been used free software tools.

They main stages have been:

- Selection of computing platform: Several options have been studied considering weight and computing capabilities. Once computing hardware was choosed, several software platforms have been studied to run on the selected hardware

²<http://www.gnu.org/licenses/gpl-3.0-standalone.html>

³http://en.wikipedia.org/wiki/Software_development_process#Code_and_fix

- Selection of electronic system: A set of battery, sensors, cables and actuators has been choosed to perform basic operations like signal detection and control of engines and/or surface controls
- Design and development of an autopilot: Several applications have been developed to implement an autopilot system that sense external signals, process them, and send commands to actuators simulating a real flight
- Design and development of user interface: To allow user interaction with the UAV a web user interface has been developed. Using any device with wifi and web browser the user can monitor the system status, set up a flying route and fly it easily only pressing buttons using an optional manual flight mode. A webcam has been added to the computing hardware whose video feed is showed in the web user interface

For more information, including videos, pictures and source code check project MediaWiki[16].

Chapter 3

Infrastructure

In this chapter the main hardware and software technologies used to build the UAV will be explained. They can be categorized in the following sections:

3.1 Android Mobile

The Samsung Galaxy Mini (figure 3.1) is a smartphone manufactured by Samsung that runs the Android operating system. Its key features for this project are GPS receiver and WiFi connectivity. It has been used as navigation computer. A summary of the most important features follows:

- Quad-Band GSM and dual-band 3G support
- 7.2 Mbit/s HSDPA
- WiFi 802.11 (b/g/n)
- Bluetooth technology v 2.1
- USB 2.0 (High Speed)
- 3.14 in (80 mm) 256K-color QVGA TFT touchscreen
- ARMv6 600 MHz processor, 384 MB RAM (only 279 MB RAM available)
- Adreno 200 GPU



Figure 3.1: Samsung Galaxy Mini

- Android OS v2.2 (Froyo), upgrade to v2.3.6 (Gingerbread) available in some places.
- 160 MB internal storage, hot-swappable MicroSD slot, 2 GB card included
- 3.15 Mpixel fixed-focus camera with geo-tagging
- 3.5 mm audio jack
- Accelerometer and proximity sensor
- Swype virtual keyboard
- MicroUSB port (charging and data transfer) and stereo Bluetooth 2.1

- Magnetic sensors: They sense earth magnetic field. This allows to use this sensors as a compass to get the heading, and as a gyroscope to get roll, pitch and yaw values
- GPS receiver with A-GPS: This device allows to know the gps coordinates of the device

The mobile operating system is Android. It's Linux-based and designed primarily for touchscreen mobile devices such as smartphones and tablet computers. The Android programming language is a customized version of Java.

Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers.

Android's open nature has further encouraged a large community of developers and enthusiasts to use the open source code as a foundation for community-driven projects.

3.2 Raspberry Pi

The Raspberry Pi (figure 3.2) is a single-board computer developed in the UK by the Raspberry Pi Foundation. It's a low cost device that can be purchased for only 25\$. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage. It has been used as a computing node to host the user interface and to allow communication between Android Mobile and Arduino.

The more important features are:

- SoC: Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and single USB port)
- CPU: 700 MHz ARM1176JZF-S core (ARM11 family)
- GPU: Broadcom VideoCore IV
- Memory (SDRAM): 512 MB (shared with GPU)
- USB 2.0 ports: 2

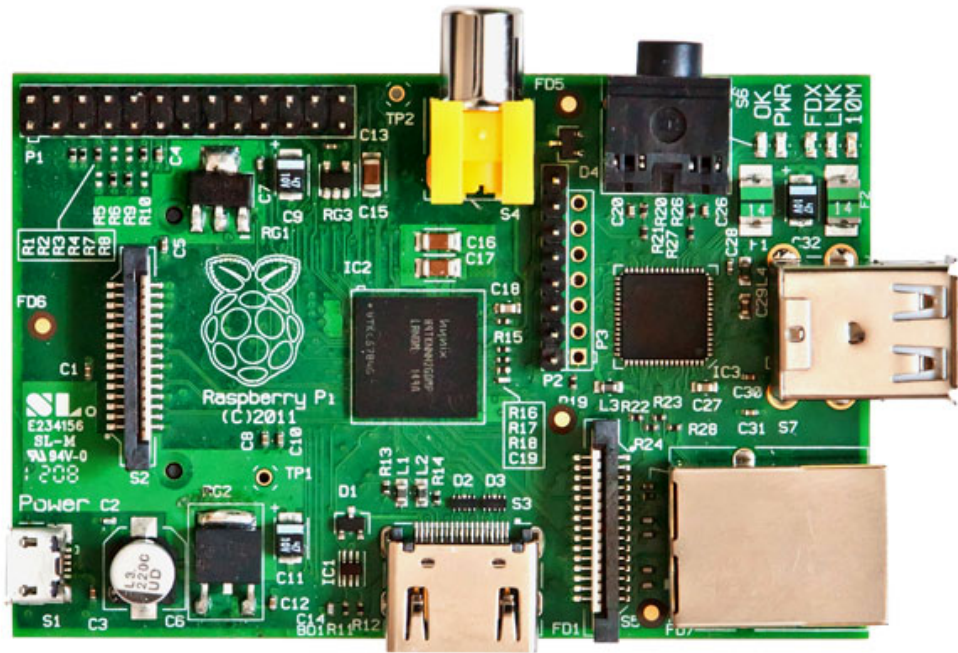


Figure 3.2: Raspberry Pi

- Video Outputs: Composite RCA (PAL and NTSC), HDMI (rev 1.3 & 1.4), raw LCD Panels via DSI
- Audio outputs: 3.5 mm jack, HDMI
- Onboard storage: SD / MMC / SDIO card slot
- Onboard network: 10/100 Ethernet (RJ45)
- Power ratings: 700 mA (3.5 W)
- Power source: 5 volt via MicroUSB or GPIO header
- Operating Systems: Debian GNU/Linux, Fedora, Arch Linux ARM, RISC OS

Raspbian is the operating system running on Raspberry. It's a Linux variant based on the ARM hard-float (armhf)-Debian 7 'Wheezy' architecture port but optimized for the ARMv6 instruction set of the Raspberry Pi

hardware. The Raspbian Linux distribution is free software and basically contains the LXDE desktop environment, the Openbox window manager, the Midori web browser and software development tools.

The programs that have been developed on this project for running on Raspberry are Java applications. Java is a general-purpose, concurrent, class-based, object-oriented computer programming language. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is one of the most popular programming languages in use, particularly for client-server web applications.

3.3 Arduino

The Arduino is a microcontroller board programmed with a language similar to C++. It has connectors exposed in a standard way, allowing the main board be connected to a variety of add-on modules known as "Shields". It's a low cost computing node that can be use to control electronic systems like servos, motors, leds, temperature sensors, etc.

There are different models of Arduino, I have used the Arduino Uno (figure 3.3) connected to the official Arduino Motor Shield (figure 3.4) to control the UAV motors. The cost of both boards was approximately 60€.

The more important features are:

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader

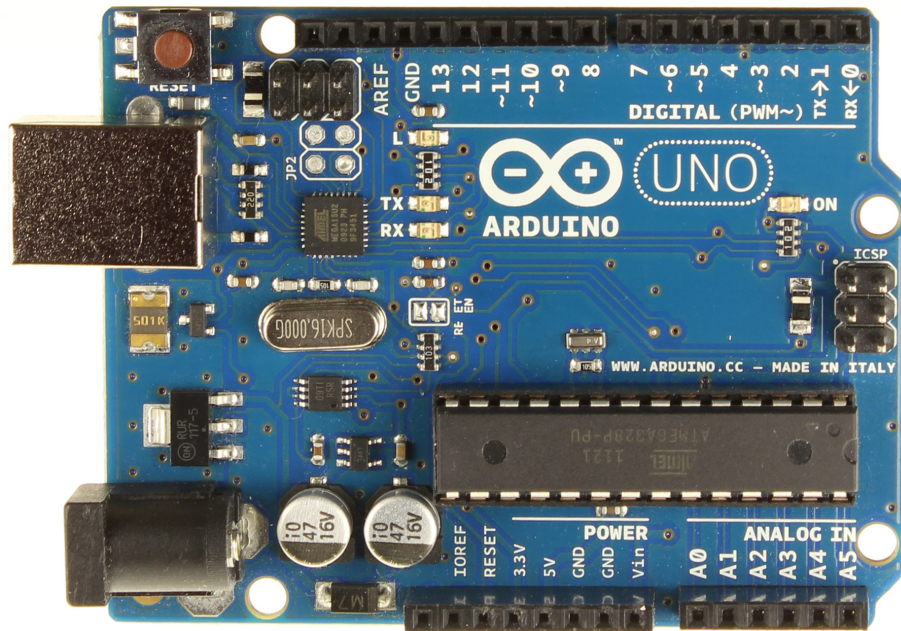


Figure 3.3: Arduino Uno

- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

The more important features are:

- Operating Voltage: 5V to 12V
- Motor controller: L298P, Drives 2 DC motors or 1 stepper motor
- Max current: 2A per channel or 4A max (with external power supply)
- Free running stop and brake function

The Arduino integrated development environment (IDE) is a cross-platform application written in Java. Arduino programs are written in C or C++. The Arduino IDE comes with a software library which makes many common input/output operations much easier. Users only need to define two functions to make a runnable cyclic executive program:

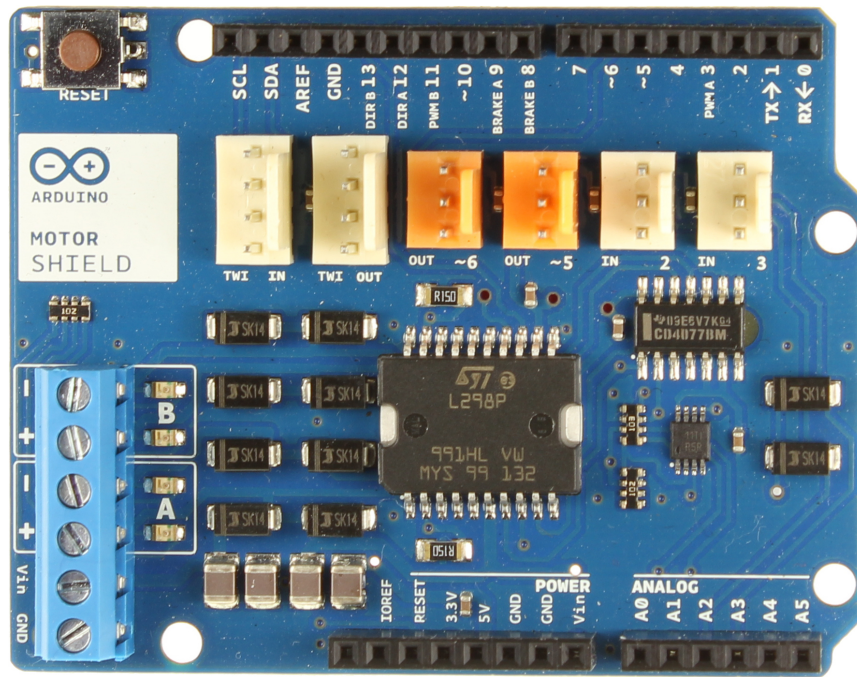


Figure 3.4: Arduino Motor Shield

- `setup()`: It runs once at the start of the program to initialize settings
- `loop()`: It's called repeatedly until the board powers off

The most of Arduino boards have a LED and a load resistor connected between pin 13 and ground, a convenient feature for many simple tests.

3.4 Web Interface Technologies

3.4.1 Apache

The Apache HTTP Server, commonly referred to as Apache, is a web server software program notable for playing a key role in the initial growth of the World Wide Web. In 2009, it became the first web server software to surpass the 100 million website milestone. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently named Oracle iPlanet Web Server). Typically Apache is run on a Unix-like operating

system, and was developed for use on Linux. In this project Apache has been used to host the user interface.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is open-source software. Since April 1996 Apache has been the most popular HTTP server software in use.

3.4.2 MySQL

MySQL is the world's most widely used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. It's a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python. Free-software-open source projects that require a full-featured database management system often use MySQL. In this project MySQL has been used to store navigation data.

Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube.

3.4.3 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. In this project PHP has been used to develop the user interface.

PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file

to process data. PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL). PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform, free of charge.

3.4.4 Motion

Motion, a software motion detector, is a free, open source CCTV software application developed for Linux. It can monitor video signal from one or more cameras and is able to detect if a significant part of the picture has changed saving away video when it detects that motion is occurring. In this project Motion has been used to capture pictures from the onboard camera, which is a standard usb webcam.

The program is written in C and is made for the Linux operating system (exploiting video4linux interface). Motion is a command line based tool whose output can be either jpeg, netpbm files or mpeg video sequences. It is strictly command line driven and can run as a daemon with a small footprint and low CPU usage.

It is operated mainly via config files, though the end video streams can be viewed from a web browser. It can also call to user configurable "triggers" when certain events occur.

3.5 Flying platform

3.5.1 DC Motors

Two brushed DC motors (figure 3.5) are used to spin the propellers of the test board.

- Free-run speed at 6V: 11500 rpm
- Free-run current at 6V: 70 mA
- Stall current at 6V: 800 mA

One brushless DC motor (figure 3.6) is used to spin the propeller of the Phantom Flying Wing. No specifications were provided by the vendor.



Figure 3.5: Brushed DC Motor



Figure 3.6: 900kv Brushless Outrunner Motor

3.5.2 Servos

Two servos (figure 3.7) are used to move flight control surfaces of the Phantom Flying Wing.



Figure 3.7: Standard Hitec Servo

- Operating Voltage: 4.8-6.0 Volts
- Operating Temperature Range: -20 to +60 Degree C
- Current Drain (4.8V): 5.4mA/idle and 150mA no load operating
- Current Drain (6.0V): 5.5mA/idle and 180mA no load operating
- Dimensions: 0.89" x 0.45" x 0.94" (22.8 x 11.6 x 24mm)
- Weight: 0.28oz (8g)

3.5.3 Propellers

The propellers (figure 3.8) move the air to provide propulsive force.

Two propellers like the first one (left) are used in the test board. They have the following specifications:

- Diameter: 4 inches

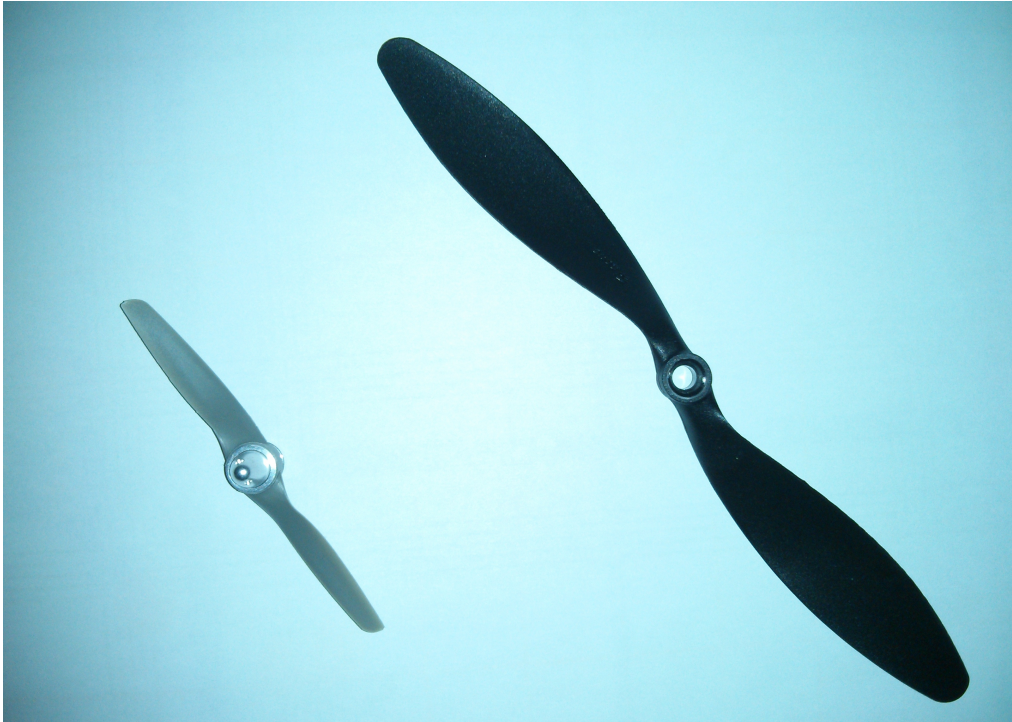


Figure 3.8: Propellers

- Blade pitch: No specified by the vendor

The second propeller (right) is the Phantom Flying Wing propeller and has the following specifications:

- Diameter: 9 inches
- Blade pitch: 6 inches

3.5.4 Phantom FPV Flying Wing

The Phantom is an aeromodeling airplane, it has been used as the flying platform, carrying onboard the rest of the components with the following features:

- Wingspan: 1550mm
- Length: 683mm



Figure 3.9: Phantom

- Flying Weight: 900g
- Made from EPO foam
- Works with 2 x 9 gram servos
- Works with 900kv Brushless Outrunner Motor

Chapter 4

UAV System Description

In this chapter the design and operation of the UAV prototype will be explained. The 4.1 "UAV Design" section provides an overview of the system. The 4.2 "Navigation" section describes how the UAV prototype manages to fly in autonomous and manual mode. The 4.3 "Web Interface" section describes how the human operator connects to the UAV prototype (displayed in figure 4.1) to monitor the system status and to control the flight. Finally the 4.4 "Sensors-Actuators Infrastructure" section describes the dataflow between the main components of the system.

The UAV software is open source and can be downloaded from the project MediaWiki[16] and svn repository[17].

4.1 UAV Design

The UAV prototype features the following hardware components:

- Mobile: It's a Samsung Galaxy Mini, which communicates with the Raspberry Pi using its integrated Wi-Fi capabilities. This smartphone also has the following integrated devices:
 - IMU: The mobile has integrated magnetic sensors which are used in this prototype as an Inertial Measurement Unit¹ which allow

¹An inertial measurement unit (IMU) is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes, sometimes also magnetometers

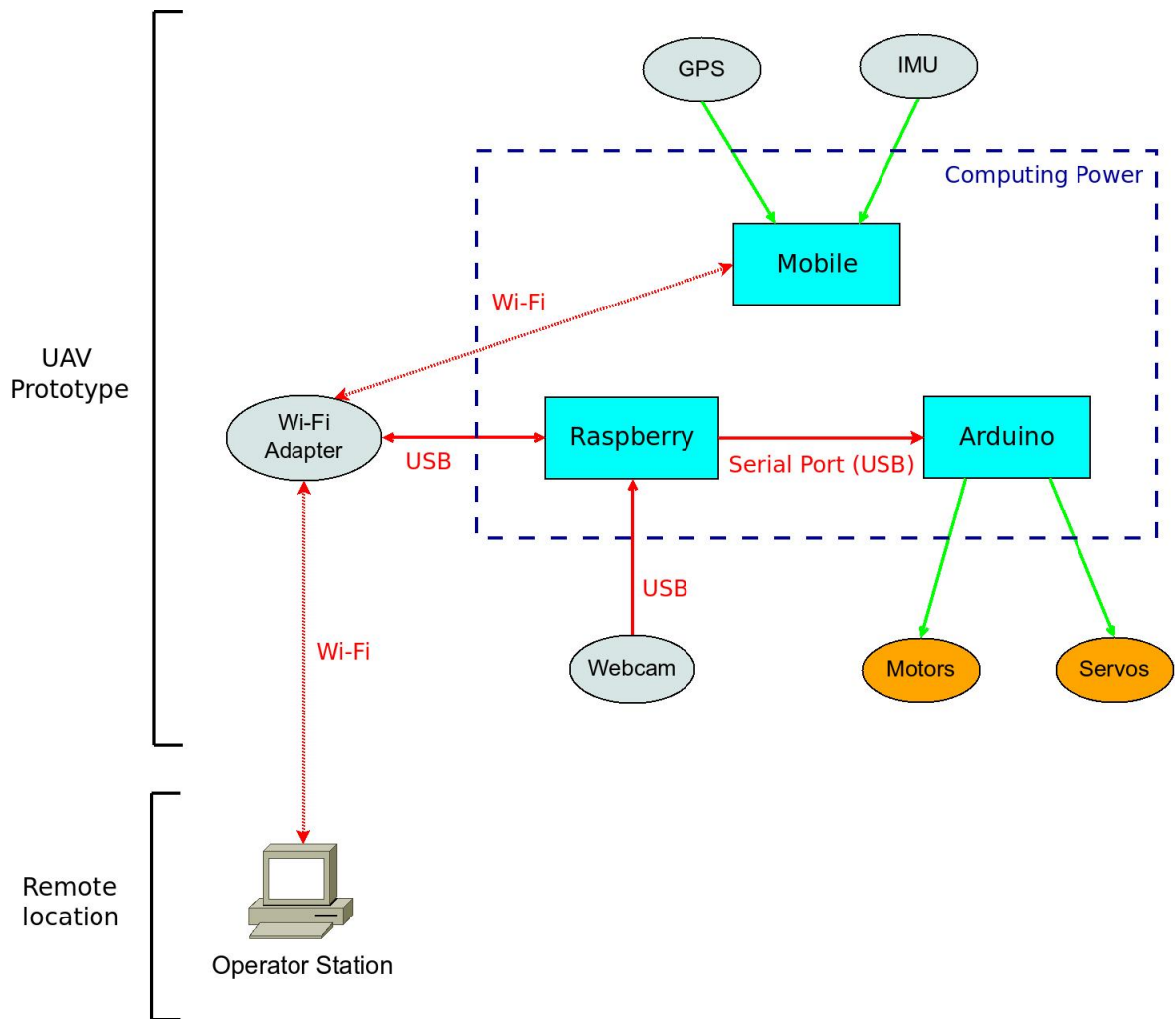


Figure 4.1: Hardware design

to know the device orientation, acceleration and inclination (roll, pitch and yaw axes).

- GPS: The mobile also has an integrated GPS sensor which is used to know the UAV current location using only the GPS satellite data. No A-GPS² or network provided location is used in this

²A-GPS (Assisted GPS) is a system that allows GPS receivers to obtain information from network resources to assist in satellite location. It is very useful in places with a poor reception of satellite radio signals.

prototype.

- Raspberry Pi: It's connected via USB with the following devices:
 - Wi-Fi USB adapter: it allows Wi-Fi connectivity with the mobile phone and the Operator Station.
 - Webcam: it's a standard webcam, which is used to generate a stream of pictures.
 - Arduino: it operates 2 DC motors according to the orders received from the Raspberry Pi through the USB connection.
- Arduino: It's connected via USB with the Raspberry Pi device. This USB connection encapsulates a virtual serial port, through which both devices communicate with each other. The Arduino Motor Shield is connected to two DC motors with propellers attached, which emulate the basic operations (forwards, backwards, yaw left and yaw right) of an airship. The Arduino Motor Shield was also connected to two servos to study how to manage the control surfaces of an airplane, but were not used in the final design.
- Operator Station: It's any device with web browser and Wi-Fi capabilities. There are no more requirements, as all the software runs inside the UAV prototype.

On this hardware works the UAV software which is subdivided in Navigation functionality, Web Interface functionality and Sensors-Actuators Infrastructure.

Navigation functionality

This functionality (as shown in figure 4.2) gets the UAV current position from the mobile device sensors, calculates how to go from the current location to the next waypoint of the flying route, and translates these calculations into motor movements. This is done by an Android application running on the mobile device, a Java application running on the Raspberry Pi and a C application running on the Arduino.

These applications perform the following tasks:

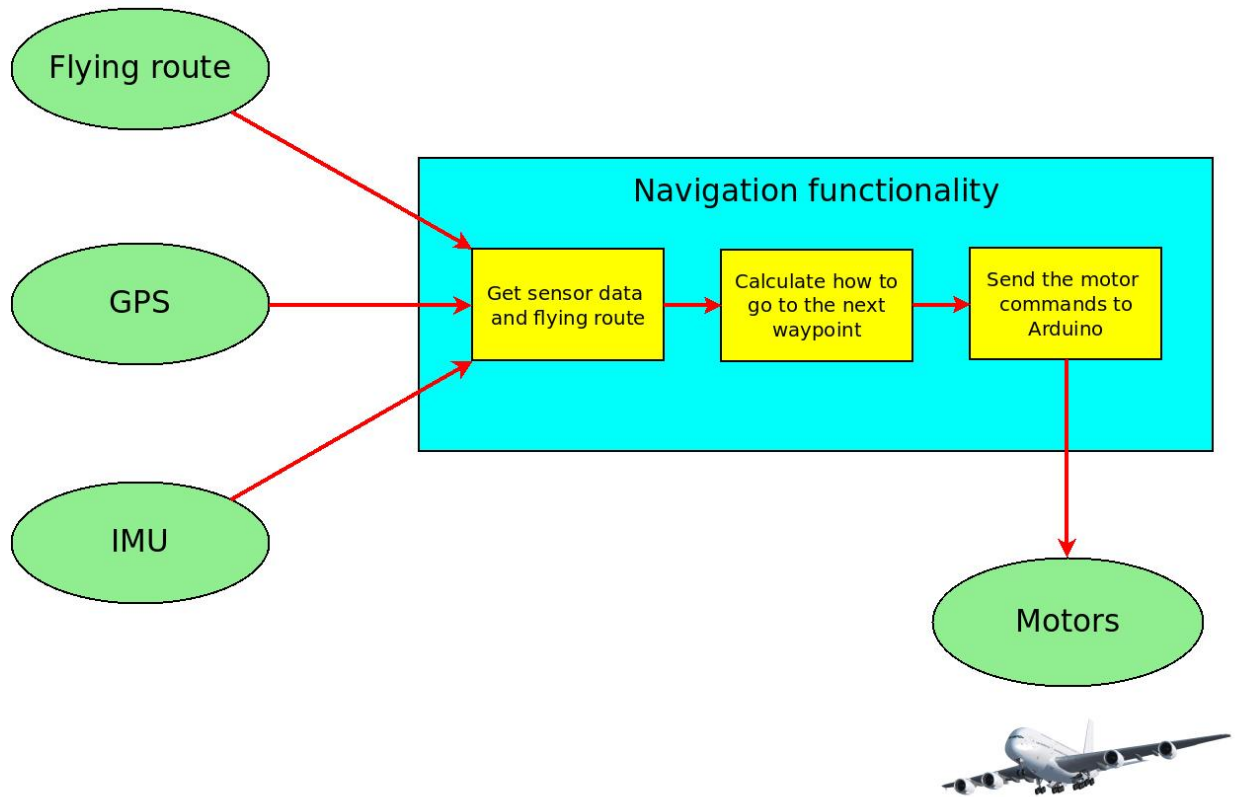


Figure 4.2: Navigation functionality

- Receive the flying route from the human operator which is composed of 4 waypoints
- Use the information provided by the mobile IMU (magnetic sensor) to get the current heading of the UAV
- Use the information provided by the mobile GPS sensor to get the current location and altitude of the UAV
- Calculate how to go from the current location to the next waypoint and send the motor commands to Arduino using a small protocol designed for this purpose.

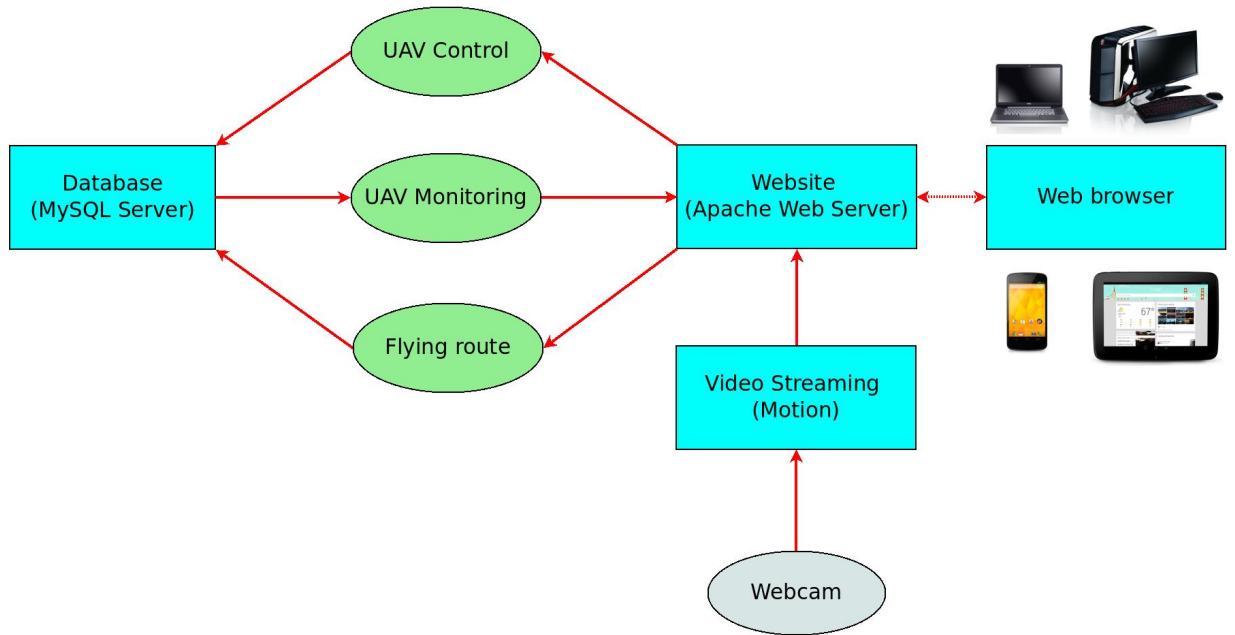


Figure 4.3: Web Interface functionality

Web Interface functionality

This functionality (as shown in figure 4.3) provides a web site which displays the UAV status, video streaming and control menus. The human operator can use this interface to monitor and control the UAV remotely using any device with web browser and Wi-Fi capabilities.

This software runs on the Raspberry Pi and is composed of an Apache web server hosting a PHP website, a MySQL server hosting a database and a Motion application hosting a video streaming obtained from the webcam connected to the Raspberry Pi. They work together to perform the following services:

- UAV Monitoring: The human operator can use one of the PHP web pages to monitor the UAV status. The status information is obtained from the database hosted by the MySQL server
- UAV Control: The human operator can use one of the PHP web pages to configure the UAV to fly in autonomous or manual mode. When the UAV is in autonomous mode, it follows the flying route waypoints. The

manual mode is a teleoperation from the web server. These commands are collected by the PHP code and stored in the database hosted by the MySQL server

- Flying route: The human operator can use one of the PHP web pages to set a flying route using up to 4 waypoints. These waypoints are collected by the PHP code and stored in the database hosted by the MySQL server. The UAV will follow this route when in autonomous mode
- Video Streaming: The human operator can use one of the PHP web pages to watch the video streaming of the webcam connected to the Raspberry Pi. This video streaming is provided by the Motion application and displayed in a web page hosted by its own web server. This web page is embedded in the PHP web page used by the human operator

Sensors-Actuators Infrastructure

The Sensors-Actuators Infrastructure (as shown in figure 4.4) describes the data flow between the main elements of the UAV prototype. This data flow can follow three different paths:

- Sensors-to-Navigation: the sensors values of the mobile phone are processed by the Navigation software and stored in the MySQL database.
- Navigation-to-Motors: the Navigation software processes the flying instructions stored in the MySQL database and sends the corresponding electric currents to the UAV motors.
- Web Interface-to-Motors: the Web Interface saves the flying instructions selected by the human operator in the MySQL database. These instructions are processed by the Navigation software which in turn sends the corresponding electric currents to the UAV motors.

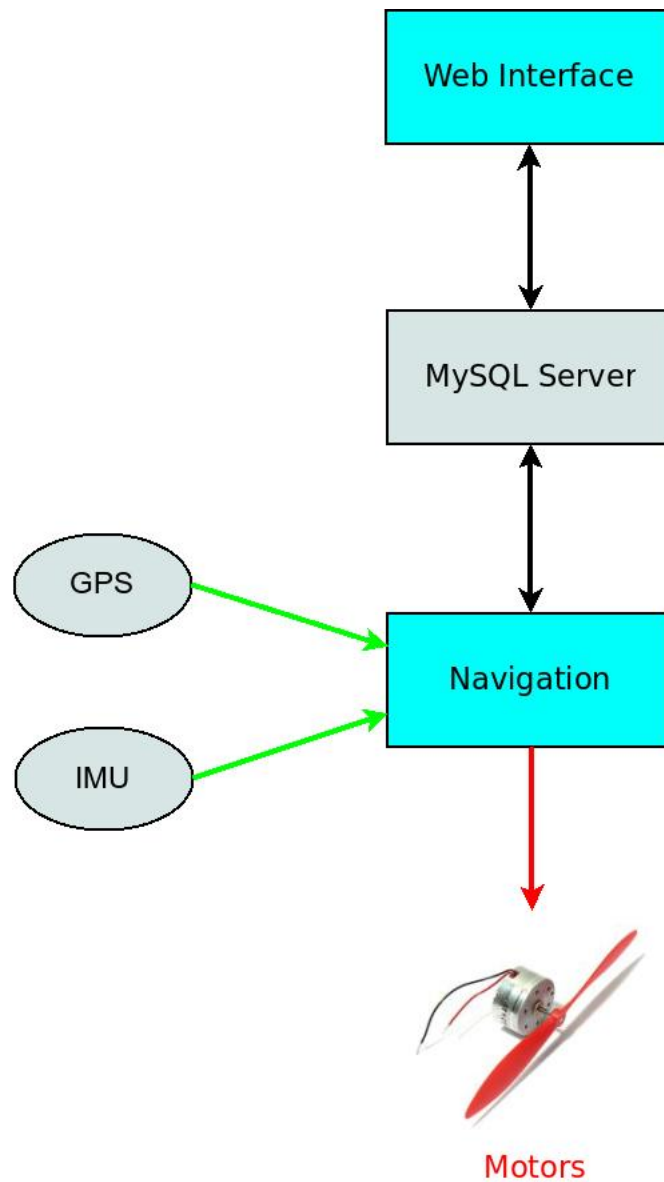


Figure 4.4: Sensors-Actuators Infrastructure

4.2 Navigation

This functionality uses the information obtained from the mobile phone GPS and IMU sensors to calculate where to go and what commands must be sent

to the motors to achieve it (as shown in figure 4.2).

This is performed by three applications running concurrently, first an Android application running on the mobile device, second a Java application running on the Raspberry Pi and a third application running on the Arduino microprocessor (as shown in figure 4.5). These applications are explained below.

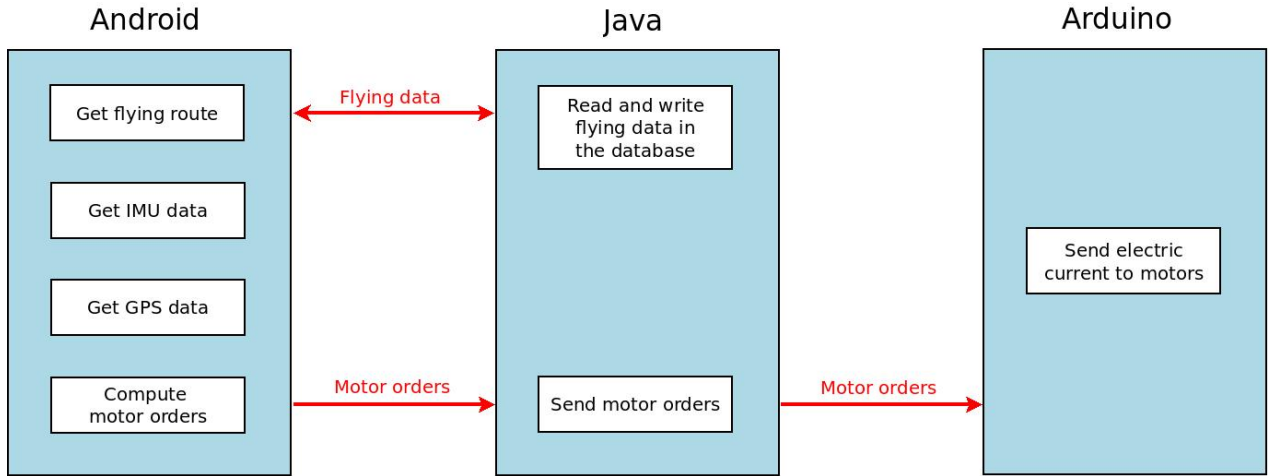


Figure 4.5: Navigation functionality applications

4.2.1 Android application

The Android application gets local information from the mobile phone IMU (magnetic sensor) and GPS sensors. The mobile phone is configured to connect automatically to the Raspberry Pi Wi-Fi network. The Android application communicates with the Java application running on the Raspberry Pi through TCP/IP protocol to read and write information about the UAV status and about the flying route from the database stored in the Raspberry Pi (as shown in figure 4.6).

It's composed of the following Android classes:

1. UAVActivity: It's the Android main class, it instantiates objects from the other classes of this list and implements the listeners which get information from mobile magnetic and GPS sensors. After the other objects are instantiated, UAVActivity waits until a sensor event triggers

the code implemented in the listeners. The GPS sensor waits until the location provided by the sensor is at least at 1 meter of distance from the last event and until it has passed at least one second from the last event. The magnetic sensor waits until it has passed at least 60 milliseconds from the last event.

2. FlightOperations: It stores in its variables part of the current UAV flight status:

- Latitude: current latitude of the UAV in decimal degrees.
- Longitude: current longitude of the UAV in decimal degrees.
- Altitude: current altitude of the UAV in meters. It's an estimation based on GPS data. It's less accurate than a real altimeter.
- Heading: current heading of the UAV in degrees. This information is provided from mobile magnetic sensors, based on the Earth magnetic field.
- Current waypoint: it stores the latitude and longitude of the current destination of the UAV. When this waypoint is reached, this information is updated with the next waypoint.

If the UAV is configured in autonomous mode, the object instantiated by this class calculates the desired heading using the above information. Comparing current heading with desired heading, the final order (turn left or turn right) is sent to UAVActivity methods which send the final order to the motors. This is done by the following methods:

```
public synchronized void setHeading(double destinationLatitude,
                                     double destinationLongitude)
{
    double auxLatitude;
    double auxLongitude;
    targetLatitude=destinationLatitude;
    targetLongitude=destinationLongitude;
    auxLatitude=destinationLatitude-shipLatitude;
    auxLongitude=destinationLongitude-shipLongitude;
    //If point is at north of ship
    if (auxLatitude>0)
    {
```

```

        headingNorth(auxLongitude);
    }
    //If point is at south
    else if (auxLatitude<0)
    {
        headingSouth(auxLongitude);
    }
    //If point is in the same latitude
    else if (auxLatitude==0)
    {
        sameLatitude(auxLongitude);
    }
    setEngines();
}

```

3. SerialTCPClient: The object instantiated by this class connects via TCP/IP with the Java application running on the Raspberry Pi called "AndroidArduinoBridge". The purpose of this connection is to ultimately send motor orders to Arduino through the Raspberry Pi. A small protocol has been designed for this purpose: An integer is send to AndroidArduinoBridge, which may have the following values:
 - '1': both motors are activated at full speed.
 - '2': both motors are stopped.
 - '3': left motor stopped, right motor full speed.
 - '4': left motor full speed, right motor stopped.
4. MySQL_TCPClient: The object instantiated by this class connects via TCP/IP with the Java application running on the Raspberry Pi called "AndroidArduinoBridge". The communication is bidirectional. The purpose of this connection is to send commands to AndroidArduinoBridge which has access to the MySQL server that is running on the Raspberry Pi. This server stores a database with the flying data, stored on a single table. There is only one row stored in the table, which contains the following columns:
 - CurrentLatitude: Current latitude of the UAV in decimal degrees.

- CurrentLongitude: Current longitude of the of the UAV in decimal degrees.
- Heading Target: The desired bearing to reach the next waypoint in degrees.
- TargetLatitude: The latitude of the current waypoint in decimal degrees.
- TargetLongitude: The longitude of the current waypoint in decimal degrees.
- Altitude: Current altitude of the UAV in meters.
- Autopilot: Autopilot status (ON/OFF).
- FlightOrder: The numeric value of the current order (i.e: '1': both motors full speed).
- Waypoint1Latitude: Waypoint 1 latitude in decimal degrees.
- Waypoint1Longitude: Waypoint 1 longitude in decimal degrees.
- Waypoint2Latitude: Waypoint 2 latitude in decimal degrees.
- Waypoint2Longitude: Waypoint 2 longitude in decimal degrees.
- Waypoint3Latitude: Waypoint 3 latitude in decimal degrees.
- Waypoint3Longitude: Waypoint 3 longitude in decimal degrees.
- Waypoint4Latitude: Waypoint 4 longitude in decimal degrees.
- Waypoint4Longitude: Waypoint 4 longitude in decimal degrees.

A numeric protocol has been created for this purpose. This protocol is a sequence of 2 double numbers for each command. The first number is the command code, and the second number is the data required for this command, or 0 if no additional information is required. Command codes are always a number less or equal to -500. Command data are always a number greater than -180. Although TCP protocol guarantees that packets are processed on the destination point in the correct order, this numeric protocol has been designed to prevent that a command data could be confused with a command code in the case of a software bug or network problem. The sequence of numbers can have the following values:

- -500,0: Ask if the current destination has changed due to a user modification of the flight plan.
- -1500,'coordinate': Records in database the current UAV latitude.
- -1501,'coordinate': Records in database the current UAV longitude.
- -1502,'heading': Records in database the current UAV heading.
- -1503,'heading': Records in database the desired UAV heading to reach the next waypoint.
- -1504,'coordinate': Records in database the UAV target latitude (the latitude of the next waypoint).
- -1505,'coordinate': Records in database the UAV target longitude (the latitude of the next waypoint).
- -1506,'altitude': Records in database the current UAV altitude.
- -1507,'autopilot status': Records in database the current UAV autopilot status (this command is used when autopilot status changes i.e. "autopilot ON").
- -1508,'flight order': Records in database the current UAV flight order (this command is used when flight order changes i.e. "turn left").
- -511,0: Gets waypoint 1 latitude.
- -512,0: Gets waypoint 1 longitude.
- -521,0: Gets waypoint 2 latitude.
- -522,0: Gets waypoint 2 longitude.
- -531,0: Gets waypoint 3 latitude.
- -532,0: Gets waypoint 3 longitude.
- -541,0: Gets waypoint 4 latitude.
- -542,0: Gets waypoint 4 longitude.
- -543,0: Gets autopilot status (ON/OFF).
- -544,0: Gets flight order (this command is used to know the current flight order i.e. "turn left").

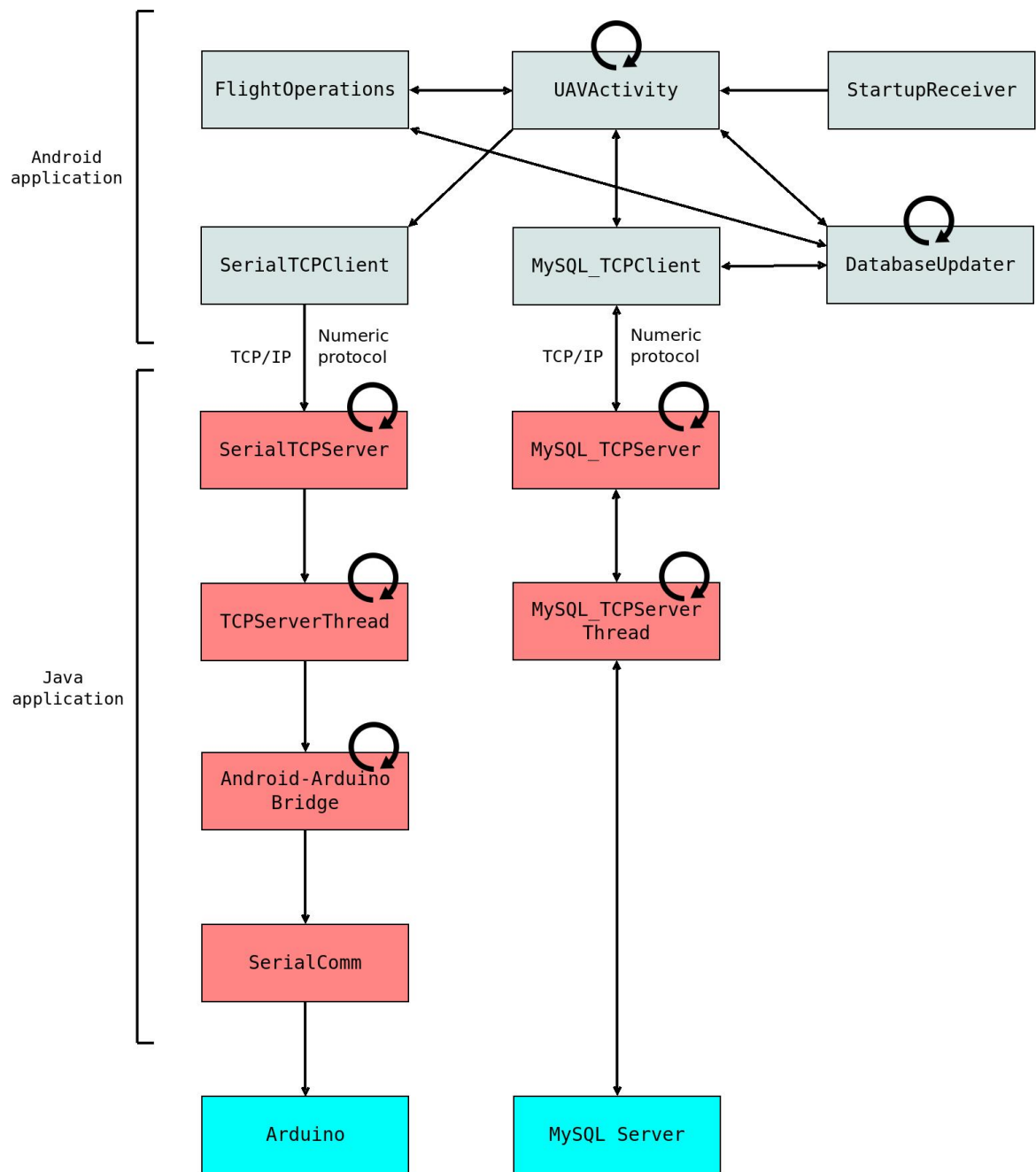


Figure 4.6: Navigation functionality software

5. DatabaseUpdater: The object instantiated by this class is a thread that reads and updates the database running on the Raspberry Pi one time per second. This is done to check if the user has changed the flight route, the autopilot mode, and if this is in manual mode, the flight order (i.e. "turn left"). It's done also to update the information about the UAV status like current coordinates, altitude, autopilot status, heading, target latitude, target longitude and flight order in the database, so the user can get the current UAV status in real time.

When UAVActivity instantiates the object of this class, the constructor of this class sends the sequence "-1507,1" to AndroidArduinoBridge. This enables the UAV autopilot which is very important if the mobile reboots itself when the UAV is in manual mode. This makes the UAV to fly to the next waypoint, preventing the UAV to fly in an uncontrolled manner and giving time to the user to activate manual control again if it's required. The constructor sends the sequence "-1508,0" to AndroidArduinoBridge too to stop motors when device boots to prevent the UAV continues with the last flight order in the case of an accidental system restart and to prevent any injury to the user if near the UAV propellers.

6. StartupReceiver: The object instantiated by this class is created when the mobile device boots, or after an accidental mobile restart. Its only purpose is to launch UAVActivity which creates the other objects as explained above. Without this class, if the mobile device reboots itself the UAV would lose any automatic or manual control capabilities and the motors would execute the last flight order (i.e. "turn left") until the arduino batteries are spent. If the mobile reboots, the system configures itself in automatic mode, calculates how to fly to the next waypoint and sends the corresponding commands to the motors.

There are 2 threads in the Android application: UAVActivity (the main application thread) and DatabaseUpdater. They share code of MySQL_TCPCClient. This code is protected by "synchronized" Java keyword to grant mutual exclusion. This shared code has neither race conditions nor deadlocks.

This Android application has 865 lines of code and is mainly an event-driven system: It does nothing until a sensor event triggers the code implemented in the listeners of the UAVActivity class with the exception of

the DatabaseUpdater thread, which runs one time per second to update the behavior of the UAV with the information stored in the MySQL database.

4.2.2 Java application

The Java application runs on the Raspberry Pi and as shown in figure 4.6 serves as a bridge first between the Android application and the Arduino device and second between the Android application and the MySQL server that runs on the Raspberry Pi.

When the Android application wants to send a motor command to the Arduino device, it must first send the motor command to the Java application, which in turn transfers the motor command to the Arduino device. When the Android application wants to update the flight information in the MySQL database it must do it through the Java application too.

It is composed of the following classes:

- AndroidArduinoBridge: the object instantiated by this class creates 2 threads from the SerialTCPServer and MySQLTCPServer classes which listen for incoming connections. It instantiates an object from the SerialComm class which allow this and other objects to send commands through serial port. This is used for AndroidArduinoBridge to make a motor test at system startup, and for TCPSThread threads to send commands through serial port.

```
private static SerialComm mySerialComm;

public static void main(String[] args)
throws InterruptedException
{
    mySerialComm = new SerialComm();
    System.out.println("Android-Arduino Bridge started");
    servoTest();
    try
    {
        Thread mySerialTCPServer = new SerialTCPServer();
        mySerialTCPServer.start();
        Thread mySQLTCPServer = new MySQLTCPServer();
        mySQLTCPServer.start();
    }
}
```

```

    }
    catch (IOException e)
    {
        System.out.println("Error: Can't create TCP
                           servers");
        e.printStackTrace();
    }
}

```

- MySQL_TCPServer: the object instantiated by this class listens for incoming connections from the Android application. When it receives a new connection it creates a thread from MySQL_TCPServerThread class.
- MySQL_TCPServerThread: the thread created from this class receives commands from the Android application. It uses the same protocol of the MySQL_TCPClient class of the Android application.

```

while (true)
{
    command=myDataInputStream.readDouble();
    System.out.println("[DB Server] Client says: "
                       + command);
    // Command codes begin with -500
    if (command>-500)
    {
        System.out.println("[DB Server] " + command
                           + " is not a valid command");
    }
    else
    {
        System.out.println("[DB Server] " + command
                           + " accepted");
        data=myDataInputStream.readDouble();
        // Command data is always >= -180
        if (data<-180)
        {
            System.out.println("[DB Server] " + data +

```

```

        " is not a valid data to command " + command);
    }
    else
    {
        System.out.println("[DB Server]
            Executing" + command + " with " + data);
        executeOrder(command, data);
    }
}
}

```

- SerialTCPServer: the object instantiated by this class listens for incoming connections from the Android application. When a new connection is received it creates a thread from TCPServerThread class.
- TCPServerThread: the thread created from this class receives the motor commands from the Android application according to the same protocol used by the SerialTCPClient class of the Android application. When the motor commands are received they are sent to the Arduino application using the object instantiated from the AndroidArduinoBridge class which in turn uses an instance of SerialComm.

```

public class TCPServerThread extends Thread
{
    private DataInputStream myDataInputStream;

    public TCPServerThread(DataInputStream serverDataInputStream)
    {
        myDataInputStream=serverDataInputStream;
    }

    public void run()
    {
        int code;
        try
        {
            while (true)
            {

```

```

        code=myDataInputStream.readInt();
        System.out.println("[Server] Client says: "
                           + code);
        AndroidArduinoBridge.setEngine(code);
    }
}
catch (IOException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

- SerialComm: the object instantiated by this class creates a serial port connection with the Arduino application. This connection is used to send the motor commands to the Arduino device.

```

public synchronized void serialEvent(SerialPortEvent oEvent)
{
    if (oEvent.getEventType() ==
        SerialPortEvent.DATA_AVAILABLE)
    {
        try
        {
            int available = input.available();
            byte chunk[] = new byte[available];
            input.read(chunk, 0, available);

            // Displayed results are codepage dependent
            System.out.print(new String(chunk));
        }
        catch (Exception e)
        {
            System.err.println(e.toString());
        }
    }
}

```

```

}

public synchronized void writeSerial(int myByte)
{
    try
    {
        System.out.println("[Server] Sending code "+
                           myByte+" to serial port");
        output.write(myByte);
    }
    catch (IOException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

There are several threads running in this layer, whose code is designed to avoid any race condition or deadlock. This threads are:

- **AndroidArduinoBridge**: it's the main application thread, which is an instance of the `AndroidArduinoBridge` class. It has a method to send commands through serial port which is used by the thread created as an instance of the `TCPServerThread` class.
- **MySQL_TCPServer**: it's created by the main application thread as an instance of the `MySQL_TCPServer` class. It waits for incoming connections, and when this happens, it creates a new thread as an instance of the `MySQL_TCPServerThread` class.
- **MySQL_TCPServerThread**: this thread is created as an instance of the `MySQL_TCPServerThread` class. It sends and receives information to and from the Android application through the TCP/IP protocol. If the connection with the Android application is lost, a new thread of the same type is created from `MySQL_TCPServer` when it receives a new connection from the Android application.
- **SerialTCPServer**: it's a thread created by the main application thread as an instance of the `SerialTCPServer` class. It waits for incoming con-

nections, and when this happens it creates a new thread as an instance of the `TCPServerThread` class.

- `TCPServerThread`: it's created by `SerialTCPServer` as an instance of the `TCPServerThread` class. This thread sends commands to the Arduino application through the serial port and receives information from the Android application through the TCP/IP protocol. If the connection with the Android application is lost, a new thread of the same type is created from `SerialTCPServer` when it receives a new connection from the Android application.

This Java application has 641 lines of code and is an event-driven system. It waits until the Android application sends it a command and then performs the tasks associated to that command.

4.2.3 Arduino application

This small application receives the motor orders from the Java application through the serial port connection and sends the corresponding electric currents to the motors through the Arduino Motor Shield outputs (as shown in figure 4.7). It uses the same protocol as the `SerialTCPClient` class of the Android application. It has 72 lines of code.

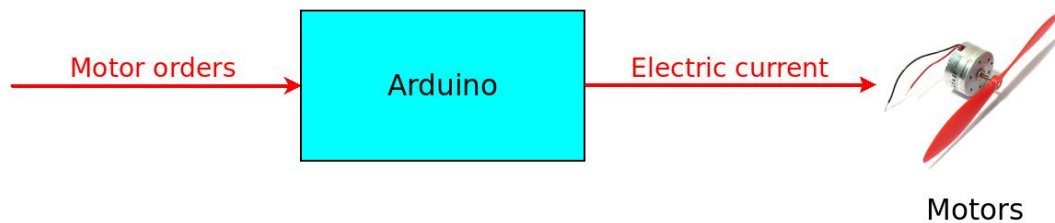


Figure 4.7: Functional diagram of the Arduino application

```
if (incomingByte==1)
{
    Serial.println("[Arduino] Propeller status: Full");
}
```

```

//Motor A forward @ full speed

//Establishes backward direction of Channel A
digitalWrite(12, LOW);
//Disengage the Brake for Channel A
digitalWrite(9, LOW);
//Spins the motor on Channel A at full speed
analogWrite(3, 1023);

//Motor B forward @ full speed

//Establishes backward direction of Channel B
digitalWrite(13, LOW);
//Disengage the Brake for Channel B
digitalWrite(8, LOW);
//Spins the motor on Channel B at full speed
analogWrite(11, 1023);
}

```

4.3 Web Interface

This software is composed of four applications running on the Raspberry Pi. These applications are:

1. UAV Manager: this is a PHP web site which is used by the human operator to monitor and control the UAV prototype.
2. Apache Web Server: this server runs on the Raspberry Pi and hosts the UAV Manager web pages.
3. MySQL server: this server stores a database with all the UAV information like internal status and the flight route selected by the human operator. This database allows information sharing between the Android application and the UAV Manager.
4. Motion: this application runs on the Raspberry Pi and is used to capture pictures from the webcam connected to the Raspberry Pi.

4.3.1 UAV Manager web application

This is a PHP web site which allows the human operator to monitor and control the UAV prototype using any device with web browser and Wi-Fi. This website is hosted on the Apache Web Server that runs on the Raspberry Pi. The user needs first to connect to the Raspberry Pi Wi-Fi network, which is done using the user device standard procedure to join to any Wi-Fi network. No special steps or additional software is required. When this is done, the user only has to open a web browser and write the IP address of the Raspberry Pi (i.e. `http://192.168.1.33/UAV`). This web site uses CSS³ via HTML⁴ "style" property. This web site has 469 lines of code. The web pages are:

- Main page: This web page only shows an intro picture and the list of hyperlinks to the other pages.
- Route: This web page has a small form which is used to program up to 4 waypoints in the UAV. The waypoints are GPS coordinates expressed in decimal degrees. These coordinates are referenced to the WGS84 datum⁵, the same used in GPS devices and Google Maps. It's assumed that there are no obstacles. The UAV calculates how to go in a straight line from the current position to the next waypoint and so on. This web page launches PHP code on the server side which accesses to the MySQL server that runs on the Raspberry Pi. This code stores the waypoints in the MySQL database, where they can be read by the Java application that runs on the Raspberry Pi. The Java application in turn transfers this information to the Android application when requested.

```
$con = mysql_connect("localhost","Drone","FoxtrotRomeo");  
  
if (!$con)  
{  
    die("I can't connect to database: " . mysql_error());  
}
```

³Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Almost all web pages use CSS style sheets to describe their presentation.

⁴HTML or HyperText Markup Language is the standard markup language used to create web pages.

⁵A geodetic datum is a reference system that describes the surface of the Earth

```

}

mysql_select_db("UAV", $con);

$status=true;

if ($status)
{
    $status=mysql_query("UPDATE Status
                        SET Waypoint1Latitude=" . $Waypoint1Latitude . " ");
}

if ($status)
{
    $status=mysql_query("UPDATE Status
                        SET Waypoint1Longitude=" . $Waypoint1Longitude . " ");
}

```

- Status: This web page shows the relevant information of the UAV including autopilot mode, current position, orientation, waypoints, etc. To do this, it launches PHP code on the server side which accesses to MySQL server that runs on the Raspberry Pi. This information is read from the MySQL database one time per second.

```

$con = mysql_connect("localhost","Drone","FoxtrotRomeo");
if (!$con)
{
    die("I can't connect to database: " . mysql_error());
}
mysql_select_db("UAV", $con);
$sql="SELECT * FROM Status";
$result = mysql_query($sql);
mysql_close($con);

$row = mysql_fetch_array($result);

if ($row["Autopilot"]==0)

```

```

{
    echo '<h1>Autopilot: OFF</h1>';
}
else if ($row[" Autopilot"]==1)
{
    echo '<p>Autopilot: ON</p>';
}

echo '<p>Latitude:  '.$row[" CurrentLatitude"].'</p>';
echo '<p>Longitude:  '.$row[" CurrentLongitude"].'</p>';
echo '<p>Altitude:  '.$row[" Altitude"].' meters</p>';
echo '<p>Heading:  '.$row[" Heading"].'&deg;</p>';
echo '<p>Target Heading:  '.$row[" Target"].'&deg;</p>';
echo '<p>Target Latitude:  '.$row[" TargetLatitude"].'</p>';
echo '<p>Target Longitude:  '.$row[" TargetLongitude"].'</p>';
echo '<p>Waypoint1 [ Latitude:  '.$row[" Waypoint1Latitude"].
    ' - Longitude:  '.$row[" Waypoint1Longitude"].' ]</p>';
echo '<p>Waypoint2 [ Latitude:  '.$row[" Waypoint2Latitude"].
    ' - Longitude:  '.$row[" Waypoint2Longitude"].' ]</p>';
echo '<p>Waypoint3 [ Latitude:  '.$row[" Waypoint3Latitude"].
    ' - Longitude:  '.$row[" Waypoint3Longitude"].' ]</p>';
echo '<p>Waypoint4 [ Latitude:  '.$row[" Waypoint4Latitude"].
    ' - Longitude:  '.$row[" Waypoint4Longitude"].' ]</p>';

```

- Control: This web page displays a video streaming from the webcam connected to the Raspberry Pi. This video streaming is provided by the Motion application which has its own web server. This web server hosts a web page with the video streaming and that web page is embedded in this one. This web page also displays the current position, altitude, heading, and a button to switch between automatic and manual modes.

In manual mode the UAV is controlled pressing buttons in the same web page. These buttons launch PHP code on the server side which accesses to MySQL server that runs on the Raspberry Pi. This PHP code stores the flight order in the MySQL database where it can be read by the Java application that runs on the Raspberry Pi. The Java application in turn transfers this information to the Android application when requested. In automatic mode the UAV calculates how to follow the planned route

flying in straight line to the next waypoint.

```
<!--This iframe displays the video streaming
      of the Motion web server-->
<iframe height="300" width="400" frameBorder="0"
        src="http://192.168.1.33:8081" >

</iframe>

<?php

mysql_select_db("UAV", $con);
$sql="SELECT Autopilot FROM Status";
$result = mysql_query($sql);
mysql_close($con);
$row = mysql_fetch_array($result);

if ($row["Autopilot"]==0)
{
    echo '<iframe src="manual.php" width="100%" height="330"
        frameborder="0" name="console">
        </iframe>';
}
else if ($row["Autopilot"]==1)
{
    echo '<iframe src="auto.php" width="100%" height="330"
        frameborder="0" name="console">
        </iframe>';
}
?>
```

4.3.2 Apache Web Server

This server (version 2.2.22) runs on the Raspberry Pi. It's a default installation from Debian repositories including the PHP5 Apache module. It hosts the UAV Manager web pages.

4.3.3 Motion

This application (version 3.2.12) runs on the Raspberry Pi and was installed from Debian repositories. The configuration of this application is done editing a text file inside the file system of the Raspberry Pi. This configuration is customized to capture a picture from the webcam connected to the Raspberry Pi and to display it in the web page hosted by the Motion web server. This web page is embedded in a PHP web page of the UAV Manager web application.

The picture captured by the Motion application is updated each 4 seconds. This is done because the computing capabilities of the Raspberry Pi are limited and don't allow a more fluent video streaming, at least with the particular webcam model that has been used.

4.3.4 MySQL server

This server (version 5.5.28-1) runs on the Raspberry Pi and was installed from Debian repositories. It hosts a single database whose size is only 16 KB. This database has a single table with a single row which is updated periodically so the database never grows in size.

4.4 Sensors-Actuators Infrastructure

Here will the data flow between the main elements of the UAV prototype be explained:

- Sensors-to-Navigation: This section explains the data flow between the UAV sensors and the Navigation software (figure 4.8)
- Navigation-to-Motors: This section explains the data flow between the Navigation software and the UAV motors (figure 4.9)
- Web Interface-to-Motors: This section explains the data flow between the Web Interface and the UAV motors used in teleoperation mode (figure 4.10)

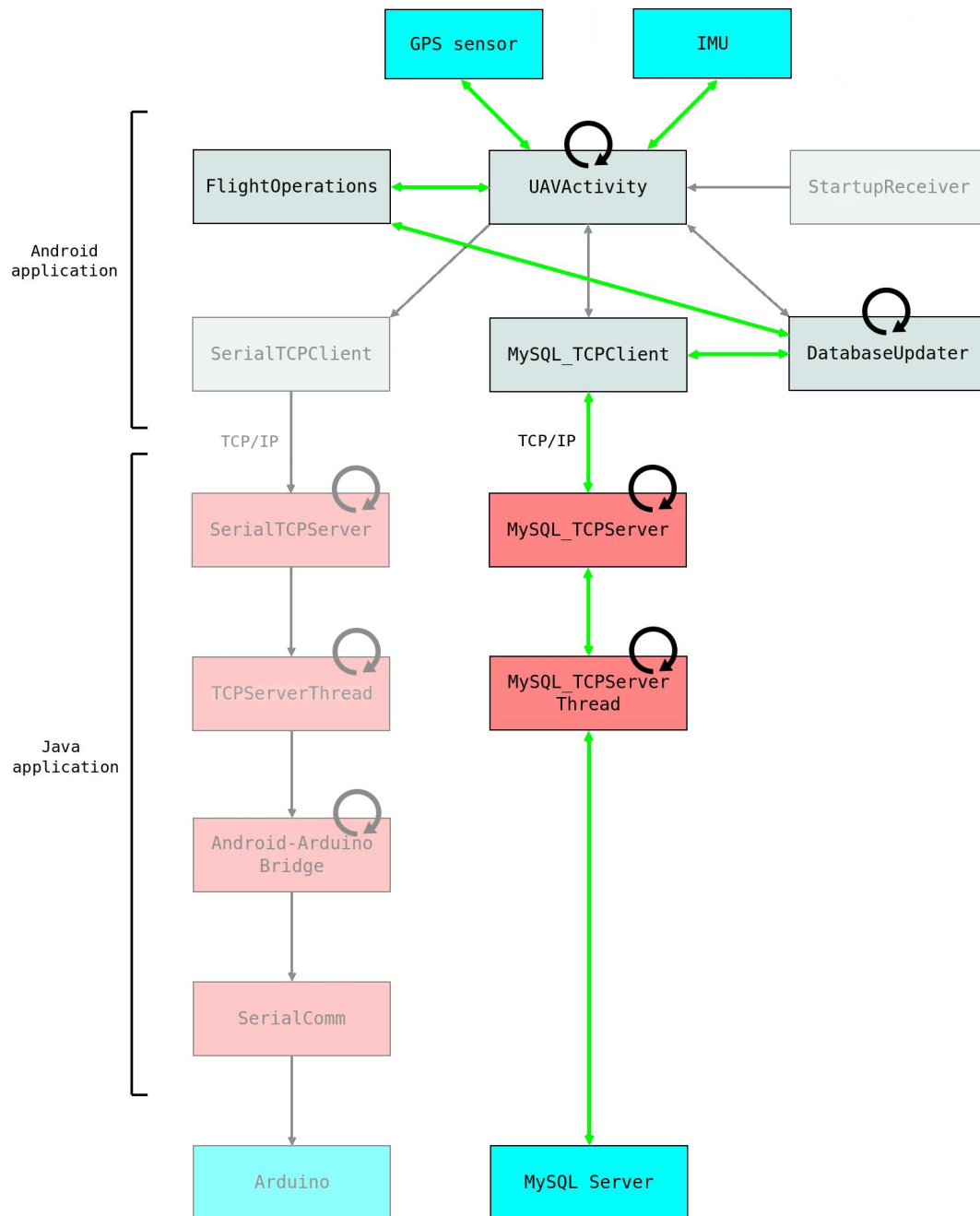


Figure 4.8: Data flow between the UAV sensors and Navigation software

4.4.1 Sensors-to-Navigation

The UAVActivity class implements sensor listeners which get the information provided by the mobile sensors (GPS and IMU) as shown in figure 4.8. These listeners are configured to update GPS values each 1 second (only if the new location is at least at 1 meter away from the last location) and to update IMU values each 60 milliseconds too. When the information provided by the mobile sensors is received, UAVActivity sends it to the FlightOperations class which stores it to make flying calculations. The DatabaseUpdater class gets the sensors information from the FlightOperations class one time per second and sends it to the MySQL_TCPClient class which connects to the MySQL_TCPServer class via TCP/IP. Then MySQL_TCPServer creates a thread as an instance from the MySQL_TCPServerThread class which receives the sensors information from MySQL_TCPClient. Finally MySQL_TCPServerThread connects with the MySQL server via TCP/IP and stores the sensors information in the database.

4.4.2 Navigation-to-Motors

The Navigation software can decide to send a motor command (as shown in figure 4.9) when a sensor event occurs or when the MySQL database is updated by the human operator. The sensor events are triggered one time per second for the GPS sensor and each 60 milliseconds for the magnetic sensor, but only when the new sensor values cause a change in the flying calculations a motor command is generated. Database updates are detected with an unknown frequency because it depends of the human operator although this frequency can't be higher than one update per second. When due to these causes the Navigation software decides to send a motor command, the UAVActivity class sends it to the SerialTCPClient class, which connects with the SerialTCPServer class via TCP/IP. Then SerialTCPServer creates a thread from the TCPServerThread class which receives the motor command from SerialTCPClient and sends it to a method from the Android-Arduino Bridge class. Then Android-Arduino Bridge sends the motor command to the SerialComm class which is connected with the Arduino application via serial port. Finally SerialComm sends the motors command to the Arduino application which sends the corresponding electric currents to the motors.

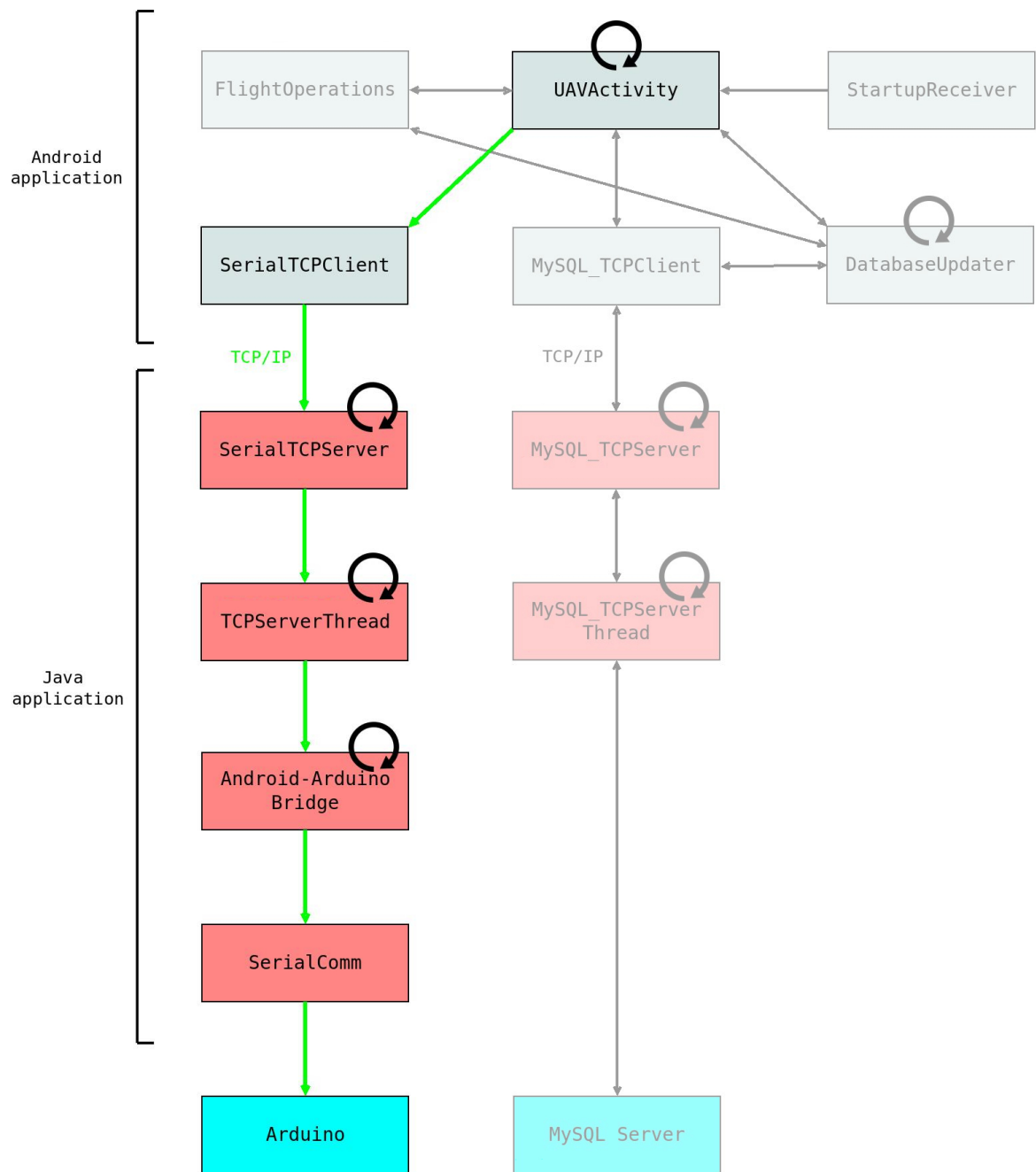


Figure 4.9: Data flow between the Navigation software and the UAV motors

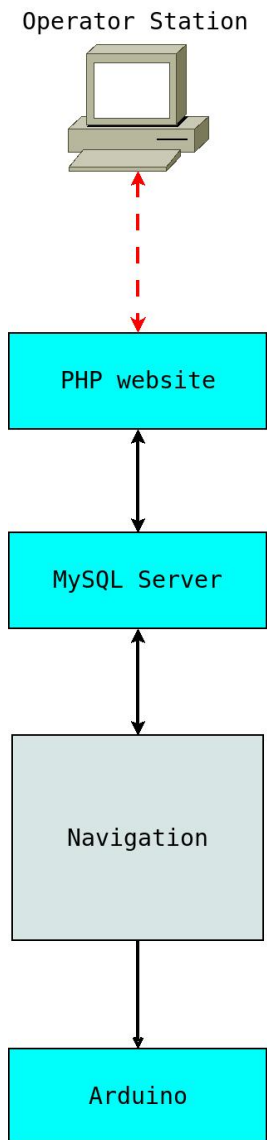


Figure 4.10: Data flow between the Web Interface and the UAV motors

4.4.3 Web Interface-Motors

The Operation Station (as shown in figure 4.10) connects via Wi-Fi to the Apache web server running on the Raspberry Pi and loads the PHP website in a web browser. Then the human operator interacts with the PHP website

to request either configure the UAV in manual mode and send it a flight order, or send a new flying route which will be used in the flying calculations when the UAV is configured in autonomous mode. In both cases the PHP code running on the server side receives this information and stores it in the MySQL database.

The Navigation software reads the MySQL database one time per second and calculates if a new motor command must be executed. If so, the Navigation software sends the motor command to Arduino which in turn sends the corresponding electric currents to the motors.

Chapter 5

Experiments

In this chapter the experiments performed with the UAV prototype will be explained. The section 5.1 provides a description of the physical implementation of the UAV prototype, the section 5.2 explains how the UAV prototype was monitored and controlled remotely and the section 5.3 explains how a real flying platform was assembled and tested.

5.1 UAV test board assembly

The main structure was a plastic grid to which the following hardware components were attached using cable-tie fasteners (as shown in figure 5.1):

- Mobile device (Samsung Galaxy Mini)
- Raspberry Pi
- Arduino Uno with Arduino Motor Shield attached
- A Wi-Fi adapter connected via usb with the Raspberry Pi
- 2 small motors with propellers attached connected with cables to the Arduino Motor Shield
- A battery to provide power to all the components excepting the mobile device which has its own battery

The webcam was not attached to the test board. When it came to test the video streaming it was connected via usb with the Raspberry Pi. The weight of all the hardware was between 1.5 and 2 kg. The test board was implemented according to the architecture described in section 4.1.

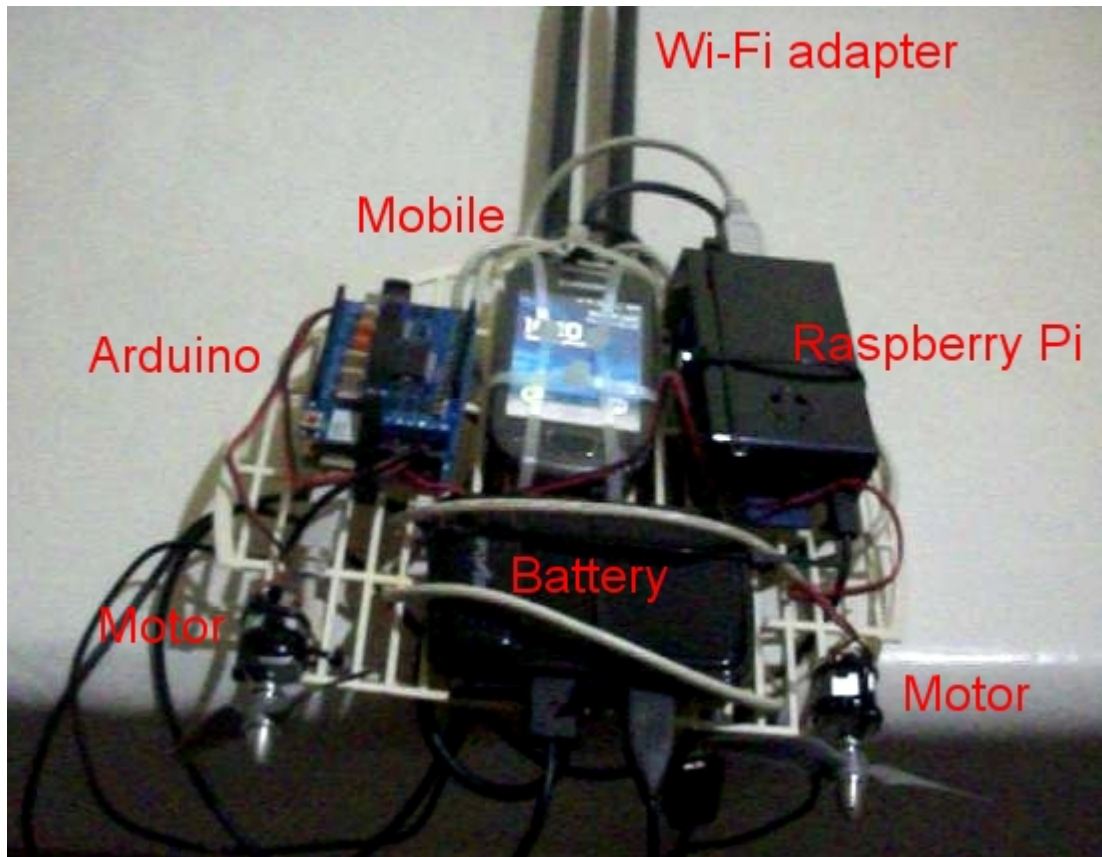


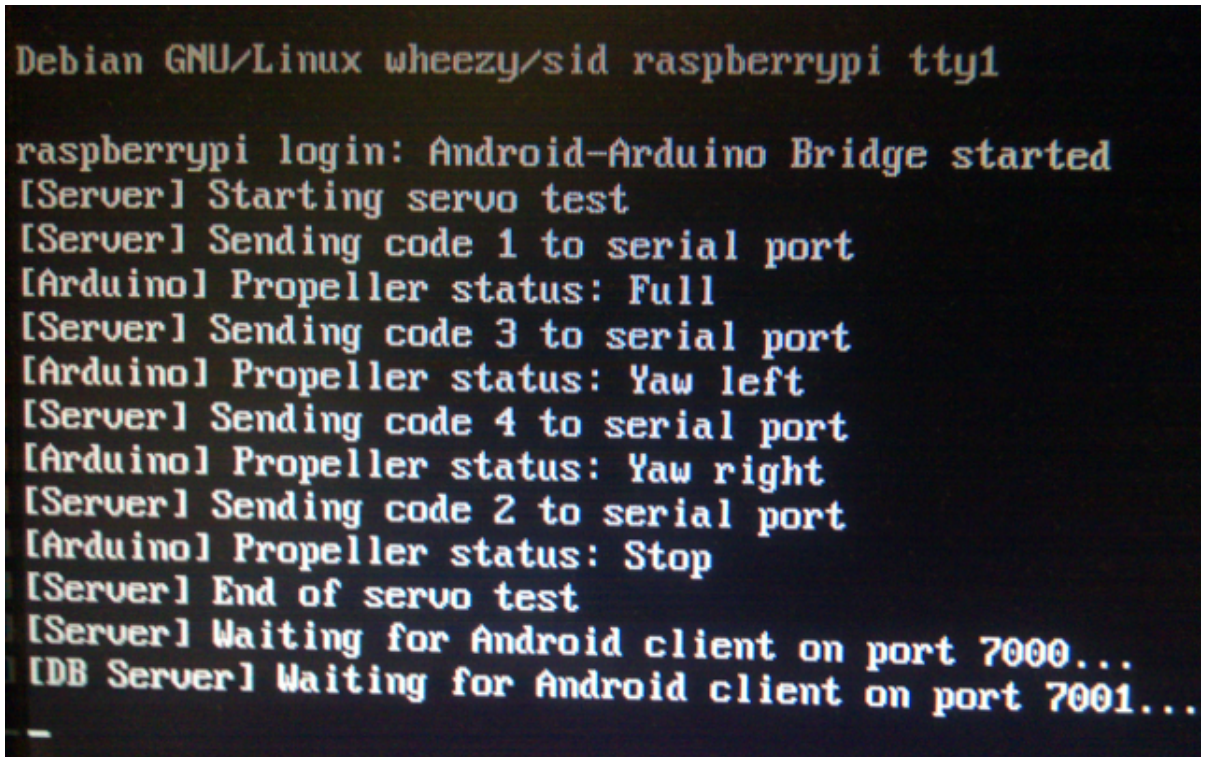
Figure 5.1: Test board

5.2 UAV platform teleoperation

This experiment was divided into several stages:

System startup

The system startup procedure was performed successfully. It was subdivided into the following stages:



```
Debian GNU/Linux wheezy/sid raspberrypi tty1

raspberrypi login: Android-Arduino Bridge started
[Server] Starting servo test
[Server] Sending code 1 to serial port
[Arduino] Propeller status: Full
[Server] Sending code 3 to serial port
[Arduino] Propeller status: Yaw left
[Server] Sending code 4 to serial port
[Arduino] Propeller status: Yaw right
[Server] Sending code 2 to serial port
[Arduino] Propeller status: Stop
[Server] End of servo test
[Server] Waiting for Android client on port 7000...
[DB Server] Waiting for Android client on port 7001...
```

Figure 5.2: System startup

1. Turn on the battery: after turning on the battery, the Raspberry Pi started and ran a test to check that the motors were working properly (as shown in figure 5.2). In this test the motors started to first spun the propellers individually, then spun both propellers simultaneously, and finally stopped both propellers.
2. Turn on the mobile device: when the mobile device was turned on, it connected automatically with the Wi-Fi access point of the Raspberry Pi and the Android application started automatically. After the Android application started, it connected via TCP/IP with the java

application running on the Raspberry Pi and turned on the motors according to the last motor order stored in the MySQL database.

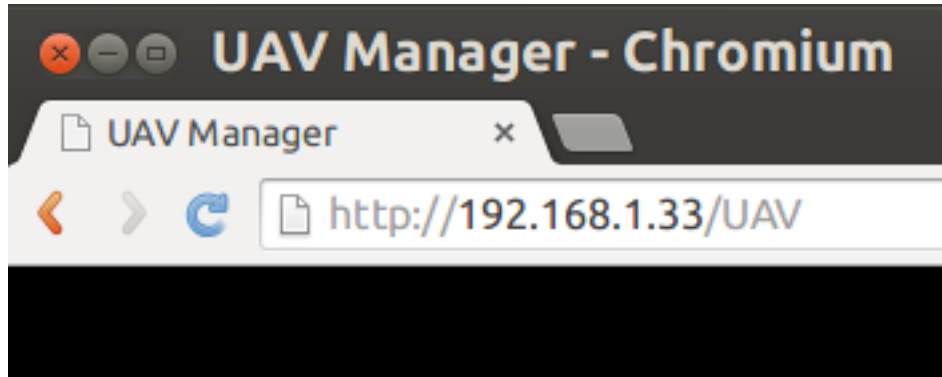


Figure 5.3: Opening the Web Interface

Remote connection

The remote connection procedure was performed successfully. It was subdivided into the following stages:

1. Connect to the Wi-Fi access point: using a desktop PC running the Ubuntu operating system, the human operator connected to the Wi-Fi access point created on the Raspberry Pi device using the usual procedure to connect to any other Wi-Fi network (i.e.: check the list of the available Wi-Fi networks and select the "Drone" network).
2. Open the Web Interface: the human operator opened a web browser and entered the URL of the Web Interface (as shown in figure 5.3) which showed the default web page (as shown in figure 5.4).

System monitoring

Once the Web Interface was opened, the "Status" tab (as shown in figure 5.5) was selected. The internal status of the UAV prototype was checked using the information displayed in this web page. This information was correct and consisted of the following elements:

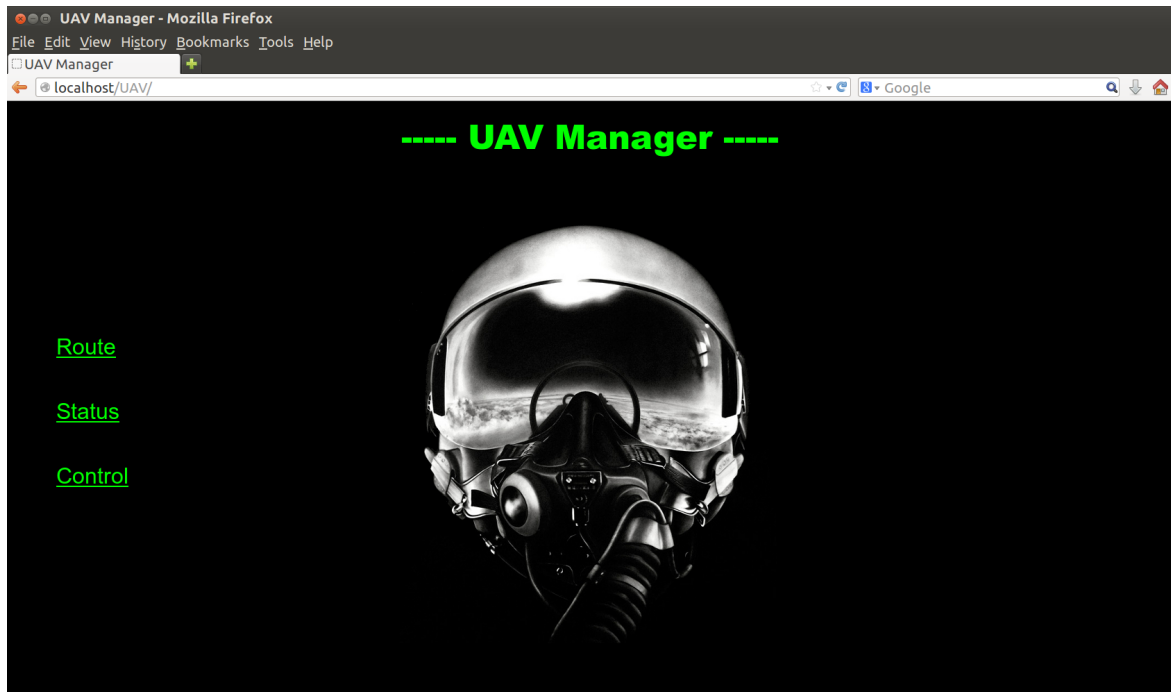


Figure 5.4: Web Interface default web page

- Autopilot status: it was the same that was previously selected by the human operator. When the mobile device had just switched on or restarted, the autopilot status was automatically configured as ON regardless its previous state.
- Location: the location values (latitude, longitude and altitude) were the same that the last values stored in the database. When the mobile device GPS sensor obtained new location values the old values were updated with the current ones.
- Heading: the heading value was compared with a real compass and they displayed roughly the same values. When the test board was rotated the heading value was updated too.
- Waypoints: the waypoints values were the same that the last values stored in the database. When the human operator updated these values using the "Route" tab (as shown in figure 5.6) they were successfully displayed.

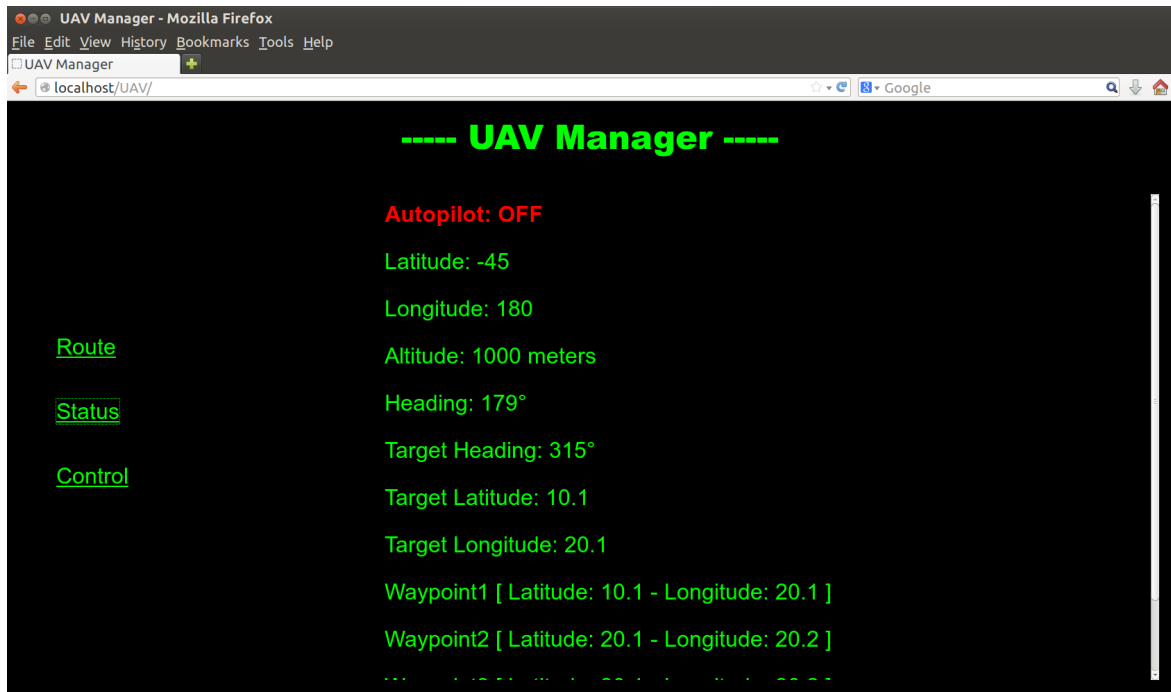


Figure 5.5: System monitoring

- Target values: the target latitude and longitude were the same as in the first waypoint. The target heading was the course which should fly the UAV prototype from the current coordinates to the target coordinates.

Teleoperation in manual mode

For this test, the "Control" tab in the Web Interface was selected (as shown in figure 5.7). The current location and heading were displayed successfully, showing the same values as in the "Status" option. When the button to enable/disable the autopilot was clicked the manual control buttons were enabled and worked successfully emulating real conditions as described below:

- "Forward" button: when pressed it turned on both motors.
- "Stop" button: when pressed it turned off both motors.
- "Left" button: when pressed it turned off the left motor and turned on the right motor.

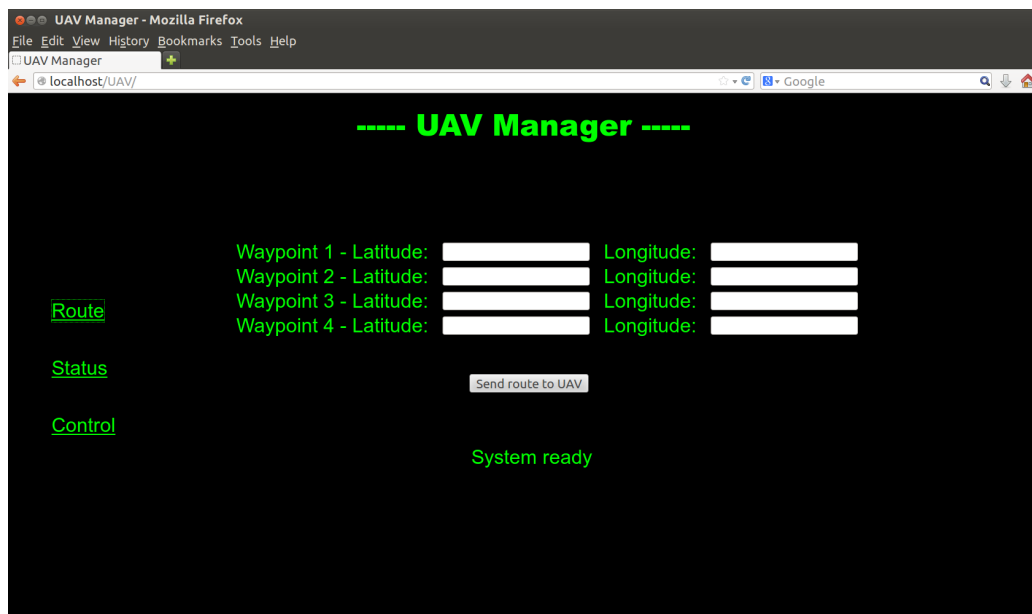


Figure 5.6: Route tab



Figure 5.7: control section of manual mode

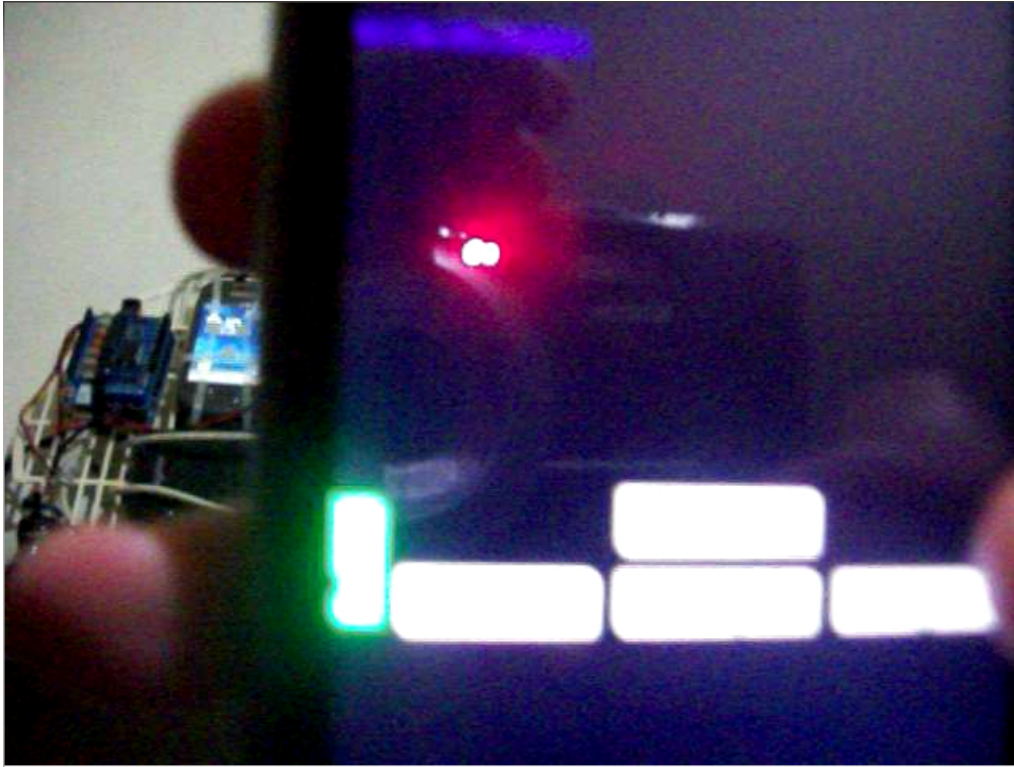


Figure 5.8: smartphone remotely connected to the UAV prototype

- "Right" button: when pressed it turned on the left motor and turned off the right motor.

There is a video on the project MediaWiki[18] showing a human operator connecting to the UAV prototype through a mobile device and using the web interface to control the motors (as shown in figure 5.8).

The "Control" web page displayed a video feed (as shown in figure 5.9) from the webcam with a very slow frame rate (1 frame every 4 seconds). The Raspberry Pi was unable to get a higher frame rate from the webcam model used for this experiment.

Autonomous mode configuration

For this test, the "Control" tab in the Web Interface was selected (as shown in figure 5.10) which displayed a video feed with the same features as the

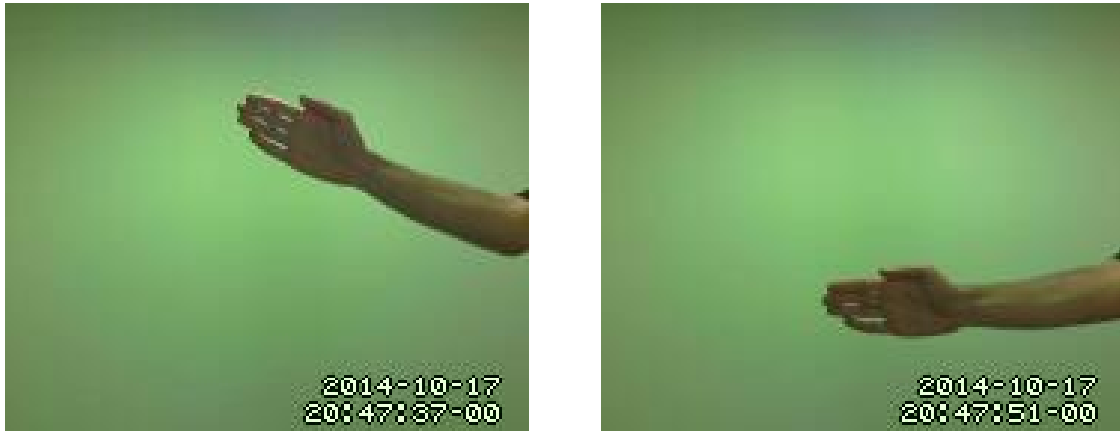


Figure 5.9: Testing video streaming



Figure 5.10: control section of autonomous mode

manual mode. The current location and heading were displayed successfully, showing the same values as in the "Status" option. When the button to enable/disable the autopilot was clicked the motors were activated successfully

emulating real conditions as described below:

- When the target heading was at the left side of the current heading a "yaw left" action was required: the left motor was turned off and the right motor was turned on.
- When the target heading was at the right side of the current heading a "yaw right" action was required: the left motor was turned on and the right motor was turned off.
- When the target heading was the same as the current heading, a "fly forward" action was required: both motors were turned on.

5.3 Assembly of the "Phantom" RC airplane

This experiment was made to study the assembly of another flying platform for future developments because the components used to build the computing architecture were too heavy to fly onboard of the first flying platform.

The "Phantom" is sold disassembled by the vendor, so the different components of the airplane had to be glued and in some cases secured with fiberglass tape (as shown in figure 5.11). The servos had to be attached to the control surfaces of the wings and the propeller was attached to the motor.

Finally a RC receiver was connected to the servos, the motor and the battery. To check the airplane was properly assembled, the following test (there is a video on the project MediaWiki[19]) was performed successfully:

- Connect to the RC receiver using a RC transmitter: this process was confirmed by the receiver with a beep sequence.
- The RC transmitter was operated to send the "pitch up", "pitch down", "roll left" and "roll right" commands to the RC receiver which in turn sent electric currents to the servos to move the control surfaces in a corresponding manner.
- The RC transmitter was operated to send a range of "thrust" commands from 0% to 100% to the RC receiver and the propeller spun in a corresponding manner.



Figure 5.11: "Phantom" RC airplane assembled

Chapter 6

Conclusions

The utility of this project has been to be a first contact with the Unmanned Aerial Vehicles. Many new technologies and concepts have been learned from scratch along this project, which will be very useful for future developments. Section 6.1 reviews the objectives described in Chapter 2 and describes to what extent they have been met. Finally section 6.2 explores some ideas for future developments.

6.1 Conclusions

The global goal for this project was divided into several subgoals:

- Design of an UAV prototype including hardware and software architecture: the UAV prototype was successfully designed and implemented as explained in section 4.1. The hardware components were assembled on a test board as explained in section 5.1 over which several tests were run successfully. Although not specified as a goal, the "Phantom" airplane was assembled and tested as explained in section 5.3.

Both the hardware architecture and the software architecture are far from being the best, but the priority was to spend as little money as possible and to finish before the deadline.

- Development of an API¹ to read the sensors and to control the actuators

¹An Application Programming Interface (API) is a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.

on the UAV prototype: this API was successfully implemented as the Sensors-Actuators Infrastructure explained in section 4.4. It allows the UAV prototype to receive the current location and orientation from the mobile phone sensors and to control the motors assembled on the test board.

- Development of a web interface to monitor and control the UAV prototype remotely: The Web Interface software explained in section 4.3 allows to monitor and to teleoperate the UAV prototype successfully. The video streaming was not specified as a goal but was implemented as a proof of concept: it worked fine but had a very slow frame rate to teleoperate a real flight. The Navigation functionality explained in section 4.2 allowed to calculate the flying route successfully. When the autonomous mode was enabled it activated the motors as expected.

Furthermore the UAV prototype was expected to have the following features, which were achieved successfully:

- The UAV prototype uses only free software tools. The full source code of the project applications is free software under GPLv3 License[20] and is available at project svn repository[17].
- Real time: The UAV prototype can be teleoperated through the Web Interface with a delay of less than a second, which may be enough to control a slow moving aircraft like an airship.
- Robustness: The UAV prototype never freeze or behave in an unsafe way.

6.2 Future developments

To build the next version of the UAV prototype described in this project, several improvements could be implemented as explained below:

- Use a lighter computing platform: the hardware used to implement the computing platform was too heavy to fly onboard of a small aircraft, therefore is critical to use lighter microprocessors like Intel Edison, NVIDIA Jetson TK1, Arduino Nano, Raspberry Pi 2, etc.

- Use a commercial drone: instead of implementing a computing platform onboard a flying platform, a commercial drone like the AR Drone 2.0 with the "Flight Recorder GPS" accessory attached could be used. The AR.Drone already provides an API to monitor and control the drone remotely.
- Improve the web interface: the video streaming should be improved using a higher quality webcam and a faster computing platform. Furthermore the flying data transfer between the web interface and the autopilot should be done over a channel faster than a database. The flying data transfer could be implemented over TCP/IP using technologies like PHP, Ruby on Rails, Django or WebRTC.
- Add new functionalities: besides implementing autonomous and tele-operated flight, many new features could be added such as implement neural networks to recognize images in the webcam video streaming or use an external Android device as a beacon: it could send its GPS coordinates via Wi-Fi to the UAV, which in turn could use this information for flying to or around the Android device.

Bibliography

- [1] Electronic Frontier Fundation, *Surveillance drones*, <https://www.eff.org/issues/surveillance-drones>, January 2015.
- [2] The Telegraph, *German railways deploys surveillance drones*, <http://www.telegraph.co.uk/news/worldnews/europe/germany/10082777/German-railways-deploys-surveillance-drones.html>, January 2015.
- [3] BBC News, *Pilotless police drone takes off*, <http://news.bbc.co.uk/2/hi/6676809.stm>, January 2015.
- [4] The Globe and Mail, *Drone start-ups woo stretched miners for survey work*, <http://www.theglobeandmail.com/report-on-business/industry-news/energy-and-resources/drone-start-ups-woo-stretched-miners-for-survey-work/article9467067>, January 2015.
- [5] Israel Defense, *The Air Mule Takes off*, <http://www.israeldefense.com/?CategoryID=472&ArticleID=543>, January 2015.
- [6] Los Angeles Times, *Radar shows U.S. border security gaps*, <http://articles.latimes.com/2013/apr/03/nation/la-na-border-radar-20130404>, January 2015.
- [7] Airbus Defence & Space, *Collaboration ERDF and Cassidian/Survey Copter*, http://www.defenceandsecurity-airbusds.com/en_US/web/guest/collaboration-erdf-and-cassidian/survey-copter, January 2015.
- [8] BBC News, *Amazon testing drones for deliveries*, <http://www.bbc.com/news/technology-25180906>, January 2015.

- [9] Reuters, *Brazil's stadiums ready for World Cup soccer warm-up in June*, <http://www.reuters.com/article/2013/05/20/us-soccer-brazil-worldcup-stadiums-idUSBRE94J0TT20130520>, January 2015.
- [10] National Geographic, *Cool Gadgets: A.R.Drone 2.0*, <http://voices.nationalgeographic.com/2013/10/07/cool-gadgets-a-r-drone-2-0/>, January 2015.
- [11] MUAC-IREN, <http://www.muac-iren-project.eu/>, January 2015.
- [12] El Confidencial Digital, *El Ejército del Aire tendrá aviones no tripulados. Defensa adjudica la compra de mini UAV Raven por 1,9 millones de euros*, http://www.elconfidencialdigital.com/politica/Ejercito-Aire-Defensa-UAV-Raven_0_1292270761.html, January 2015.
- [13] defensa.com, *El Atlante realiza con éxito el primer vuelo en Europa de un avión no tripulado desde un aeropuerto civil*, http://defensa.com/index.php?option=com_content&view=article&id=10239:el-atlante-realiza-con-exito-el-primer-vuelo-en-europa-de-un-avion-no-tripulado-desde-un-aeropuerto-civil&catid=54:espana&Itemid=162, January 2015.
- [14] Escuela Industriales UPM, *Un grupo de investigación de altos vuelos*, http://www.escuelaindustrialesupm.com/escuela-industriales-upm/un-grupo-de-investigacion-de-altos-vuelos/#.VLQy29_081U, January 2015.
- [15] FUVÉ, *Future Vehicles and Entrepreneurs Association*, <http://fuve-english.jimdo.com/fuve-e/>, January 2015.
- [16] jderobot.org, *Livio Calvo-pfc-itis*, http://jderobot.org/index.php/Livio_Calvo-pfc-itis, January 2015.
- [17] svn.jderobot.org, <https://svn.jderobot.org/users/kenshiro/pfc-itis/>, January 2015.
- [18] jderobot.org, *Livio Calvo-pfc-itis*, http://jderobot.org/index.php/Livio_Calvo-pfc-itis#Test_Board_v2, January 2015.

- [19] jderobot.org, *Livio Calvo-pfc-itis*, http://jderobot.org/index.php/Livio_Calvo-pfc-itis#First_Prototype, January 2015.
- [20] Free Software Foundation, *GNU General Public License*, <http://www.gnu.org/licenses/gpl-3.0-standalone.html>, January 2015.