



Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias de Informação

IoT Enabled Aquatic Drone for Environment Monitoring

João Ricardo Baptista de Matos

Dissertação submetida como requisito parcial de obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador(a):

Professor Octavian Adrian Postolache,
ISCTE-IUL

Co-orientador(a):

Professor Anders Lyhne Christensen,
ISCTE-IUL

Outubro de 2016

ABSTRACT

This thesis presents a platform that tackles environment monitoring by using air and water quality sensors to provide data for the user to know what is happening in that surveilled area. The hardware is incorporated in a sensing module in order to be used with an Unmanned Surface Vehicle (USV).

It presents a monitoring system based on Raspberry Pi platform and a multichannel sensing module associated with water quality and air quality measurement parameters. Thus, the temperature, relative humidity and gas concentration are measured as well as the underwater acoustic signals using a hydrophone. The data is stored on the memory of the drone's computational platform (Raspberry Pi), and synchronized with a remote server database. Audio streaming capabilities were implemented in the server side. Additionally, a mobile application was developed to be used by people working in the field for data visualization, audio streaming playback and statistical analysis (by showing plotted data).

Keywords: Drones; USV; Sensors; Hydrophone; Raspberry Pi; Streaming; Android

RESUMO

O intuito desta dissertação é apresentar uma plataforma de monitorização ambiental através da instalação de sensores de qualidade do ar e da água de forma a fornecer dados ao utilizador daquela área vigiada. O hardware é apresentado num módulo onde estão presentes todos os componentes por forma a poder ser usado num drone aquático.

É apresentado um sistema de monitorização baseado no sistema de processamento Raspberry Pi e um módulo multicanal de sensores de medição de qualidade do ar e qualidade da água. Sensores esses de medição da temperatura, humidade relativa e concentrações de gases tal como a medição de sinais de áudio debaixo de água com o uso de um hidrofone. Os dados estão alojados na memória do sistema computacional do drone (Raspberry Pi) e estão sincronizados com uma base de dados remota alojada num servidor cloud. Um sistema de streaming de áudio foi também implementado do lado do servidor. Adicionalmente, foi desenvolvida uma aplicação móvel que permite visualizar os dados provenientes dos sensores, reproduzir a stream de áudio e também análise de estatísticas (com apresentação gráfica dos dados).

Palavras-chave: Drones; USV; Sensores; Hidrofone; Raspberry Pi; Streaming; Android

ACKNOWLEDGMENTS

Firstly, I would like to thank Professor Octavian Postolache for all the hard work in providing all the materials, the workspace and all the help in the making of the thesis in general. Even with busy schedules, always showed availability to help in whatever he could.

Would like to thank Professor Anders Christensen and the BioMachines lab for providing a Raspberry Pi 2 and some sensors right in the beginning that allowed me to fully explore, learn and test on the Raspberry that enabled me to work more flawlessly in the upcoming months. Also for providing an aquatic drone in order to do the field tests.

To the lab technician Hugo Silva for the hard work in making the final circuit board that made the live tests possible. For all the persistency and patience in having the final product fully working, thank you Hugo.

To my friends and fellow Master's colleagues Nuno Santos, Nuno Pardal and Rui Rosa. We spent a lot of time together doing our own dissertations with continuous inputs, opinions and ideas that made this journey easier.

To my friends Tiago Saraiva, André Silvério and Diogo Ferreira for showing concern, providing inputs and ideas whenever I got stuck with issues.

FIGURES

Figure 1 - BeagleBone Black	7
Figure 2 - Arduino Uno	8
Figure 3 - Arduino Mega.....	9
Figure 4 – Raspberry Pi 3.....	10
Figure 5 - Cirrus Logic card installation	11
Figure 6 - Cirrus Logic Audio Card and the possible connections [21]	12
Figure 7 - Plantronics USB Audio Adapter	13
Figure 8 – Sea-Phone Hydrophone [22].....	13
Figure 9 - SQ26-H1B Hydrophone [23]	14
Figure 10 – Digital Thermometer DS18B20 [24]	15
Figure 11 - LM35	15
Figure 12 - HIH4000	16
Figure 13 - Structure of the TGS2600.....	17
Figure 14 - TGS2600 gas sensor [27]	17
Figure 15 – Sensing element of the TGS800 [28].....	18
Figure 16 - TGS800 gas sensor	18
Figure 17 - Inside structure (cut view) of TGS800 [29]	18
Figure 18 - Adafruit ADS1015 12bit ADC.....	19
Figure 19 - Structure of the MCP3004 [31]	19
Figure 20 - MCP3004.....	20
Figure 21 - Cellular standards comparison	24
Figure 22 - Hardware system architecture overview	27
Figure 23 - Circuit schematics	28
Figure 24 - HIH 4000 connections	29
Figure 25 - LM35 connections	30
Figure 26 - Adafruit's Ultimate GPS Breakout v3	30
Figure 27 - Hydrophone connected to the USB audio card	31
Figure 28 - Raspberry Pi 3 with the components connected.....	32
Figure 29 - Voltage Divider	32
Figure 30 - ZTE MF910 LTE Mobile Router	33
Figure 31 - Software system architecture.....	35
Figure 32 - Data acquisition script flow chart.....	37
Figure 33 - Signals script flow chart	38
Figure 34 - Raspberry signalling scheme	39
Figure 35 - Data acquisition python functions	39
Figure 36 - Code snippet of the sensors' thread	40
Figure 37 - Code snippet of the signals' script	41
Figure 38 - Databases' structure	42
Figure 39 - Example of an NMEA message	43
Figure 40 - cgps output (GPSd).....	43
Figure 41 - FFmpeg output.....	44
Figure 42 - Code snippet of FFT (android)	45
Figure 43 - FFT data output example.....	46
Figure 44 - flow chart for the full data returning scripts	47

Figure 45 - Flow chart for the latest data returning script.....	48
Figure 46 - Flow chart for the signal triggering script	49
Figure 47 - PHP signals' script code snippet	49
Figure 48 - PHP latest value script code snippet.....	50
Figure 49 - PHP all data returning script code	51
Figure 50 - JSON output example.....	51
Figure 51 - Wowza Streaming Engine main screen	52
Figure 52 - Main Activity flowchart	53
Figure 53 - Main Activity screen shot.....	54
Figure 54 - Graphs activity flowchart	55
Figure 55 - Sensors' data in graph format	56
Figure 56 - FFT data Activity flowchart	57
Figure 57 - FFT audio data Activity.....	58
Figure 58 - Audio waveform (time domain)	59
Figure 59 - Android code snippet of data receiving interface method.....	60
Figure 60 - Android code snippet get data method	61
Figure 61 - AsyncTask code snippet	61
Figure 62 - JSON data parsing and plotting	62
Figure 63 - Android fft graph code snippet.....	63
Figure 64 - USV components.....	65
Figure 65 - USV with the components installed plus mobile application.....	65
Figure 66 – Deploying the USV in the lake	66
Figure 67 - USV deployed	66
Figure 68 - USV test trajectory (start red circle; finish yellow circle).....	67
Figure 69 - Main screen after the test.....	68
Figure 70 - Audio waveform in the time domain.....	69
Figure 71 - Audio FFT data from the tests	70
Figure 72 - Audio FFT with wider spectrum	71
Figure 73 – Relative humidity values (%)	72
Figure 74 - Temperature values (°C).....	73
Figure 75 - Gas sensor values	74

TABLES

Table 1 - BeagleBone original and BeagleBone Black specifications [17]	7
Table 2 - Arduino Uno and Mega specifications [18].....	8
Table 3 - Raspberry Pi 3 specifications [20]	10
Table 4 - Mobile OS comparison	22
Table 5 - Wi-Fi popular standards comparison.....	23

ACRONYMS & ABBREVIATIONS

IoT	Internet of Things
USV	Unmanned surface vehicle
US	Unmanned Systems
UV	Unmanned Vehicles
GPS	Global Positioning System
DSR	Design Science Research
DSRMPM	Design Science Research Methodology Process Model
ICCPCT	International Conference on Circuit, Power and Computing Technologies
GUI	Graphical User Interface
CPU	Central Processing Unit
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
GPIO	General-purpose input/output
GNSS	Global Navigations Satellite System
LTE	Long-Term Evolution
OS	Operating System
IDE	Integrated Development Environment
ADT	Android Development Tools
API	Application Program Interface
AVD	Android Virtual Device
UMTS	Universal Mobile Telecommunications System
HSPA	High Speed Packet Access
LTE-A	Long-Term Evolution – Advanced

CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iv
FIGURES	v
TABLES	vi
ACRONYMS & ABBREVIATIONS	vii
CONTENTS	viii
1	1
1.1 Overview	2
1.2 Motivation	3
1.3 Context	3
1.4 Objectives	5
1.5 Literature Review	6
1.5.1 Computational Platforms	6
1.5.1.1 BeagleBone	6
1.5.1.2 Arduino	7
1.5.1.3 Raspberry Pi 3	9
1.5.2 Audio	11
1.5.2.1 Cirrus Logic Audio Card	11
1.5.2.2 USB Audio solution	12
1.5.2.3 Plantronics USB Audio Adapter	12
1.5.2.4 SS03-10 Sea-Phone Hydrophone	13
1.5.2.5 SQ26-H1B Hydrophone	13
1.5.3 Temperature Sensors	14
1.5.3.1 Waterproof Digital Thermometer DS18B20	14
1.5.3.2 LM35 Temperature Sensor	15
1.5.3.3 HIH 4000 Humidity Sensor	16
1.5.4 Gas sensors	16
1.5.4.1 TGS2600	16
1.5.4.2 TGS800 Gas Sensor	17
1.5.5 Analog to Digital Converters	18
1.5.5.1 Adafruit ADS1015 12-Bit Analog to Digital Converter (ADC)	19
1.5.5.2 Microchip MCP3004	19

1.5.6	GPS	20
1.5.7	Mobile application platform.....	20
1.5.7.1	Android	21
1.5.7.2	iOS	21
1.5.7.3	Windows Phone	22
1.5.8	Wireless Communication	23
1.5.8.1	Wi-Fi.....	23
1.5.8.2	Cellular Network.....	23
1.6	Dissertation structure.....	25
2	26
2.1	Hardware Overview.....	27
2.2	Adafruit ADS1015 Analog to Digital Converter.....	28
2.3	Analog Sensors	29
2.3.1	HIH 4000.....	29
2.3.2	LM35.....	29
2.4	Ultimate GPS Breakout v3	30
2.5	USB Audio Card & Sea-Phone Hydrophone.....	30
2.6	Raspberry Pi.....	31
2.7	Conditioning circuits.....	32
2.8	Network communication.....	33
2.8.1	ZTE MF910 LTE Mobile Router	33
3	SOFTWARE.....	34
3.1	Software Overview	35
3.2	Embedded Platform Software	35
3.2.1	Python	36
3.2.2	MySQL	36
3.2.3	FFmpeg.....	36
3.2.4	DDclient.....	36
3.2.5	Python Scripts.....	36
3.2.6	Python Scripts Code Highlights	39
3.2.7	MySQL Database	41
3.2.8	GPS Software.....	42
3.2.9	Audio Stream.....	43
3.2.9.1	Fast Fourier Transform (FFT).....	44
3.3	Server Side Software	46
3.3.1	PHP	46

3.3.2	Wowza.....	46
3.3.3	PHP Scripts.....	47
3.3.4	PHP Scripts Code Highlights	49
3.3.5	JSON Format.....	51
3.3.6	Wowza Streaming Service	51
3.4	Android Application.....	52
3.4.1	Main screen.....	52
3.4.2	Sensors' graphical data	54
3.4.3	Audio FFT activity	56
3.4.4	Audio Waveform (time domain) sound activity.....	58
3.4.5	Mobile Application Code Highlights	59
4	RESULTS	64
4.1	Field Tests	65
5	CONCLUSIONS AND FUTURE WORK	75
5.1	Conclusion	76
5.2	Future Work.....	76
	REFERENCES.....	A
	ANNEX.....	E

1 INTRODUCTION

1.1 Overview

With today's advancements in technology, it became possible to address a larger number of problems regarding accessibility and also monitoring and interacting more thoroughly with systems. This work presents the design and implementation of a system that can be used for water quality monitoring, the system being designed especially for monitoring tasks on rivers and estuaries to help maintain sustainability. The possibilities are numerous of what can be done with a USV, it can help doing maritime tasks using multi-drone swarms [1], it can address more concerning events such as cleaning oil spill [2], or for general environmental condition monitoring.

The Internet of Things help to solve a lot of problems in different fields: in smart cities, IoT applications are related with parking issues, noise, traffic, illumination monitoring [3]; emergency systems for earthquakes [4]; precision agriculture applications in culture process optimization [5]. IoT are used to deliver information from the sensors and to the actuators.

The goal of this thesis is to present an aquatic drone setup equipped with a Raspberry Pi that is connected to an array of sensors for air and the water quality monitoring. For air quality measurements, temperature, humidity and gas sensors were considered. For water quality monitoring, hydrophone sensing unite were implemented in order to measure underwater noise pollution related with human activities in rivers, estuaries and seas. The system is prepared to include additional sensors for water quality monitoring such as turbidity, DO, conductivity and pH however in this first phase considering the budget limitation and the current needs only the above-mentioned measurement channels were implemented.

Several tasks were carried out during the system implementation. Thus, the sensors interfacing with Raspberry Pi platform was considered and different signal conditioning modules were implemented. Referring to the system software, Python scripts were developed in order to read the values from the sensors and to write them to a local database. A server setup was carried out to receive the data from the local database implemented on Raspberry Pi level using database replication, in which the server database will always update whenever there are changes in the sensor readings. In order to be possible to access and visualize the data in an appropriate GUI an Android application for mobile device was developed. The applications fetch the data from the server database and provide live readings for the system user.

1.2 Motivation

The ‘Internet of Things’ is not an unknown term nowadays as it continues to grow in popularity. A lot of companies are working on new devices and new solutions to deal with the growth of the connected devices as this industry continues to develop. It carries the tools that can transform every industry such as: agriculture, transportation, manufacturing, housing and even entire cities. Companies continue to work on making new devices to address in the digitalization of these industries but also improving communication protocols with security being a top priority. It is estimated that around 2015 10 billion devices were connected and by 2020 will be connected 20 to 30 billion devices, the main reason being the costs continue to drop and demand continues to grow [6].

The need for a good ecosystem and its preservation is important to maintain a sustainable environment and it requires continues searches for new ideas to promote a clean planet. The oceans are the home of thousands of different species that are important for the ecosystem, the ocean also provides more than half of the oxygen in the atmosphere and also absorbs the most carbon from it. Excess human activity such as the degradation of marine habitats and species, overharvest of resources present a danger to the marine life and a good way to fight this problem is by protecting areas with monitoring and conservation [7].

On the other hand, the rivers provide a rich source of fresh water and nutrients required by animals and also by humans. In many areas rivers are used to water crops, fishing and also transportation. It is also the main source of water of wild animals that ensure their survival with a sustained supply of fresh water and food. More than the oceans, the rivers should be preserved to maintain and guarantee a good ecosystem [8].

Another problem that should not be ignored and must be monitored as well is the underwater noise. With the vast amount of human activity in the sea with submarines, ships equipped with sonar devices, even natural causes such as earthquakes can disrupt fish migrations, communication and even their food hunting [9]. With this system it is hoped to provide monitoring of air and water quality factors in order to help fight these problems.

1.3 Context

Unmanned Systems (US) are devices or any sort of machines that do not require human intervention to perform actions. It consists of sensors, processors, controls and communication systems to exchange data with an operator. The operator can then input a set of actions for the

US to do as well as receive data from the US and make decisions upon that [10]. These US can also be categorized as Unmanned Vehicles (UV) and are divided through different types:

- Unmanned Ground Vehicle (UGV)
- Unmanned Aerial Vehicle (UAV)
- Unmanned Surface Vehicle (USV)
- Unmanned Undersea Vehicle (UUV) [11]

In this project, a USV will be used. A USV is a boat type vehicle that operates on the surface of the water, in order words, an aquatic drone. Nowadays the USVs are mostly used for military service such as training purposes, transporting supplies, equipped with weapons for action, for scouting activity, etc. USVs are also used for research purposes for environmental control and this project will assist in this direction.

Since various types of smart sensors will be connected to it, there will be a whole new dimension of actions that the USV will manage to do. It is possible for the USV to make decisions of its own regarding the programming behind it. For example, if a swarm of USVs is scouting an area and they are equipped with hydrophones they can move towards where the sound is originating from (for a set of specific frequencies), being useful to chase a shoal of fish or any other thing. Or, another example, if one boat in the swarm detects weird sounds (i.e. a frequency different than normal) coming from another boat it can let the user know and perform maintenance upon the receiving of that information. There are innumerable actions that can be made using smart sensors and can be explored to the max.

The BioMachines Lab in Institute of Telecommunications in ISCTE-IUL, Lisbon is developing a swarm of aquatic drones designed to perform maritime tasks in the sea with an ad-hoc network architecture via Wi-Fi. Each drone is equipped with a compass, GPS and a Raspberry Pi 2 unit with a Wi-Fi interface and they form a distributed network (controlled by an artificial neural network based controller) without a central coordination, they keep broadcasting their position to the drones in their neighborhood every second. The neural network performs actions depending on what it receives, the sensor readings feed the neural network and then it will control the drone based on the information received [12].

Reported research at the International Conference on Circuit, Power and Computing Technologies (ICCPCT) 2015 consisted of developing of a static system that includes Raspberry Pi B+ model connected to some extra sensors such as: temperature sensor. The

Raspberry is part of an IoT module that will send the data to a cloud server, the cloud server then sends the data to a web server with a user interface [13].

Another interesting research using buoys composed by android phones as an electronic device for sensing and sending information, using the phones' GPS to report the position and also meteorological data such as wind speed and direction. The buoys track oil spilt areas without human intervention [14].

Another project done in the United States, the Water on the Web (WOW) gathers information from lakes and rivers across the different states of the country, it uses a RUSS (Remote Underwater Sampling Station) which is a floating platform equipped with conductivity, dissolved oxygen, pH, temperature and turbidity sensors and it uses cellular networks to transmit the data to the website [15][16].

This thesis has similarities with the above-mentioned projects, an aquatic drone will be used but instead of using Wi-Fi connection, a cellular solution will be adopted to eliminate the distances problem, and with today's strong infrastructures of 3G and 4G the latency will not be considered as an issue. A Raspberry will be used with similar sensors as described in the second project to monitor the water quality but also to perform statistics with the data and it will feature a local database that will keep feeding a remote server with data to be then consulted by a GUI application developed for mobile. Like the WOW project, this thesis will focus on lakes and rivers but also estuaries.

1.4 Objectives

The main goal is to build a water and air quality-monitoring module for aquatic drones using Raspberry Pi platform connected to a wide array of sensors for data gathering to monitor the area. Sensors for temperature readings, humidity, pH, salinity and conductivity levels, also underwater noise. Mostly air and water quality sensors. The connection is done by Wi-Fi or 3G/4G but mostly 3G/4G since large distances are dealt with. The thesis contents is expressed by following parts:

- Air and water quality sensors connected to the Raspberry computation platform that have python scripts to control the acquisition, to process the data and to upload the data to a local database on the SDcard.
- Configured and developed server with a database (replication) that receives data from the Raspberry Pi.

- 3G/4G communication between the Raspberry and the cloud server and vice-versa.
- Mobile front end application in Android that fetches the sensors' data from the server. It obtains sensors' values and the states. Can also provide statistics.

1.5 Literature Review

1.5.1 Computational Platforms

In this section is discussed the several embedded systems considered for the core component of the project.

1.5.1.1 BeagleBone

Beaglebone is an embedded open source computer developed by Texas Instruments. The board is meant for educational purposes and people in general that which to experiment and create something using this small affordable computer. It can run several distributions of Linux OS and Android. There are several versions of BeagleBone boards but only two were considered: BeagleBone Black and the original BeagleBone. The hardware of both boards is described in the following table:

	BeagleBone Black	BeagleBone (original)
Processor	AM3358 ARM Cortex-A8 1GHz	AM3358 ARM Cortex-A8 720MHz
Analog Pins	7	7
Digital Pins	65 (3.3V)	65 (3.3V)
Memory	512MB DDR3	256MB DDR2
Connectivity	miniUSB 2.0 client port, USB 2.0 host port, Ethernet, HDMI	miniUSB 2.0 client port, USB 2.0 host port, Ethernet
Video	microHDMI, cape add-ons	cape add-ons
Audio	microHDMI, cape add-ons	cape add-ons
Supported Interfaces	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus,

		4 Timers, FTDI USB to Serial, JTAG via USB
MSRP	\$49	\$89

Table 1 - BeagleBone original and BeagleBone Black specifications [17]

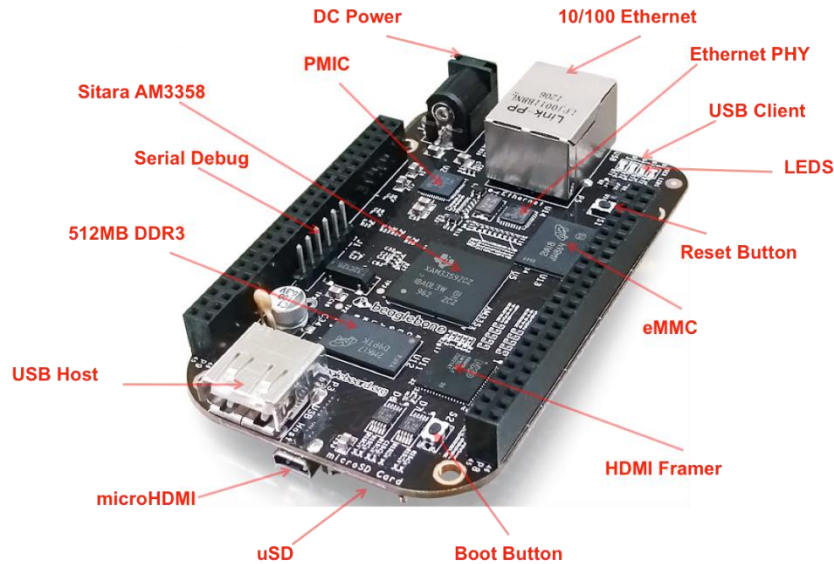


Figure 1 - BeagleBone Black

Out of the two boards, the choice would go to the BeagleBone Black as it has more memory and a slightly faster processor, the HDMI interface is not important for the project and it has less interfaces than the original but those are also not important. The most important interfaces for the project are the UART and I2C and the Black model already include those and is cheaper. A big downside is the lack of Wi-Fi support which is crucial for the communication.

1.5.1.2 Arduino

Arduino is an open-source hardware platform that is able to read inputs (sensors or other type of hardware connected to it) and trigger an action/output on hardware or software level. It can be programmed by writing a set of instructions using the Arduino programming language and the Arduino IDE to write and upload the code, available for Windows/Mac/Linux. There are several boards available but only two were considered, the Uno and Mega:

	Arduino Uno	Arduino Mega
Processor	ATmega328P 16MHz	ATmega2560 16MHz

Operating/Input Voltage	5 V / 7-12 V	5 V / 5-12 V
Analog In/Out	6/0	16/0
Digital IO/PWM	14/6	54/15
EEPROM [kB]	1	4
SRAM [kB]	2	8
Flash [kB]	32	256
USB / UART	Regular / 1	Regular / 4
MSRP	€20	€35

Table 2 - Arduino Uno and Mega specifications [18]

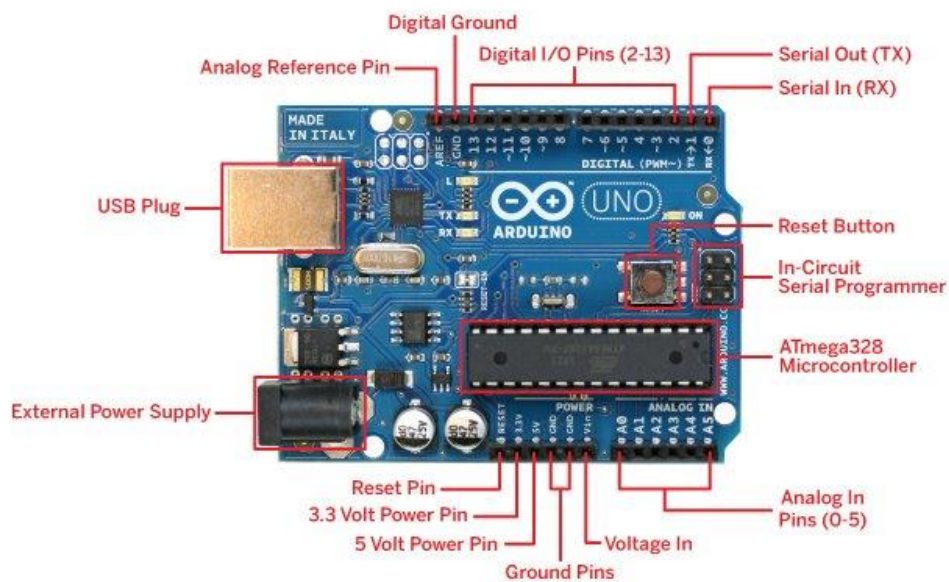


Figure 2 - Arduino Uno

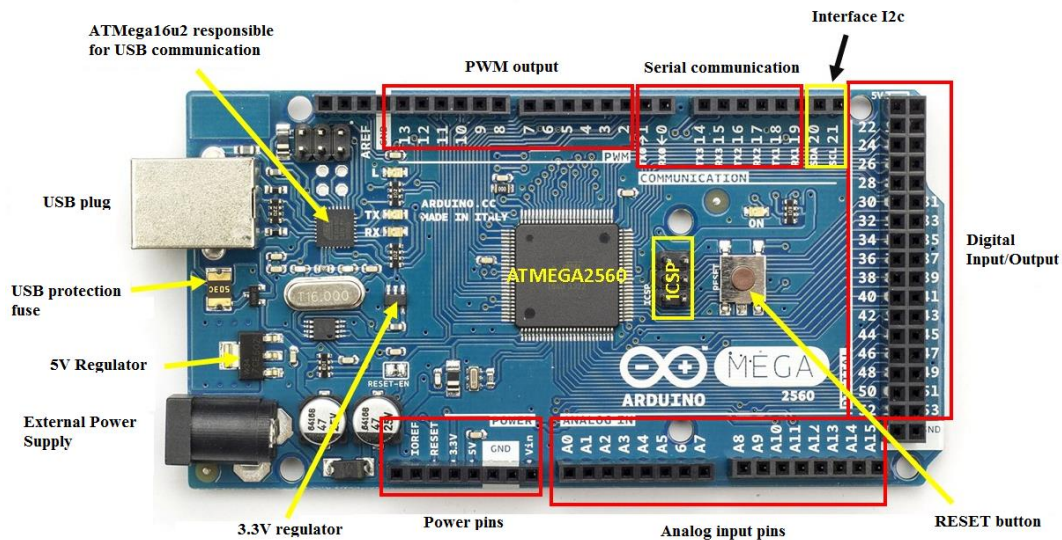


Figure 3 - Arduino Mega

The use of Arduino in this project is not viable due to several limitations. To ensure that the system would work, it would necessary to install external shields for the Wi-Fi and GPS but at the end would not look friendly and come out as even more expensive. The streaming would be possible but the performance would drop with so many things working at once and for the similar price, there are better alternatives for this project's environment.

1.5.1.3 Raspberry Pi 3

The Raspberry Pi is a very small and compact computer manufactured for educational purposes. It is basically a computer with a size of a credit card, it possesses an ARM CPU, graphics capabilities, USB ports, Ethernet port, HDMI port, programmable ports (GPIO) and a combined audio and composite video jack. Its characteristics make it ideal to use in the environment of this project, it is possible to connect all the sensors needed and to communicate with a server to keep uploading data. It also does not require much power (5V, ~1A), allowing it to be attached to a rather small battery for mobile purposes. There are several models of Raspberry Pi, the latest being the Raspberry Pi 3 that was considered for the project [19][20].

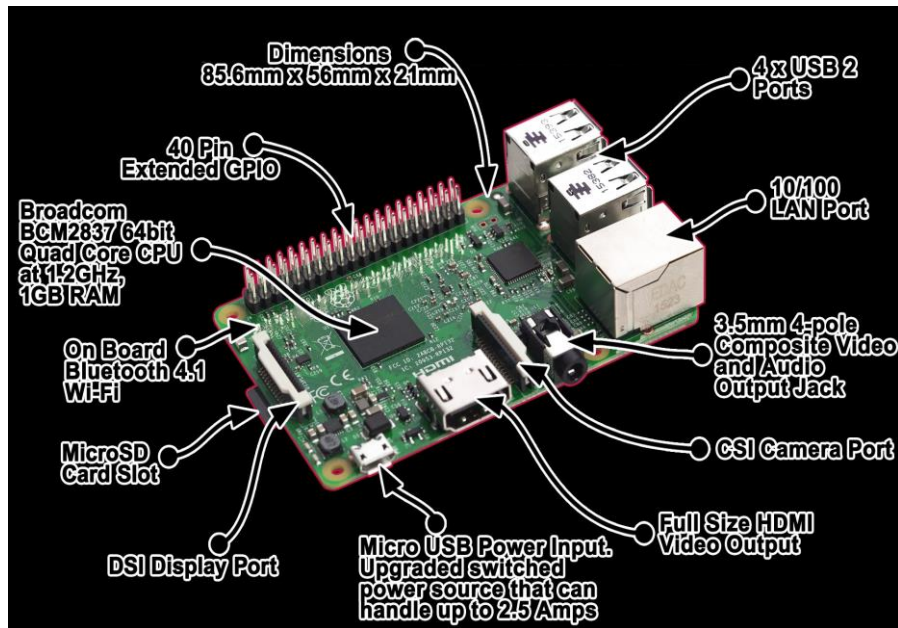


Figure 4 – Raspberry Pi 3

Raspberry Pi 3	
Processor	BCM2837 1.2GHz 64-bit Quad-Core ARMv8
Memory	1GB SDRAM
Connectivity	4x USB 2.0 host port, Ethernet, HDMI, Wi-Fi 802.11n, Bluetooth 4.1
Video	Full HDMI port
Audio	HDMI, 3.5mm audio jack
Supported Interfaces	40 GPIO pins with I2C, SPI, 1-Wire, UART, Digital input/output, JTAG, PCM, DPI, GPCLK (General purpose clock),
MSRP	\$35

Table 3 - Raspberry Pi 3 specifications [20]

The Raspberry Pi is the overall best candidate, it has a quite powerful CPU and ram size, built in Wi-Fi which is very important, I2C and UART interfaces, Linux software, more USB ports

and with the best price considering all the hardware it has. The only downside is the lack of audio input interface but it does not present much of a problem as the Raspberry has a lot of support for external components and getting an analog interface is not hard.

1.5.2 Audio

In order to capture audio from the microphone, an external audio card is necessary to provide audio input. In this section, the audio cards are analyzed.

1.5.2.1 Cirrus Logic Audio Card

Although the Raspberry Pi is equipped with audio capabilities, it does not allow for a crucial factor needed for this project, which is audio input. This module designed specifically for the Raspberry not only has much better audio performances (in both analog and digital forms) but also allows for audio input which will be needed to process audio. Although this card has really high quality, its installation is not ideal as it will occupy all of the GPIO ports of the Raspberry, like in the following image:



Figure 5 - Cirrus Logic card installation

As it is clear in the image, this card will not let anything else connect to the GPIO ports of the Raspberry, which would make the project undoable.

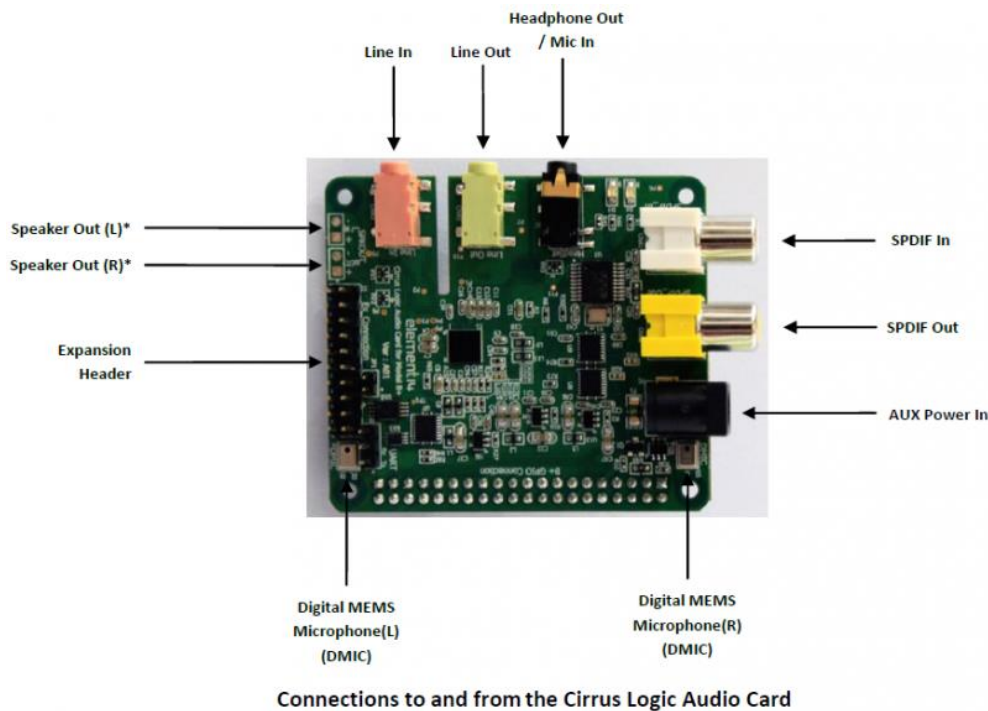


Figure 6 - Cirrus Logic Audio Card and the possible connections [21]

1.5.2.2 USB Audio solution

At first a more complete solution was thought to assure audio input capabilities and a more reliable sound quality such as the Cirrus Logic Audio card, but it came to a conclusion that its implementation style (shield onto the GPIO ports) on the Raspberry Pi would raise some difficulties on connecting the rest of the components so it was decided to migrate to a simpler solution and an audio USB card was chosen. Since Raspberry is running a Linux kernel, it is important to look for a card that has a driver-free chip, in other order just plug and play. There are just a few cards like these in the market so the choice is not difficult. There is no point in comparing these sorts of cards because they all feature similar chips and look practically the same (some with bigger size than others).

1.5.2.3 Plantronics USB Audio Adapter

This Plantronics USB audio adapter features a C-Media chipset which does not required additional drivers to work and it has full support for arm-based Linux distributions, making it ideal to work with a Raspberry Pi, has a rather good quality and most importantly allows for audio input.



Figure 7 - Plantronics USB Audio Adapter

1.5.2.4 SS03-10 Sea-Phone Hydrophone

A hydrophone is basically an underwater microphone; it will be used to listen to all types of underwater noises such as animals and derivatives from human activity with a frequency range from 0.020 to 50 kHz. This model is powered by a 9V battery that can last up to 18 hours and gathers the audio through a 3.5mm stereo jack connector that will be used to connect to the audio card in the Raspberry. It also allows to adjust the gain from 0 to 30dB.



Figure 8 – Sea-Phone Hydrophone [22]

1.5.2.5 SQ26-H1B Hydrophone

This hydrophone is a more recent model and also suited for general purpose audio-band signal detection. It is a little bit smaller than the SS03-10 and has a slightly smaller frequency range, going from 0.020 to 45kHz and can be paired with a Bluetooth speaker system. The battery is longer, going up to 34 hours of life but the gain and the jack connector are the same as the SS03-10.



Figure 9 - SQ26-H1B Hydrophone [23]

Although the SQ26-H1B is more recent and has some extra features and improvements, it is a little bit more expensive than the SS03-10 and those extra features do not bring anything useful considering the context of this project. The extra battery time could be useful but the tests never last 18 hours straight so the extra time is not needed. The older hydrophone was used instead and serves the purpose just as good.

1.5.3 Temperature Sensors

For temperature sensing, analog and digital models were considered and the next few paragraphs talk about them.

1.5.3.1 Waterproof Digital Thermometer DS18B20

This temperature sensor provides temperature readings from -55°C to 125°C with a $\pm 0.5^{\circ}\text{C}$ accuracy from -10°C to $+85^{\circ}\text{C}$. It uses the 1-Wire protocol for data communication which means that uses only 1 wire to exchange data with the Raspberry Pi and in addition, it can also draw power from that same line due to its feature called “parasite power” eliminating the need for an external power supply, making it really simple to setup. Each DS18B20 sensor also has a unique 64-bit code which allows to setup multiple installations to work on the same 1-Wire bus. The sensor also includes an integrated EEPROM that reduces external component count.



Figure 10 – Digital Thermometer DS18B20 [24]

1.5.3.2 LM35 Temperature Sensor

The LM35 is a compact temperature sensor that can be powered by 5V supplied by the Raspberry but it can operate from 4V to 30V, it provides an accuracy of 0.5°C and I can go from -55°C to 150°C. The sensor is already calibrated directly in Celsius and it provides an output voltage linearly proportional to the centigrade temperature [25].

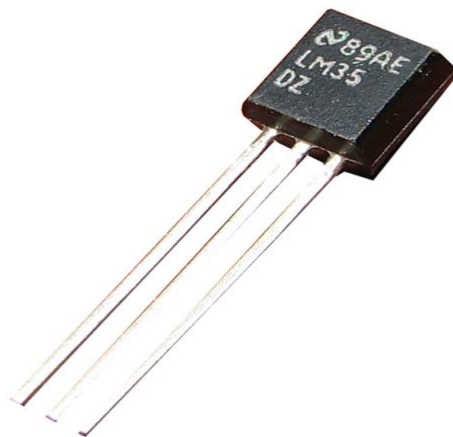


Figure 11 - LM35

Comparing the two temperature sensors, the LM35 came as the preferable choice. Its analog interface makes it easier to deal with as it does not need any drivers and it is cheaper.

1.5.3.3 HIH 4000 Humidity Sensor

This humidity sensor will provide the percentage value of the relative humidity present in the air with the necessary conditioning, it provides the values in voltage has an enhanced accuracy, fast response times and it requires low power [26], ideal for our work environment.

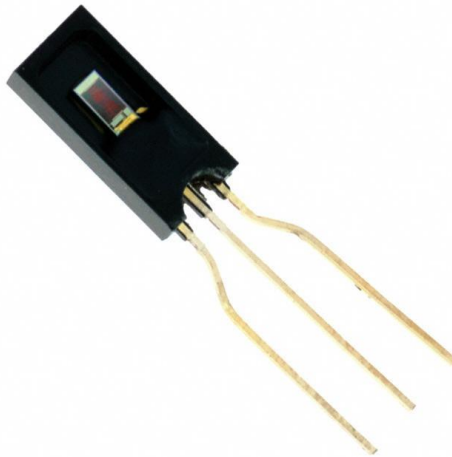


Figure 12 - HIH4000

1.5.4 Gas sensors

In the gas sensing electronics, the search must be done by the target gas(es) and since for this project outside air is the target environment, the results will be better if a sensor designed for air quality control is selected. Inside air contaminants, there are not many options to go for, two were considered, the TGS2600 and TGS800 both from Figaro.

1.5.4.1 TGS2600

The TGS2600 is the latest model from Figaro for air quality control, it is comprised of a metal oxide semiconductor layer formed on an alumina substrate with an integrated heater (which is common in gas sensing electronics). The sensor reacts to the gas by increasing its conductivity depending on the gas detected. It features low power consumption, high sensitivity to air gas contaminants, small size and has a simple electrical circuit. Its structure is showed in the following figure:

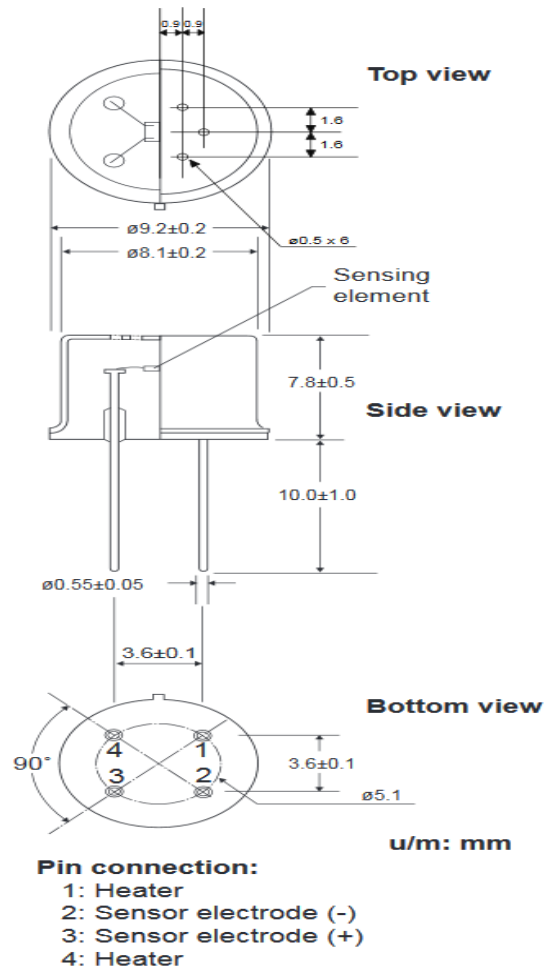


Figure 13 - Structure of the TGS2600

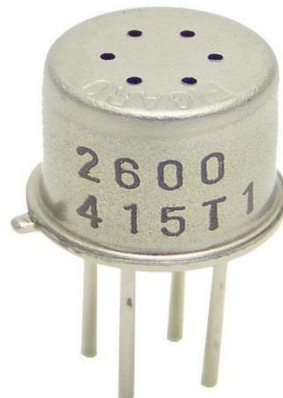


Figure 14 - TGS2600 gas sensor [27]

1.5.4.2 TGS800 Gas Sensor

The TGS800 serves the same purpose as the TGS2600 and has a similar construction but it is an older version and, for that reason, it is cheaper. The sensor has a heater in an alumina ceramic tube and the semiconductor material is mounted on the tube, like in the following picture:

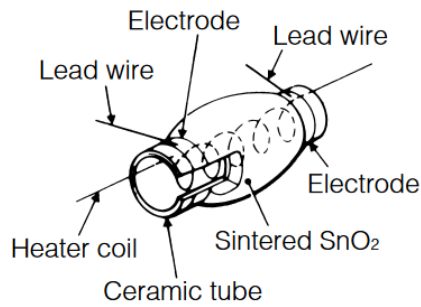


Figure 15 – Sensing element of the TGS800 [28]

The sensor is then housed by a resin base and cover painted in red, like in the figure 15:



Figure 16 - TGS800 gas sensor

The overall inner structure of the sensor looks like this:

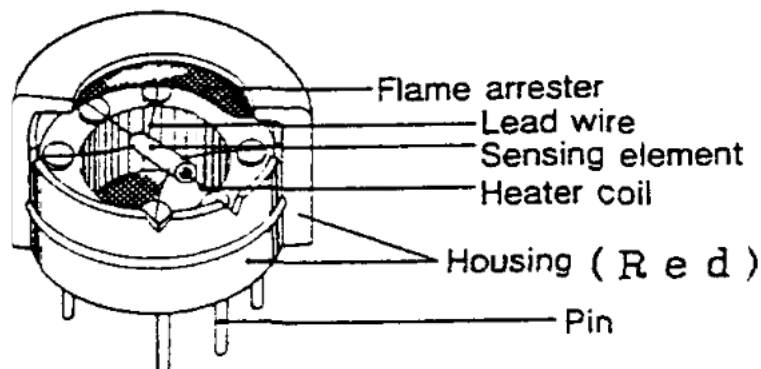


Figure 17 - Inside structure (cut view) of TGS800 [29]

The choice would go to the TGS800 due to solely being cheaper, as having the latest model TGS2600 does not present that much of an improvement to justify spending extra.

1.5.5 Analog to Digital Converters

Since the Raspberry Pi lacks analog support and most of the sensors are analog, there is a need for an external component where the sensors will connect to. With Raspberry support, two options were considered, the ADS1015 and the MCP3004.

1.5.5.1 Adafruit ADS1015 12-Bit Analog to Digital Converter (ADC)

This Adafruit model has an I2C communication interface that can be connected directly to the Raspberry through the GPIO ports with proper configuration and it is easy to use. This ADC is powered by 5V and has 4 analog input ports (A0-A4) on which the analog sensors will be connected. It also features a PGA (Programmable Gain Amplifier) which allows to tune for more precision for sensors that have lower voltage readings. This model has a really good support for the Raspberry Pi, Adafruit has developed a library in python to aid in the development. With this hardware it is possible to work with the analog sensors that are needed for the project. [30].

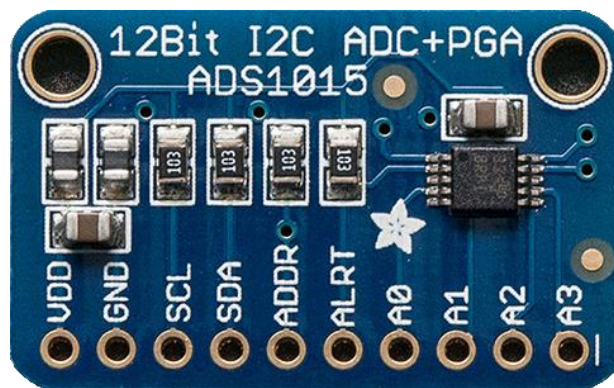


Figure 18 - Adafruit ADS1015 12bit ADC

1.5.5.2 Microchip MCP3004

The MCP3008 is an ADC featuring the SPI interface with a 10-bit resolution, it also has 4 analog input ports and is powered by 5V. It does not have an amplifier and has a low power CMOS technology. This ADC has no specific support for the Raspberry, but it works as it has an SPI interface. The structure of this ADC is the following:

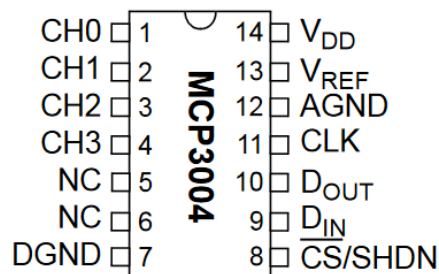


Figure 19 - Structure of the MCP3004 [31]



Figure 20 - MCP3004

Between these two ADC's, the ADS1015 from Adafruit was the preferable choice as it offers a more specific support for the Raspberry Pi with a custom-made library developed by Adafruit. The MCP3004 being SPI, it could present a solution with better performance but the implementation of the SPI without specific support for the Raspberry would bear a considerable amount of work. The speed, being the main difference between both interfaces [50], it would not be a problem in this project, as the I2C interface suffices for the scenario and has a simpler implementation.

1.5.6 GPS

GPS is a type of Global Navigations Satellite System (GNSS) to determine something's position on earth in coordinates, it uses trilateration to calculate the exact position. The GPS receiver must receive signal from at least 3 satellites to determine the 2D (latitude and longitude) position plus track movement and 4 or more satellites to determine the 3D position (latitude, longitude and altitude). Other information can be provided such as speed, bearing, track, trip distance, distance to destination, sunrise and sunset time and more. A unit is connected to the Raspberry Pi in order to know the position of the USV at all time, can also be used to interact with the smart sensors to execute actions in certain locations [32].

1.5.7 Mobile application platform

For the mobile development platform, there are some important factors to be concerned about: the market share, development costs, licenses and OS family. The next few paragraphs compare the three most used mobile platforms in the market and which has the most benefits for the project.

1.5.7.1 Android

Android is a Unix based operating system currently developed by Google to compete against iOS, Symbian and Windows Phone but with the difference that it is open source which is a must for programming enthusiasts to be able to develop the best performance-wise applications. It uses Linux kernel and is designed for smartphones, tablets and more recently wearables, TV and automobiles. This OS has been increasing in popularity in the past years becoming a market leader leaving iOS in second place. Android is an excellent option to develop a GUI front end application for data reading and statistics processing, it also has direct connections with other Google products such as Google maps which enrich development with the assist of these tools [33].

To develop in Android (which is Java + XML), Android Studio is the main IDE to develop Android applications, it is also developed by Google to succeed the Eclipse ADT Plugin that was the previous officially supported method to develop android applications. Android Studio features a handful of new capabilities such as the Gradle-based build system that improves dramatically the ease of importing APIs and other libraries to merge into the application being developed. It also features Android Virtual Device (AVD) that allows to test apps in different sorts of virtual smartphone models in case a physical device cannot be used. Still the best method would indeed to use a physical device, it provides the most realistic scenario for application testing [34].

1.5.7.2 iOS

iOS is a proprietary operating system developed by Apple Inc. and is also Unix based like Android and is exclusively targeted for Apple products. It is the second most popular mobile OS in the market at this point and the latest version is the iOS 10, major versions are released annually [35].

The development of applications for iOS is done with the Xcode IDE from Apple itself and is the official platform to develop for Apple products. It has the downside that it is only available for Macintosh computers (that feature Apple's Mac OS), so it is necessary to have an Apple computer to be able to develop applications to these platforms [36]. Within the Xcode programming, there are two possible choices for the programming languages: Objective-C and Swift. Objective-C was the main language to develop applications for Apple products but recently Apple created their own programming language Swift which became the primary language for development. Support for Objective-C was not dropped, it is possible to program

in that language but it is recommended to start using Swift as it has a lot of improvements towards Objective-C [37][38].

1.5.7.3 Windows Phone

Windows Phone is developed by Microsoft and is a mobile operating system, the latest being Windows 10 Mobile. It is also closed source like iOS but is not locked for Microsoft products only. This OS is primarily aimed to the consumer market to try and compete with Android and iOS [39].

The development of applications for Windows Mobile is done via the Visual Studio IDE from Microsoft and uses a mixture of programming languages. For the User Interface, XAML is used, C# or Visual Basic for the code and optionally C++ to take full advantage of the phone's graphics hardware.

To make a general comparison of the three considered mobile operating systems, a table was constructed:

	Android	iOS	Windows Phone
Company	Google	Apple, Inc	Microsoft
Market Share (2015Q2) [40]	82.8%	13.9%	2.6%
License	Free and open source	Proprietary	Proprietary
OS family	Linux	Darwin	Windows Phone
Package Manager	APK	iTunes	XAP, APPX
SDK platform(s)	Linux, Mac OS X, Windows	Mac OS X, macOS	Windows
Cost to develop	Free	Free with Xcode7	Free[41]

Table 4 - Mobile OS comparison

With all considered, the ideal mobile OS is Android. It presents a lot of advantages to its competitors: it is free and open source, it has the largest market share with a huge margin, it can be developed in all the three major Operating Systems and has zero cost to develop.

1.5.8 Wireless Communication

For the wireless communications, there are not many technologies to choose from, but they present big differences from each other and the important factor will be the distances and high bandwidth capabilities.

1.5.8.1 Wi-Fi

Wi-Fi is a wireless technology that allows for computers/devices to connect to a Wireless Local Area Network (WLAN) using mostly 2.4Ghz or 5Ghz radio bands. The IEEE (Institute of Electrical and Electronics Engineers) LAN/MAN Standards Committee has standardized the technology as 802.11 and is responsible for its maintenance. Since its creation in 1997, the protocol has been having constant improvements that are categorized in letters [42], the main ones being 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac and a lot more. Today's mostly used standards are 802.11n and ac with still older ones like 802.11g still widely used. Making a comparison of these two, we would have:

	IEEE 802.11n	IEEE 802.11ac
Approx. Range outdoor (m)	250	No clear value (Beamforming technology)
Band (GHz)	2.4/5	5
Bandwidth (MHz)	20/40	20/40/80/160
Data rate (Mbps)	Up to 450	Around 1331
MIMO Streams	4	8

Table 5 - Wi-Fi popular standards comparison

The actual speed of the transfers is hard to measure as it depends of the environment, interferences and router settings. In the case of the 802.11ac standard it works differently, the Beamforming technology works by detecting where the target device is in space and intensifies the signal in its direction, which works much better than the traditional Omni-directional signal propagation of the 802.11n [43][44].

1.5.8.2 Cellular Network

Cellular network is a form of communication using cell towers that are distributed in different areas, known as base stations. The most used standards (technologies) for cellular communication are: UMTS, HSPA, LTE and LTE-A (in order of release). UMTS can provide

speeds up to 384Kbps, HSPA can provide up to 42Mbps of Downlink and 28Mbps of Uplink (HSPA+), LTE can provide up to 300Mbps of Downlink and 75Mbps of Uplink and LTE-Advanced can go up to 3Gbps of Downlink and 1.5Gbps of Uplink, the latency is highest in UMTS and lowest in LTE-A. The best scenario would be to have always service coverage from HSPA or even LTE or LTE-A which offer the highest speeds and the lowest latencies [45][46][47][48], but since bandwidth is not much of a problem for the type of data being dealt with (of what any of these standards can provide), the only concern will be the latency. As long as at least a good UMTS coverage is provided the system should theoretically work fine.

	WCDMA (UMTS)	HSPA HSDPA / HSUPA	HSPA+	LTE
Max downlink speed bps	384 k	14 M	42 M	300 M
Max uplink speed bps	128 k	5.7 M	28 M	75 M
Latency, time approx	150 ms	100 ms	50ms (max)	~10 ms
3GPP releases	Rel 99/4	Rel 5 / 6	Rel 7	Rel 8
Approx years of initial roll out	2003 / 4	2005 / 6 HSDPA 2007 / 8 HSUPA	2008 / 9	2009 / 10
Access methodology	WCDMA	WCDMA	WCDMA	OFDMA / SC-FDMA
Bandwidth	5 MHz	5 MHz	5 MHz	1.4 ~20MHz

Figure 21 - Cellular standards comparison

In Portugal, ANACOM conducted a study from September 2014 to December 2014, an evaluation of the cellular performances throughout the country (UMTS, HSPA and LTE combined). The worst scenarios tested were throughout the highways of the country and the mean latency values the tests showed, considering all of the ISPs, was 42ms, which is still a great value to work with. Also, globally, the coverage values for LTE were all above 88% and above 95% for the rest of the older standards [49].

Considering both technologies, the right approach for communication is by using cellular networks of 3G/4G connections. The deciding factor is mainly distance as with by going with a cellular approach, the distance becomes irrelevant as long as the area has good service coverage. If Wi-Fi was used, a router would have to be nearby to supply internet connection to the system and the limited range of the Wi-Fi technology used (802.11n or 802.11ac) would limit the movability of the USV

1.6 Dissertation structure

The dissertation is composed by five sections:

- Section 1 (Introduction) – Where an overview of the system and early studies are described.
- Section 2 (Sensors and computational platform) – Where all the sensors and embedded computers are described and also the implementation of the hardware part.
- Section 3 (Software) – Where all the developed server sided scripts, embedded platform scripts, streaming and additional software is described and explained.
- Section 4 (Results and evaluation) – Where the tests results are represented.
- Section 5 (Conclusions and future work) – Where final thoughts of the development of the system are described and also possible improvements for future iterations.

2

SENSORS AND COMPUTATIONAL PLATFORM

2.1 Hardware Overview

The architecture of the system includes sensor measurement channels connected to the Raspberry Pi, that allows signal acquisition, signal processing using Python scripts implemented on the level of computation platform expressed by Raspberry.

The system architecture is represented in figure 22:

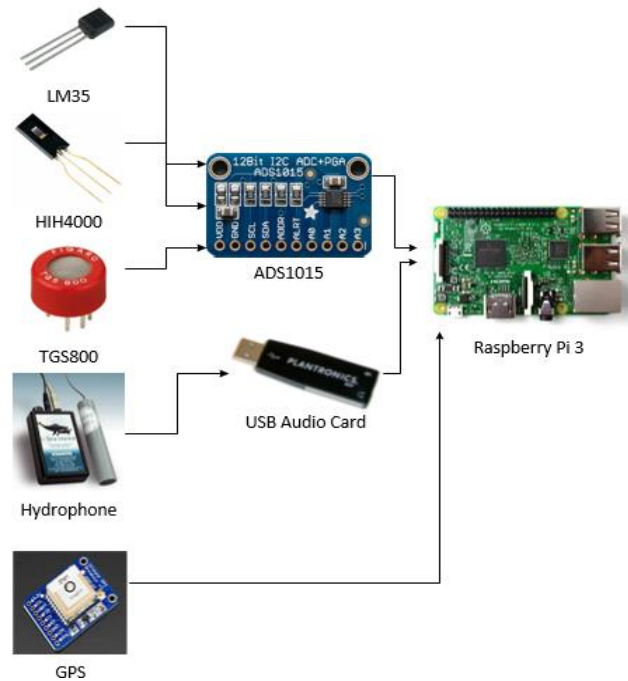


Figure 22 - Hardware system architecture overview

The schematic of the implemented architecture is presented in Figure 21:

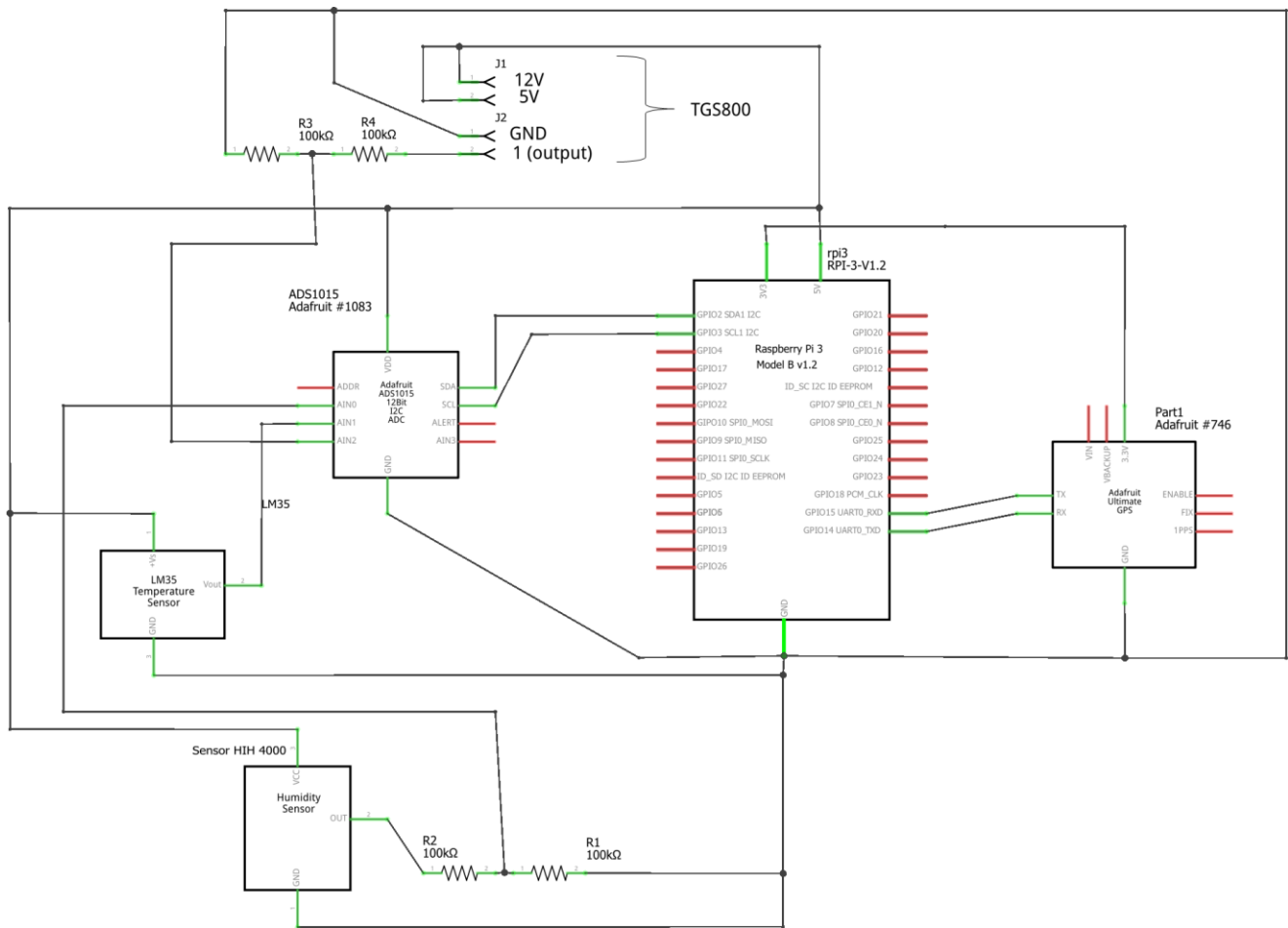


Figure 23 - Circuit schematics

The analog sensors are connected to the Raspberry Pi via an ADC board based on ADS1015. The hydrophone channel requires special analog to digital conversion and interfacing capabilities thus an USB Audio Card was employed and connected to a Raspberry USB port. The GPS is connected to the Raspberry via the UART TX (transmit) and RX (receive) ports in the GPIO.

2.2 Adafruit ADS1015 Analog to Digital Converter

The ADC will play as a middleman for the data acquisition from the sensors and inputting them to the Raspberry. All of the analog sensors are connected to the analog ports (A0 – A4) of the ADC. It communicates with the Raspberry using an I2C interface and with the support of the Adafruit library it is possible to work with the device. Since this ADC is 12bit, it outputs values between -2048 to 2047 (of which 2047 means it is at its peak of 4.096V with the PGA configured to 1), the PGA is a Programmable Gain Amplifier, which amplifies weak signals for better readings, it goes from 1 to 16 [30]. To proceed with digital conditioning, it is necessary to obtain the values in volts, and to do so a formula was applied:

$$\mathbf{Volts} = \frac{4.096 \cdot A_{0-4}}{2047} \quad (1)$$

Where 4.096 is the maximum voltage for PGA = 1, 2047 is the maximum value the ADC outputs at 4.096V, and A0-4 are the analog input ports that will receive the data from the sensors.

2.3 Analog Sensors

2.3.1 HIH 4000

The humidity sensor has got three wires for output and power (Vin, output and ground). It is powered by the Raspberry with 5V and the output is connected to an analog port in the ADC module.

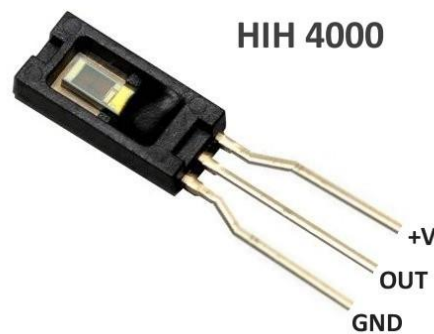


Figure 24 - HIH 4000 connections

The implemented relation to extract the relative humidity (%) values out of the voltage is:

$$\text{RH}(\%) = \frac{\frac{V_{\text{out}}}{V_{\text{supply}}} - 0.16}{0.0062} \quad (2)$$

2.3.2 LM35

The temperature sensor has a similar setup as the humidity sensor, it also presents with three connections: Vs, GND and Vout. It is also powered with 5V from the Raspberry and the output is connected directly to the ADC module without additional conditioning required.

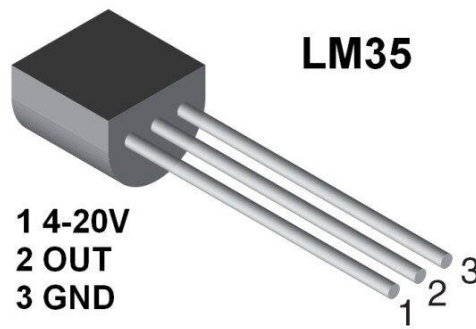


Figure 25 - LM35 connections

2.4 Ultimate GPS Breakout v3

This GPS model is made by Adafruit and it requires a 3.3V power input, which the Raspberry can provide but it can also be powered with 5V as it has a dropout regulator. It features an external antenna for better signal coverage, a -165dBm high sensitivity receiver and can tack up to 22 satellites on 66 channels and only draws 20mA of current. It also has an LED to show the GPS status on satellite connections, it blinks every second while it is searching for satellites and blinks once every 15 seconds when it has abled to make a connection.

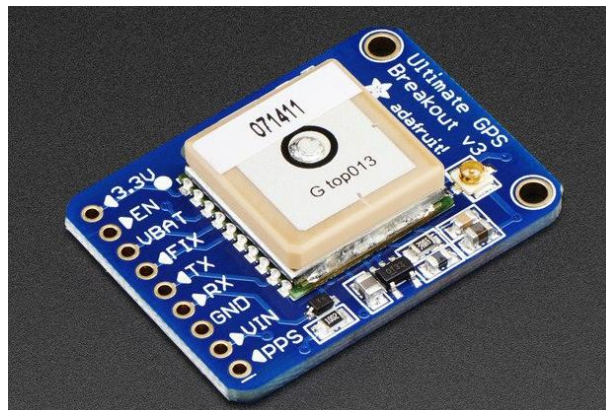


Figure 26 - Adafruit's Ultimate GPS Breakout v3

To connect this GPS model to the Raspberry Pi, the UART interface is used. The RX and TX connections of the GPS are cross-connected with the ones from the Raspberry Pi, then we connect the 3.3V pin to the 3.3V from the Raspberry and also the GND.

2.5 USB Audio Card & Sea-Phone Hydrophone

The audio card, since it is USB and possesses a chip that requires no additional drivers in Linux, it is already configured to work, it only requires some tweaks in the software side such as gain adjustments and audio input enabling.

The Hydrophone requires a AAA battery of 9V to power the device, other than that is just connecting the main microphone to the capture device and then connecting to the USB audio card in the input port with a 3.5mm jack cable.



Figure 27 - Hydrophone connected to the USB audio card

2.6 Raspberry Pi

The Raspberry Pi is the core of the system; all the sensors are connected to it and it handles all the analog processing. It is powered by Raspbian, the official Linux distribution for the Raspberry, it offers official support for specific Raspberry functionalities and not disregarding the advantages of having a Linux kernel. It receives data from all the analog sensors, the GPS and the hydrophone and saves the data in a local database and replicates to a remote server and also streams the audio received by the hydrophone. It will host all the python scripting and terminal commands needed to keep uploading/streaming data to the remote server. The Raspberry will only have the signaling script running at runtime, from there everything can be controlled by the android application.

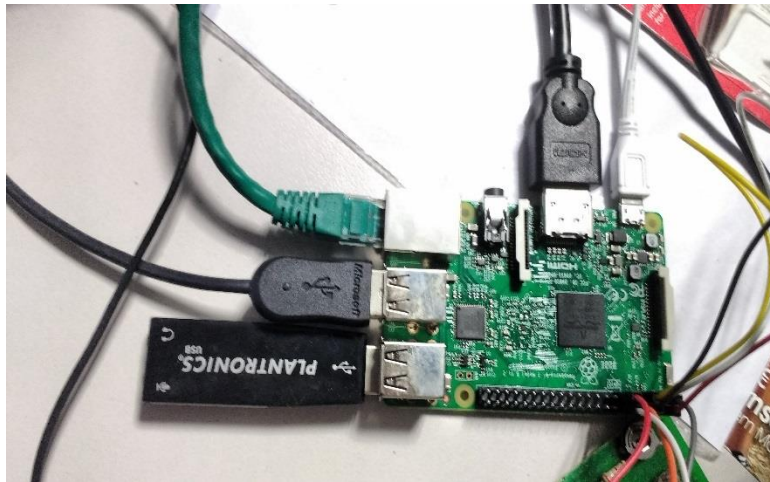


Figure 28 - Raspberry Pi 3 with the components connected

2.7 Conditioning circuits

For the gas and humidity sensors it was verified that, due to the high voltage output values, the ADC module could not handle such values and it saturated rather quickly, as the value could go above 5V. To solve this problem, it was applied a voltage divider in order to attenuate the values the sensor outputted.

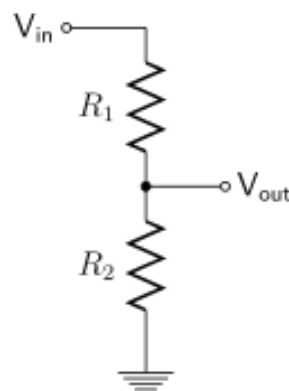


Figure 29 - Voltage Divider

The figure 10 shows how the voltage divider was implemented. V_{in} is the voltage value outputted from the HIH 4000 and the gas sensor, R_1 and R_2 in this particular case are the same which means cutting the voltage value in half, so if the the sensor outputs 1.2V, the voltage divider will attenuate the value to just 0.6V. It then outputs the new attenuated value at V_{out} where it is connected to an analog port of the ADC. With this solution it is possible to avoid saturation and the cut value is compensated back at a software level.

2.8 Network communication

The communication is done via 3G/4G from the Raspberry pi to the cloud server and vice versa to provide access to the data, the idea was to use a mobile router that stays in the USV with the Raspberry Pi to provide cellular internet access.

2.8.1 ZTE MF910 LTE Mobile Router

This mobile router will provide internet connection to the Raspberry Pi via Wi-Fi. This router receives connection through cellular networks, from UMTS to LTE, depending on what is available in the area of deployment and provides the internet connection via Wi-Fi to the connected interfaces. This router has a built-in battery but can also be connected to an external source for extra power.



Figure 30 - ZTE MF910 LTE Mobile Router

3 SOFTWARE

3.1 Software Overview

This chapter presents the system software describing every backend and frontend functionality within the Raspberry Pi, databases, remote server and the mobile application.

The software system architecture is represented in the following figure:

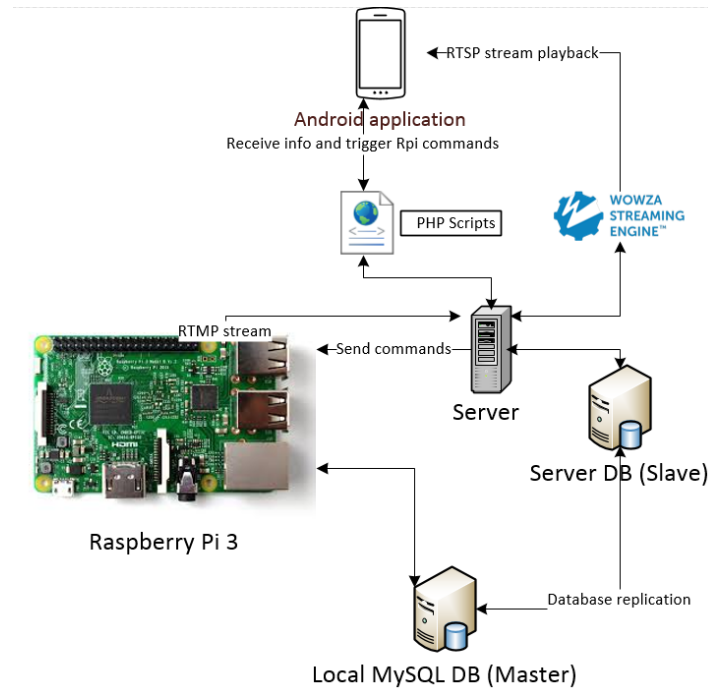


Figure 31 - Software system architecture

As it is presented in the figure 31, on the Raspberry level, it has a local database implemented that is fetching the data from the sensors, the values and the respective timestamps. The local database is then configured to be replicated by the remote server's own database, of which acts like a slave and copies the data from the master 1:1, every change done in the master is replicated by the slave. For the server side a set of PHP scripts were developed, the scrips being accessed by the mobile application that retrieve the sensors' data. As for the streaming, the Raspberry will stream the audio in RTMP to the server, and then the Android device will receive the streaming as RTSP.

3.2 Embedded Platform Software

The Raspberry Pi has a few scripts running and software to allow the system to work flawlessly. The used software platforms were Python, MySQL, FFmpeg and DDclient.

3.2.1 Python

Python is a widely used high-level programming language for general purpose, this language was selected due to having the most support on the Raspberry platform, including external devices, making it the right choice for this thesis. This programming platform is also easy to learn and emphasizes on code readability which makes it visually appealing and it can do a lot with few lines of code.

3.2.2 MySQL

MySQL is an open source software to manage relational databases. It is widely used in web applications and has a pretty solid replication capability which is very important for this project. MySQL is also going to be used in the server as the data is replicated.

3.2.3 FFmpeg

FFmpeg is a multimedia framework that is incredibly complete, it allows to decode, encode, transcode, mux, demux, stream, filter and play most of the codecs out there, community and corporation made. It is divided in 4 different tools: ffmpeg which converts multimedia files, ffmpeg which is a multimedia streaming server, ffplay which is a media player and ffprobe, a multimedia stream analyzer. For the thesis only the ffmpeg was used, in order to handle the audio streaming.

3.2.4 DDclient

DDclient is a software developed with Perl used to update DNS entries for accounts on DDNS (Dynamic Domain Name System) services. The usage of a DDNS is necessary to overcome the problem of the dynamic IP address by the ISP of the 3G/LTE connection. By registering a domain name and associating it to the DDclient, the software will keep updating the domain name with the latest active dynamic IP given by the ISP. With this solution, there is no longer need to worry about IP changes as the connection is made using a DNS address instead.

3.2.5 Python Scripts

Two python scripts were developed. One of them is designed to handle the data acquisition thus allowing the Raspberry to obtain the data from the sensors. The Adafruit's ADC library was used in this case. The sensor data is uploaded to the local MySQL database installed in the Raspberry pi SD card. The script starts by executing a thread for each sensor with a user inputted reading interval, it then keeps reading the data with the necessary conditioning from the ADC module every X seconds and uploading to the database at the same time. The script ends with

a SIGINT signal sent from the keyboard combination CTRL+C or with a killall/kill command which terminates all the threads and exits.

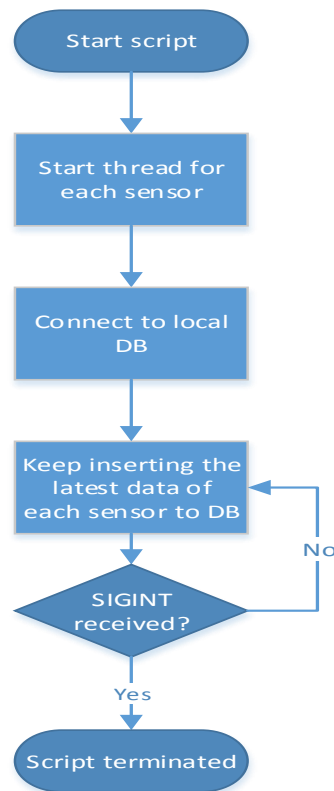


Figure 32 - Data acquisition script flow chart

The second script is designed to handle the signalling received from the server “signals” database that is triggered by the mobile phone. The script is running a thread that is scanning the database every second for changes in the values, the database is updated through a PHP script that the android application calls and triggers the database change. Whenever there are changes triggered by the android application, the script will proceed to execute the requested actions. There are five possible actions that can be executed:

1. Start or stop the data acquisition script
2. Start or stop the stream
3. Shutdown the Raspberry

This way the application has a more dynamic way of controlling the whole system without the constant need to log into the Raspberry, everything can now be made at a click of a button.

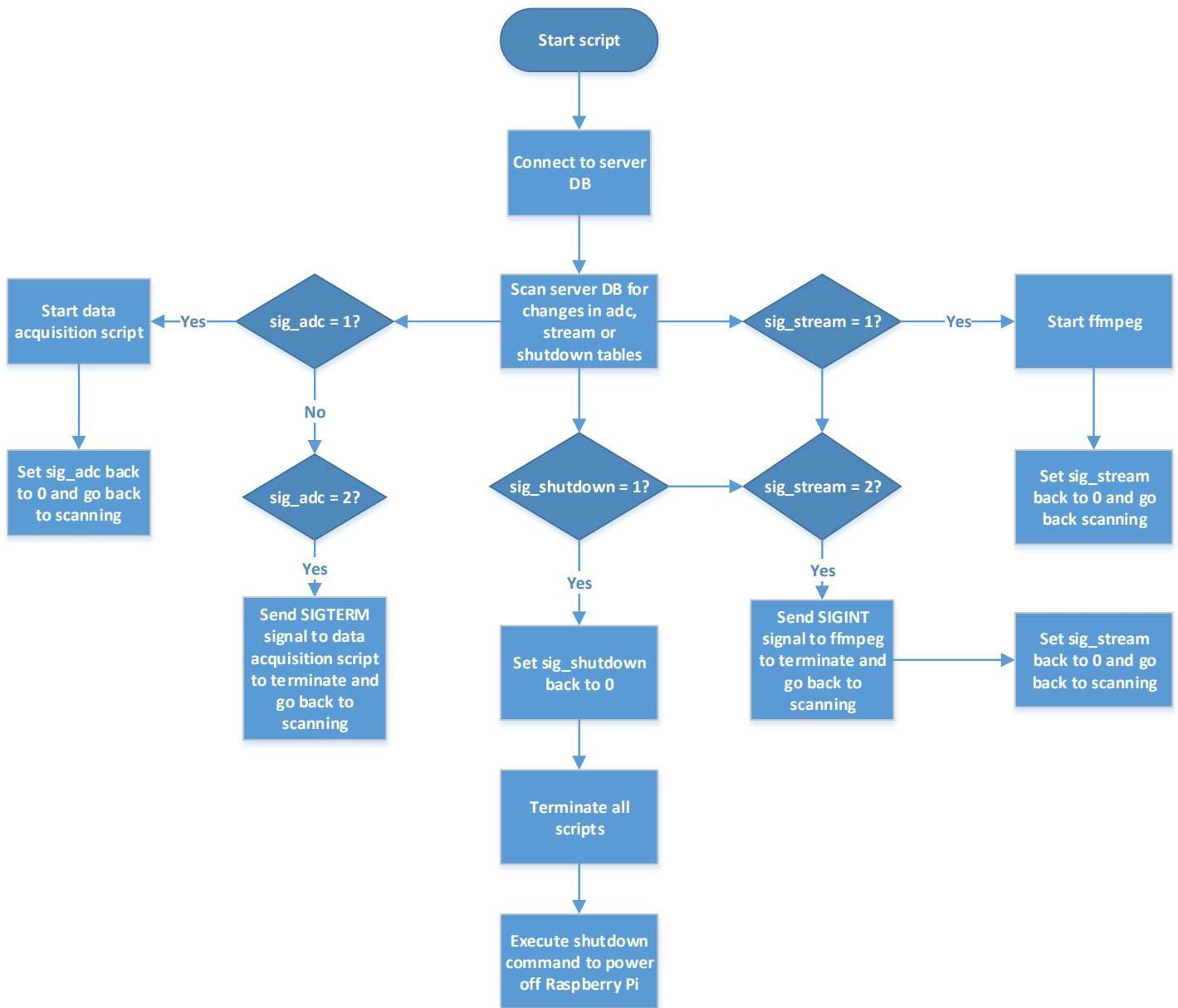


Figure 33 - Signals script flow chart

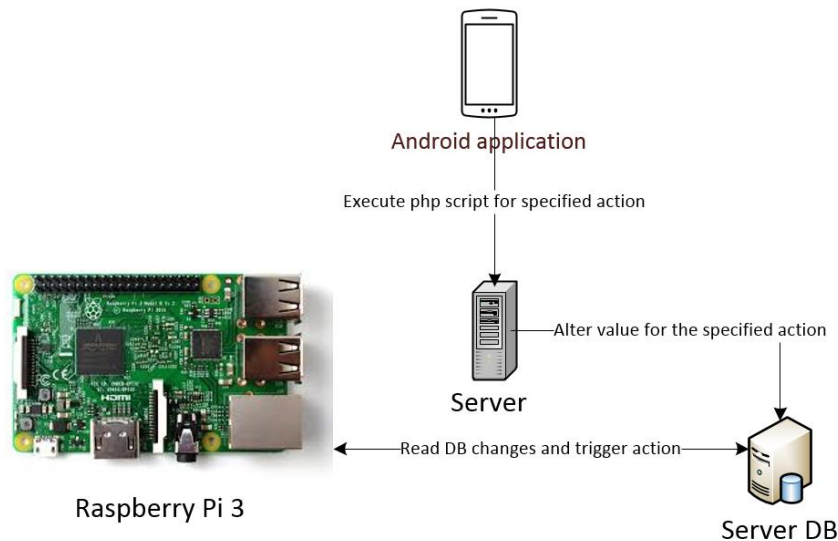


Figure 34 - Raspberry signalling scheme

3.2.6 Python Scripts Code Highlights

For the data acquisition script, a function was made to convert the ADC values into volts and another function to apply the correct formula based on which sensor is connected in which port (A0-A4):

```
# function to convert the ADC values into volts
def adc_volts (analog_port):
    result = float((4.096*(adc.read_adc(analog_port, gain=GAIN)))/2047.0)
    return result

# function to apply the formula correct formula to each sensor
def get_sensor_data (sensor_name):
    if 'hum' in sensor_name:
        result = ((5/(adc_volts(1)*2))-0.16)/0.062
        return result
    elif 'temp' in sensor_name:
        result = adc_volts(0)*100
        return result
    elif 'gas' in sensor_name:
        result = adc_volts(2)*2
        return result
    else:
        print "Invalid sensor name"
```

Figure 35 - Data acquisition python functions

For the data insertion, a thread for each sensor is constructed with different time intervals of data insertion:

```
class workThread (threading.Thread):  
  
    def __init__(self, ID, sensor, delay):  
        threading.Thread.__init__(self)  
        self.ID = ID  
        self.sensor = sensor  
        self.delay = delay  
        self._stop = threading.Event()  
  
    def stop(self):  
        self._stop.set()  
  
    def stopped(self):  
        self._stop.isSet()  
  
    def run(self):  
        db = MySQLdb.connect("localhost", "████", "████", "drone")  
        db.autocommit(False)  
        cursor = db.cursor()  
  
        while not exitflag:  
            if 'hum' in self.sensor:  
                try:  
                    cursor.execute("insert into Humvalue (value) values (%s)", (get_sensor_data(self.sensor)))  
                    print "HUMVALUE INSERTED"  
                    db.commit()  
                except:  
                    db.rollback()  
                    time.sleep(self.delay)  
  
            elif 'temp' in self.sensor:
```

Figure 36 - Code snippet of the sensors' thread

For the signals' script that triggers linux signals based on the information detected on the database triggered by the mobile application, a threading principle was also used to keep reading for database changes every second:

```
while not exitflag:
    if 'adc' in self.action:
        try:
            cursor.execute("SELECT * FROM sig_adc")
            db.commit()
        except:
            db.rollback()
        results = cursor.fetchall()

        for row in results:
            if row[0] == 1:
                status = os.system("/home/pi/python/testdb.py &")
                print "result adc: ", status
                try:
                    cursor.execute("UPDATE sig_adc SET sig_state = 0")
                    db.commit()
                except:
                    db.rollback()
            elif row[0] == 2:
                status = os.system("sudo killall -s SIGTERM testdb.py")
                print "result sig 2 adc: ", status
                try:
                    cursor.execute("UPDATE sig_adc SET sig_state = 0")
                    db.commit()
                except:
                    db.rollback()
            time.sleep(self.delay)

    elif 'stream' in self.action:
        try:
            cursor.execute("SELECT * FROM sig_stream")
```

Figure 37 - Code snippet of the signals' script

The script in figure 37, keeps accessing the database every second and sees if the values saved are 1 or 2. If the value is 1 it starts the data acquisition script and if the value is two, sends a SIGTERM to terminate the script. Whenever these actions are done, the value is always reverted back to 0.

3.2.7 MySQL Database

The MySQL database has a straightforward design, the first database has a table for each sensor and their fields are: id, value and timestamp. This allows the storing of every sensor reading and the respective time of when it was measured.

The second database is called signals and has the tables sig_adc, sig_stream and sig_shutdown. Each of these tables have a field called sig_state which can have the values 0, 1 or 2, apart from the sig_shutdown table which only goes until 1. The default value is 0, which is the state of not doing anything. For the remaining values, the logic is the following for each of the tables:

- sig_adc: if the value is set to 1, it means the mobile application wishes to start the data acquisition script, and if it is set to 2 it wishes to stop the script instead (if it is already running).
- sig_stream: if the value is set to 1, the Raspberry will start the ffmpeg streaming command and the streaming process will start. If it is set to 2, a SIGINT signal is sent to the ffmpeg process in order for it to terminate safely, which ends the stream.
- sig_shutdown: if the value is set to 1, Raspberry receives a shutdown order.

The values are reverted back to 0 whenever an action is requested and triggered. The overview of all the databases look like this:

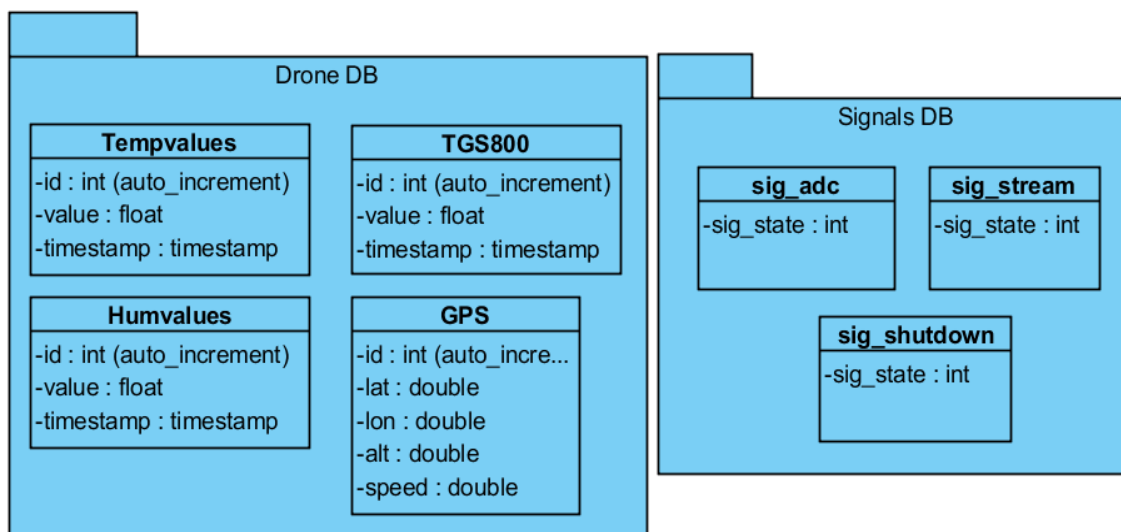


Figure 38 - Databases' structure

3.2.8 GPS Software

In order for the GPS model from Adafruit to work, the Raspberry Pi must have a program that handles the messages from the GPS and translate them into human readable, and that software is GPSd.

GPSd is a service daemon that monitors GPS's receivers that are connected to a computer through USB, serial, etc. and sends the data to the TCP port 2947 of the host computer. This way it is possible to access the localhost and the TCP port 2947 and the GPS data is accessible in a readable way, but of course the data requested must be specified: coordinates, current speed, altitude, etc. To do that, Adafruit provides a library for python that can be used in the data acquisition script and listen to the localhost and specify the data do be extracted, i.e query the GPSd for GPS data.

With all these tools, it is not necessary to deal with the raw NMEA messages and all the errors the might surface, as there is already software that deals with that. A typical NMEA message looks like this:

```
$GPGSA,A,3,01,11,32,20,14,28,19,17,,,,,1.9,1.1,1.6*3E
$GPRMC,160301.000,A,5556.5410,N,00311.4375,W,0.63,331.80,211012,,,A*71
$GPGGA,160302.000,5556.5410,N,00311.4376,W,1.08,1.1,76.2,M,49.6,M,,0000*7E
$GPGSA,A,3,01,11,32,20,14,28,19,17,,,,,1.9,1.1,1.6*3E
$GPRMC,160302.000,A,5556.5410,N,00311.4376,W,0.00,331.80,211012,,,A*74
$GPGGA,160303.000,5556.5410,N,00311.4376,W,1.08,1.1,76.2,M,49.6,M,,0000*7F
$GPGSA,A,3,01,11,32,20,14,28,19,17,,,,,1.9,1.1,1.6*3E
$GPRMC,160303.000,A,5556.5410,N,00311.4376,W,0.00,331.80,211012,,,A*75
$GPGGA,160304.000,5556.5409,N,00311.4376,W,1.08,1.1,76.2,M,49.6,M,,0000*70
$GPGSA,A,3,01,11,32,20,14,28,19,17,,,,,1.9,1.1,1.6*3E
$GPGSV,3,1,12,01,83,262,34,11,72,142,35,32,66,182,35,20,38,219,30*7B
$GPGSV,3,2,12,14,32,058,28,28,30,274,28,19,25,153,34,17,23,309,23*79
$GPGSV,3,3,12,22,11,056,16,24,09,095,,27,08,003,,09,06,352,*78
$GPRMC,160304.000,A,5556.5409,N,00311.4376,W,0.00,331.80,211012,,,A*7A
$GPGGA,160305.000,5556.5410,N,00311.4376,W,1.08,1.1,76.2,M,49.6,M,,0000*79
```

Figure 39 - Example of an NMEA message

This is the raw data the most GPS devices emit that need to be parsed in order to present readable data. With the use of GPSd, the data is already parsed and looks like this:

Time:	2016-10-23T14:34:56.000Z	PRN:	Elev:	Azim:	SNR:	Used:
Latitude:	38.752814 N	23	73	118	44	Y
Longitude:	9.149126 W	17	54	259	29	Y
Altitude:	111.0 m	9	51	193	16	Y
Speed:	0.0 kph	3	50	049	49	Y
Heading:	233.9 deg (true)	19	48	288	25	Y
Climb:	-6.0 m/min	136	42	158	32	Y
Status:	3D FIX (97 secs)	22	34	064	45	Y
Longitude Err:	+/- 3 m	1	32	117	36	Y
Latitude Err:	+/- 3 m	6	28	310	12	Y
Altitude Err:	+/- 9 m	11	16	139	41	N
Course Err:	n/a	31	07	041	26	N
Speed Err:	+/- 24 kph					
Time offset:	-510841.697					
Grid Square:	IM58ks					

Figure 40 - cgps output (GPSd)

With the help of python, it is possible to access this data directly for the data acquisition script.

3.2.9 Audio Stream

After connecting the hydrophone and the USB Audio card to the Raspberry it is necessary to activate the microphone and tweak the volume using the ALSA software. ALSA allows to configure the multimedia hardware connected to the Raspberry, in this case we want to configure the USB Audio card, which is mostly just activate the audio input functionality and control the volume.

To be able to stream the audio from the hydrophone, the ffmpeg command will receive the data from the USB audio card through ALSA and encapsulate the audio in an RTMP stream that is sent to the server. As the ffmpeg is in fact a powerful command and allows for a lot of variety of different approaches to construct a good command for the needs of the user, it was necessary to test different approaches and protocols in order to obtain a satisfying stream. The output of the stream is represented in the following picture:

```
ffmpeg version N-80205-g43a4276 Copyright (c) 2000-2016 the FFmpeg developers
built with gcc 4.9.2 (Raspbian 4.9.2-10)
configuration: --arch=armel --target-os=linux --enable-gpl --enable-nonfree
libavutil      55. 24.100 / 55. 24.100
libavcodec     57. 44.101 / 57. 44.101
libavformat    57. 37.101 / 57. 37.101
libavdevice    57.  0.101 / 57.  0.101
libavfilter     6. 46.100 /  6. 46.100
libswscale      4.  1.100 /  4.  1.100
libswresample  2.  0.101 /  2.  0.101
libpostproc   54.  0.100 / 54.  0.100
Guessed Channel Layout for Input Stream #0.0 : stereo
Input #0, alsa, from 'plughw:1':
  Duration: N/A, start: 1476716787.617039, bitrate: N/A
  Stream #0:0: Audio: pcm_s16le, 8000 Hz, 2 channels, s16, 256 kb/s
Output #0, flv, to 'rtmp://52.222.22.222/live/myStream':
  Metadata:
    encoder      : Lavf57.37.101
  Stream #0:0: Audio: aac (LC) ([10][0][0][0] / 0x000A), 8000 Hz, stereo, fltp, 96 kb/s
  Metadata:
    encoder      : Lavc57.44.101 aac
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_s16le (native) -> aac (native))
Press [q] to stop, [?] for help
```

Figure 41 - FFmpeg output

3.2.9.1 Fast Fourier Transform (FFT)

It is important to understand how strong or weak are the sound emitting sources and to know that, it is necessary to have a spectrum of frequencies to analyse. These frequencies will also help in the identification of the audio source and to obtain the frequencies, an implementation of the FFT algorithm was included. The FFT algorithm will compute the Discrete Fourier Transform (DFT) which will convert the signal, that is in the time domain, into a representation in the frequency domain.

In order to feed the algorithm with data, it first needs some understanding. The audio WAV file comes in a raw byte format, and the first step is to calculate the number of FFT points, also known as the bin size. To do this, the following formula is used:

$$FFTSize = \frac{f_s}{f_{\Delta}} = f_s \cdot T_{\Delta} \quad (3)$$

Where f_s is the sampling frequency (configured in ffmpeg, 8kHz used) and f_Δ is the frequency resolution, but since the length of the audio file is used, it needs to be converted in time, becoming T_Δ .

The FFT algorithm only works with a bin size value that is a power of two so, after applying the formula, the actual result will be an equivalent power of two value of the bin size result in order to feed FFT with data. The FFT algorithm receives an array of complex values (after converting the bytes into complex values) and outputs the FFT data also in complex format. To have the right values, it is necessary to compute the magnitude of the output complex array to obtain the amplitude values in a double format. It is now possible to plot the FFT data using the output double array as the Y axis (amplitude values) and X as the frequencies in Hz.

```
Complex[] y;  
Complex[] complexSignal = new Complex[fftPoints];  
double[] absSignal = new double[fftPoints/2];  
//Convert byte array into a complex array  
for(int i = 0; i < fftPoints; i++){  
    if ((2*i+1) < signal.length) {  
        temp = (double) ((signal[2 * i] & 0xFF) | (signal[2 * i + 1] << 8)) / 32768.0F;  
        complexSignal[i] = new Complex(temp, 0.0);  
    } else {  
        complexSignal[i] = new Complex(0.0, 0.0);  
    }  
}  
//perform FFT  
y = FFT.fft(complexSignal);  
  
//compute magnitude of complex values  
for(int i = 0; i < (fftPoints/2); i++){  
    absSignal[i] = y[i].abs();  
}  
return absSignal;
```

Figure 42 - Code snippet of FFT (android)

In order to compute the frequencies, it is necessary to calculate the frequency resolution, which by manipulating the formula (3), it goes like this:

$$f_\Delta = \frac{f_s}{FFTSize} \quad (4)$$

The resulting value, multiplying with the iterator value of the length of the output double array from the FFT algorithm, presents the frequency spectrum of the audio file plotted in the X axis.

The result:

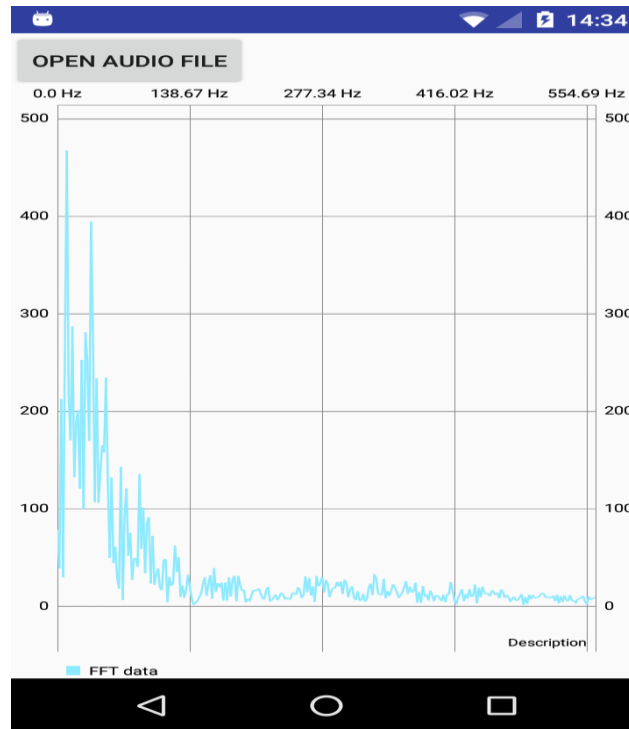


Figure 43 - FFT data output example

3.3 Server Side Software

The server is responsible for delivering all the data to the mobile application using PHP scripts, Wowza streaming engine and MySQL serving as a slave of the Raspberry.

3.3.1 PHP

PHP is a programming language mostly used for web development but is also suited for general use, it is going to be used as a form of communication between the android application and the server using the REST architecture.

3.3.2 Wowza

Wowza is streaming service that is installed in the server and allows to receive and distribute streams through publish and subscribe. It supports almost all of the streaming protocols available in the market and also a large number of video and audio codecs. It has the advantage of making the streaming process much simpler has it is very flexible of using different protocols when publishing and when subscribing a stream, allowing for an easy support for the android application.

3.3.3 PHP Scripts

The server houses a couple of PHP scripts to allow the mobile application to access the sensors' data. A total of 5 scripts were developed, three of them return all of the saved data from each sensor in a JSON format and they work like this:

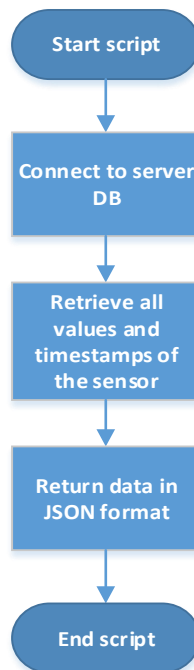


Figure 44 - flow chart for the full data returning scripts

The fourth script will return the latest recorded value for each sensor, depending on the “GET” variable that is passed by the mobile application. If a GET string “temp” is passed, the script returns the latest temperature value recorded, the same goes for the “hum” and “gas” strings for their respective sensors. The purpose of this script is to allow the mobile application to keep getting the latest value when it desires to show live readings within a certain interval.

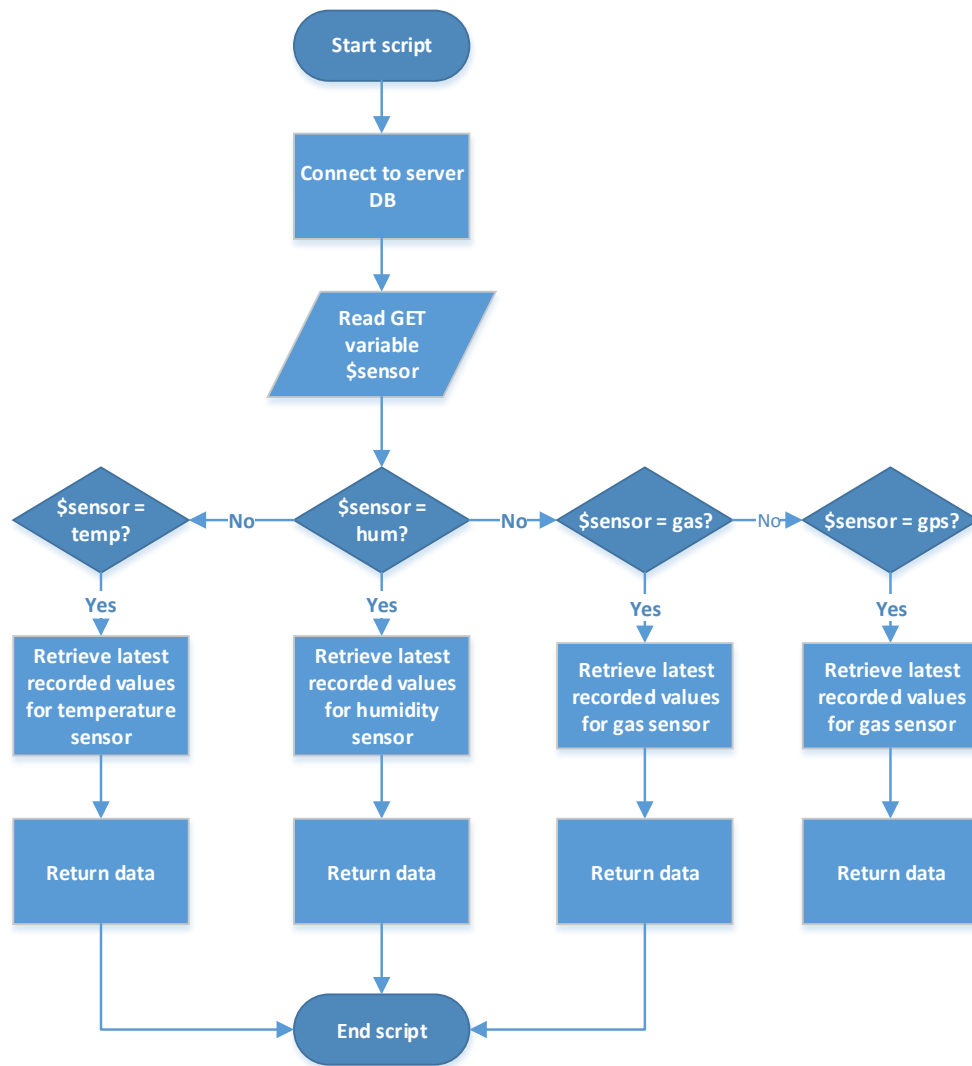


Figure 45 - Flow chart for the latest data returning script

The last script is responsible for sending the signals to the Raspberry to trigger actions by communicating with the Raspberry Pi through the cloud server's database to trigger signals.

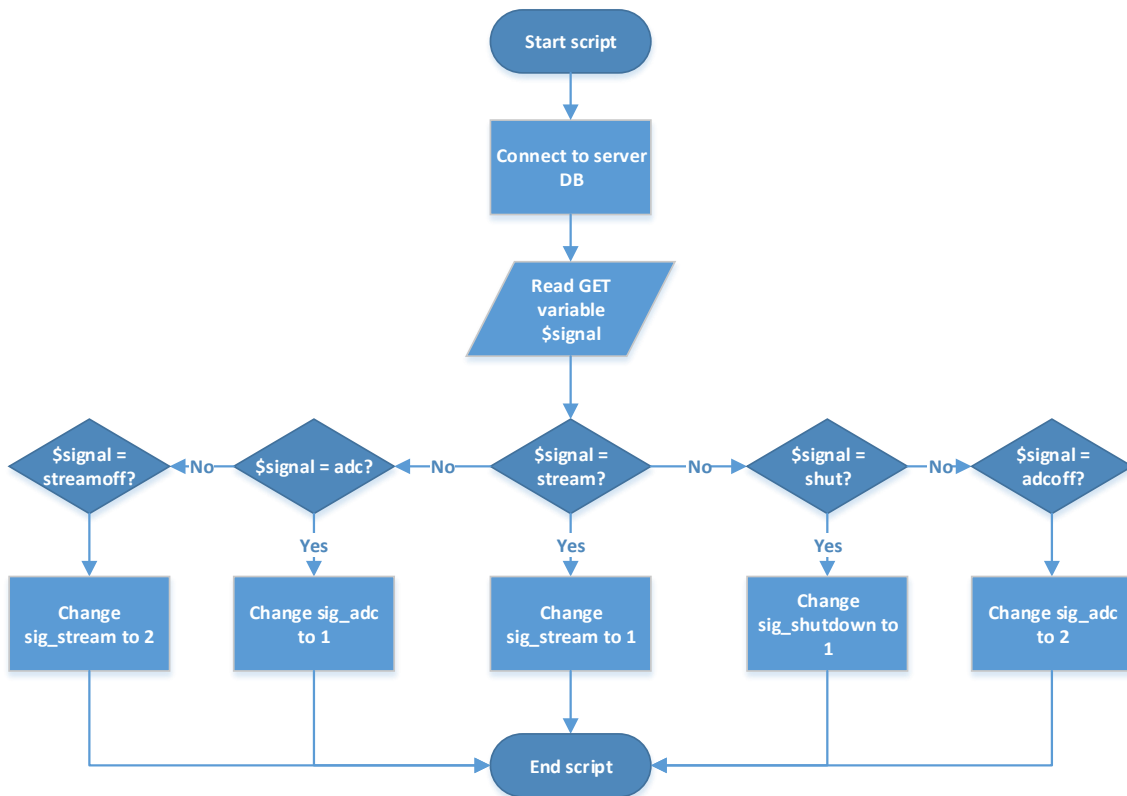


Figure 46 - Flow chart for the signal triggering script

3.3.4 PHP Scripts Code Highlights

For the signals' script, it's mainly just change the database value when the mobile application wants to trigger something on the Raspberry Pi:

```
<?php
$signal = $_GET['signal'];

ini_set('error_reporting', E_ALL);
ini_set('display_errors', 1);

$config = parse_ini_file('../info.ini');

$connection = mysqli_connect($config['serverlocation'], $config['user'], $config['password'], "signals");

if (!$connection){
    die ('Error: ' . mysqli_connect_error());
}

$signal = mysqli_real_escape_string($connection, $signal);

if ($signal === "adc"){

    $result = mysqli_query($connection, "UPDATE sig_adc SET sig_state = 1") or die ('Could not update entry');
    if ($result)
        echo "Successfully updated sig_adc";
}

elseif ($signal === "stream"){

    $result = mysqli_query($connection, "UPDATE sig_stream SET sig_state = 1") or die ('Could not update entry');
    if ($result)
        echo "Successfully updated sig_stream";
}
}
```

Figure 47 - PHP signals' script code snippet

The script will receive a GET value passed with the script, like this: “http://92.222.xx.xxx/drone/trigger_signal.php?signal=streamoff” and in this case will ask to stop the FFmpeg streaming program. The same for all other trigger actions: adc (to start data acquisition script), stream (to start FFmpeg), adcoff (to stop data acquisition script) and shut to shutdown the Raspberry Pi.

The script that returns the latest recorded value is implemented like this:

```
<?php
$sensor = $_GET['sensor'];

ini_set('error_reporting', E_ALL);
ini_set('display_errors', 1);

$config = parse_ini_file('../info.ini');

$connection = mysqli_connect($config['serverlocation'], $config['user'], $config['password'], $config['dbname']);

if (!$connection){
    die('Error: ' . mysqli_connect_error());
}

$sensor = mysqli_real_escape_string($connection, $sensor);

if ($sensor === "temp"){

    $result = mysqli_query($connection, "select value, timestamp from Tempvalue order by id desc limit 1") or die ();
    if (mysqli_num_rows($result) > 0){
        $row = mysqli_fetch_row($result);
        echo $row[0], " ", $row[1];
    }
}

elseif ($sensor === "hum"){

    $result = mysqli_query($connection, "select value, timestamp from Humvalue order by id desc limit 1") or die ();
    if (mysqli_num_rows($result) > 0){
```

Figure 48 - PHP latest value script code snippet

After connecting to the database just like all the PHP scripts have to do first, in this case, the script will get the latest recorded value in the database based on the sensor name passed in the GET variable triggered with the http link.

The last set of scripts are the ones that return all of the recorded data for each sensor:

```
<?php
$config = parse_ini_file('.././../info.ini');

$conn = mysqli_connect($config['serverlocation'], $config['user'], $config['password'], $config['dbname']);

if (!$conn){
    die('Error: ' . mysqli_connect_error());
}

$result = mysqli_query($conn, "select value, timestamp from Tempvalue") or die ('Could not select');

$rows = array();
while ($row = mysqli_fetch_row($result)){
    $rows[] = $row;
}

echo json_encode($rows);

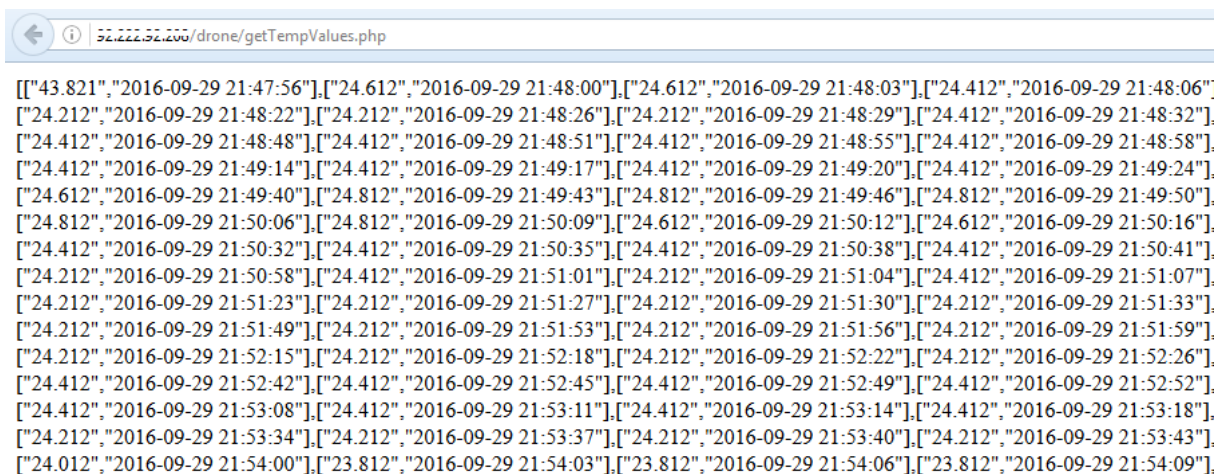
mysqli_close($conn);
?>
```

Figure 49 - PHP all data returning script code

These scripts are the simplest ones, which will just access the server's database and return all of the data for each sensor in a JSON format. There are three of these, one for each sensor.

3.3.5 JSON Format

The JSON format used in the data exchange is just arrays with the sensors' values and timestamps. The reason to use JSON is mainly because large data sets are manipulated and this format provides an easy parsing of the values, as JSON has libraries in practically all programming languages making it easy to implement. If regular strings had to be used with all the data, the substring parsing would take far more time to pull through. Example of a JSON output of the temperature values:



```
52.222.52.255/drone/getTempValues.php
[[["43.821","2016-09-29 21:47:56"],["24.612","2016-09-29 21:48:00"],["24.612","2016-09-29 21:48:03"],["24.412","2016-09-29 21:48:06"],
["24.212","2016-09-29 21:48:22"],["24.212","2016-09-29 21:48:26"],["24.212","2016-09-29 21:48:29"],["24.412","2016-09-29 21:48:32"],
["24.412","2016-09-29 21:48:48"],["24.412","2016-09-29 21:48:51"],["24.412","2016-09-29 21:48:55"],["24.412","2016-09-29 21:48:58"],
["24.412","2016-09-29 21:49:14"],["24.412","2016-09-29 21:49:17"],["24.412","2016-09-29 21:49:20"],["24.412","2016-09-29 21:49:24"],
["24.612","2016-09-29 21:49:40"],["24.812","2016-09-29 21:49:43"],["24.812","2016-09-29 21:49:46"],["24.812","2016-09-29 21:49:50"],
["24.812","2016-09-29 21:50:06"],["24.812","2016-09-29 21:50:09"],["24.612","2016-09-29 21:50:12"],["24.612","2016-09-29 21:50:16"],
["24.412","2016-09-29 21:50:32"],["24.412","2016-09-29 21:50:35"],["24.412","2016-09-29 21:50:38"],["24.412","2016-09-29 21:50:41"],
["24.212","2016-09-29 21:50:58"],["24.412","2016-09-29 21:51:01"],["24.212","2016-09-29 21:51:04"],["24.412","2016-09-29 21:51:07"],
["24.212","2016-09-29 21:51:23"],["24.212","2016-09-29 21:51:27"],["24.212","2016-09-29 21:51:30"],["24.212","2016-09-29 21:51:33"],
["24.212","2016-09-29 21:51:49"],["24.212","2016-09-29 21:51:53"],["24.212","2016-09-29 21:51:56"],["24.212","2016-09-29 21:51:59"],
["24.212","2016-09-29 21:52:15"],["24.212","2016-09-29 21:52:18"],["24.212","2016-09-29 21:52:22"],["24.212","2016-09-29 21:52:26"],
["24.412","2016-09-29 21:52:42"],["24.412","2016-09-29 21:52:45"],["24.412","2016-09-29 21:52:49"],["24.412","2016-09-29 21:52:52"],
["24.412","2016-09-29 21:53:08"],["24.412","2016-09-29 21:53:11"],["24.412","2016-09-29 21:53:14"],["24.412","2016-09-29 21:53:18"],
["24.212","2016-09-29 21:53:34"],["24.212","2016-09-29 21:53:37"],["24.212","2016-09-29 21:53:40"],["24.212","2016-09-29 21:53:43"],
["24.012","2016-09-29 21:54:00"],["23.812","2016-09-29 21:54:03"],["23.812","2016-09-29 21:54:06"],["23.812","2016-09-29 21:54:09"],
```

Figure 50 - JSON output example

3.3.6 Wowza Streaming Service

In order of the streaming to work, it is also necessary to configure the server to act as the distributor of the audio stream. It receives a stream publication from the FFmpeg software in

the Raspberry Pi and the mobile application will subscribe to that same stream that was published by the ffmpeg using RTMP. For all this to work, the Wowza Streaming Engine was installed in the server to facilitate the streaming process. Wowza generates a link where it is allowed to publish streams with almost any audio/video streaming protocol and at the same time allows for subscriptions that lets a client software to play the audio/video, in this case, the mobile application. And since Wowza is flexible on the protocol usage, it is possible to play the stream using RTSP which Android has official support.

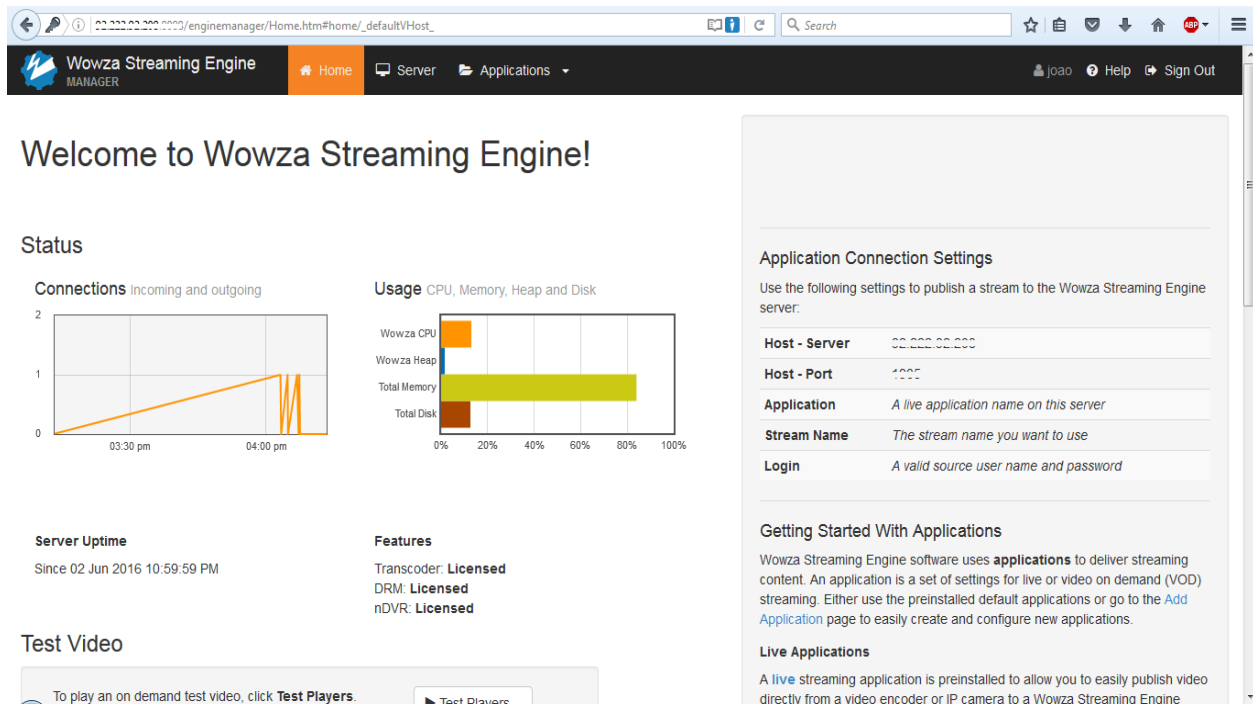


Figure 51 - Wowza Streaming Engine main screen

3.4 Android Application

The android application acts as a terminal of which a user can access all the data visually, the audio stream, audio data and also perform some actions on the Raspberry Pi.

3.4.1 Main screen

The activity that shows the data from the sensors, gps data and drone location on the map. The data is read live and in the case of the map, it tracks the movement of the drone by leaving a trail of markers of all its positions plus the current one. It keeps getting info from the PHP scripts and the RTSP stream from the cloud server.

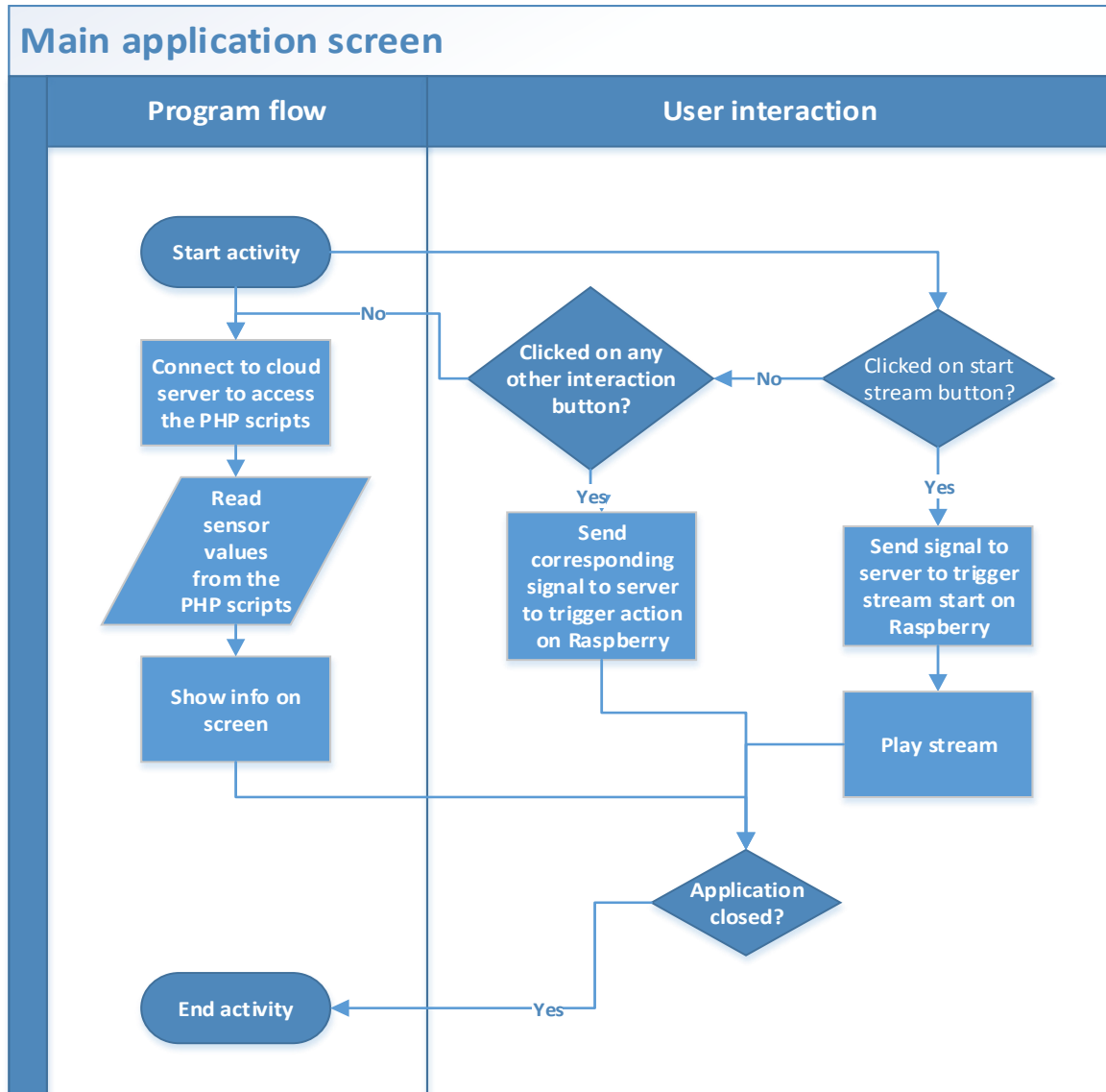


Figure 52 - Main Activity flowchart

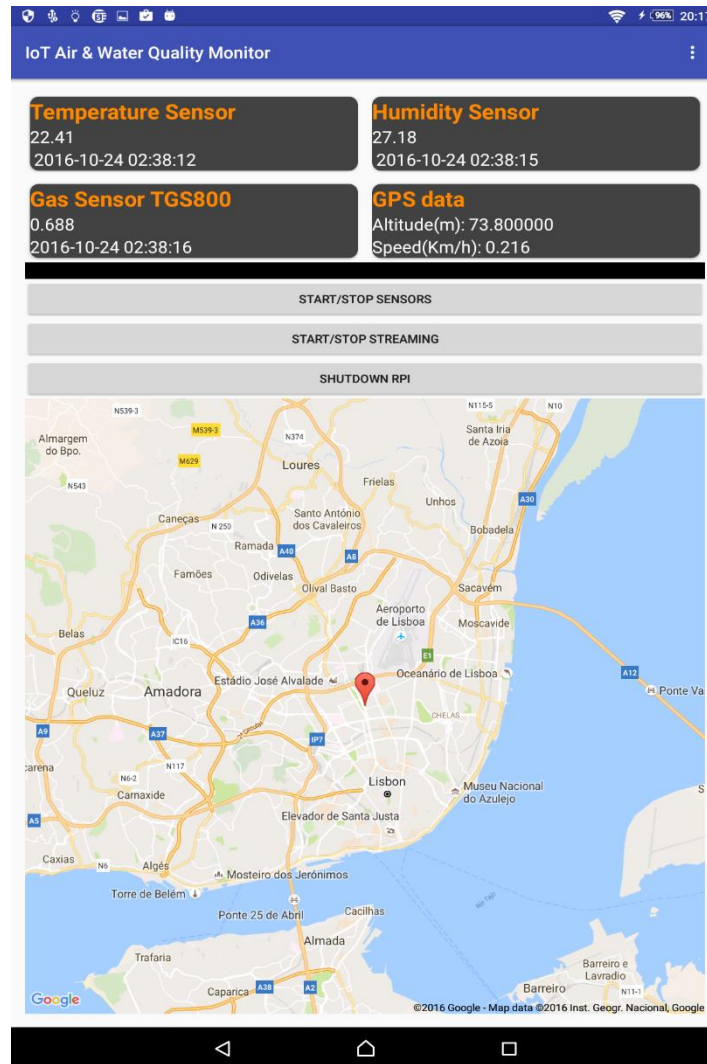


Figure 53 - Main Activity screen shot

3.4.2 Sensors' graphical data

This activity shows all the data from each sensor saved in the cloud's database by accessing the PHP files and parsing the output JSON data into graphs.

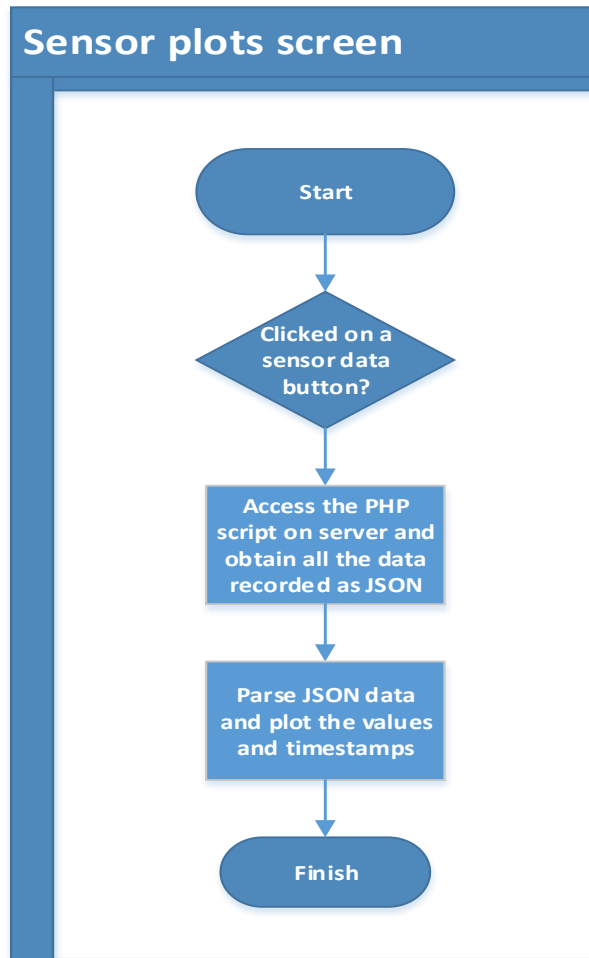


Figure 54 - Graphs activity flowchart

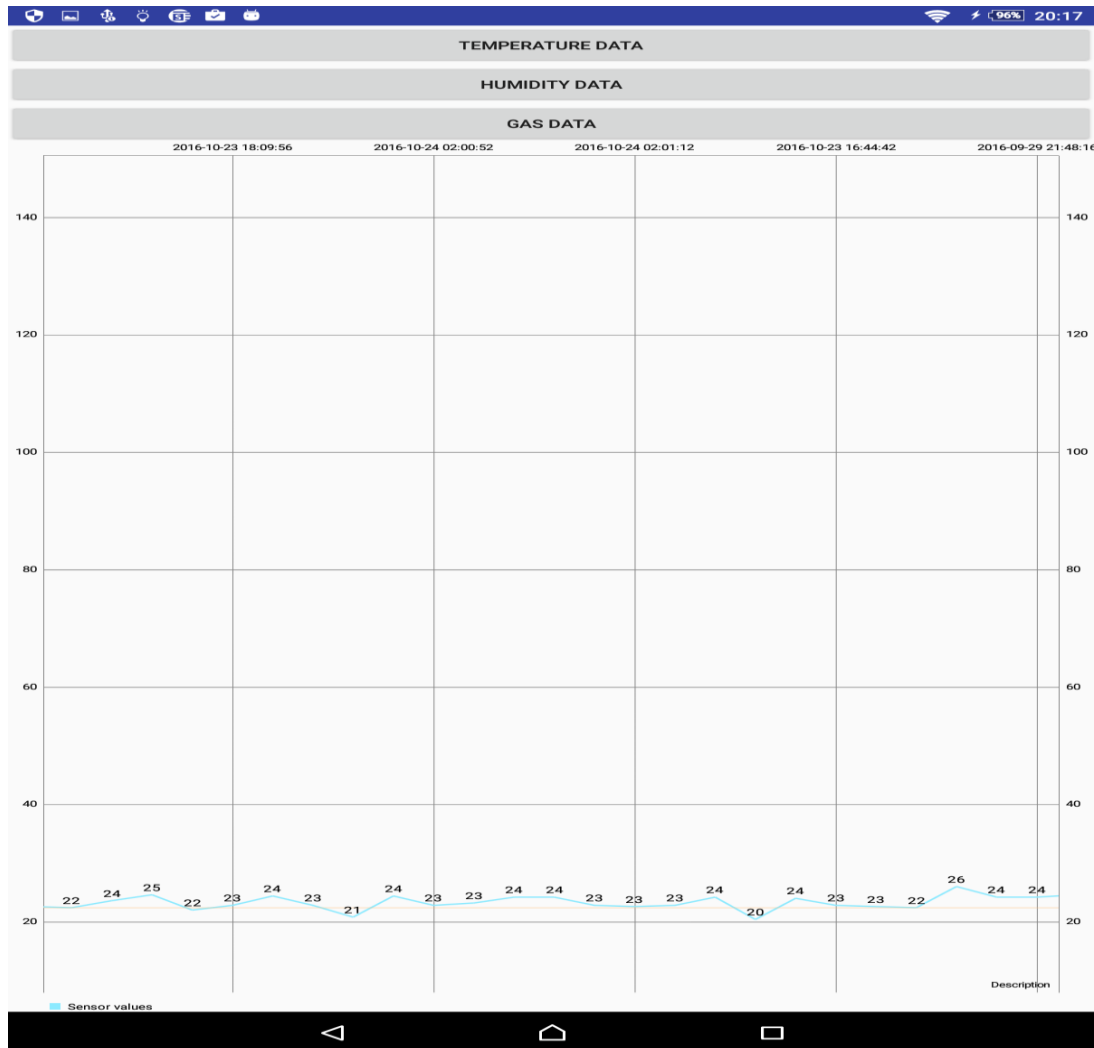


Figure 55 - Sensors' data in graph format

The activity screen, showed in figure 48 shows a graph of all the data from the sensor requested in one of the buttons. The graph is fully interactional; it is possible to zoom in/out and overall navigation. The timestamp can be seen in the X axis which shows the date and time of the measured value.

3.4.3 Audio FFT activity

This screen performs the FFT algorithm to the latest audio file downloaded from the server.

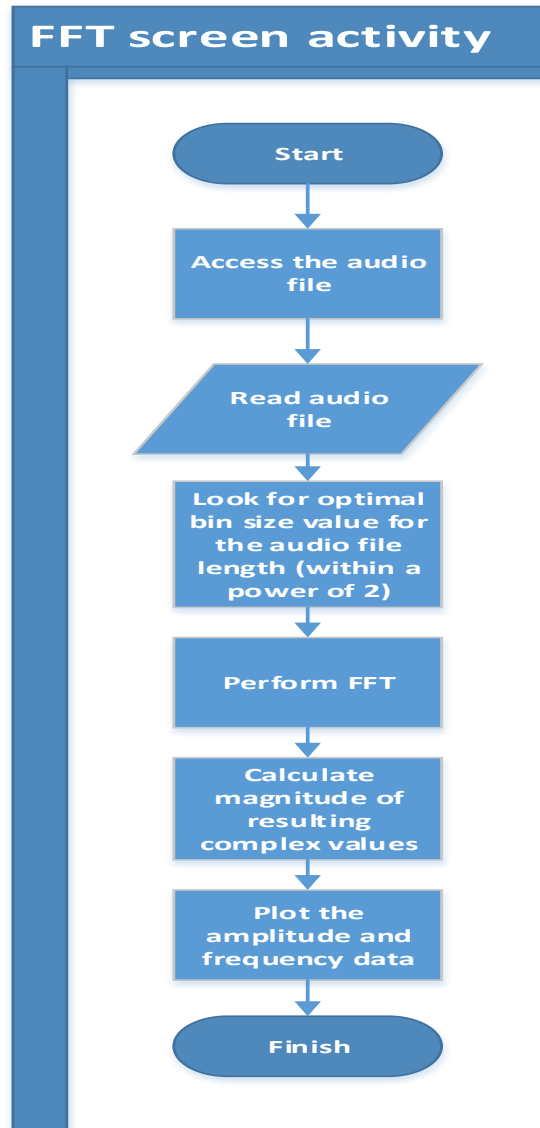


Figure 56 - FFT data Activity flowchart

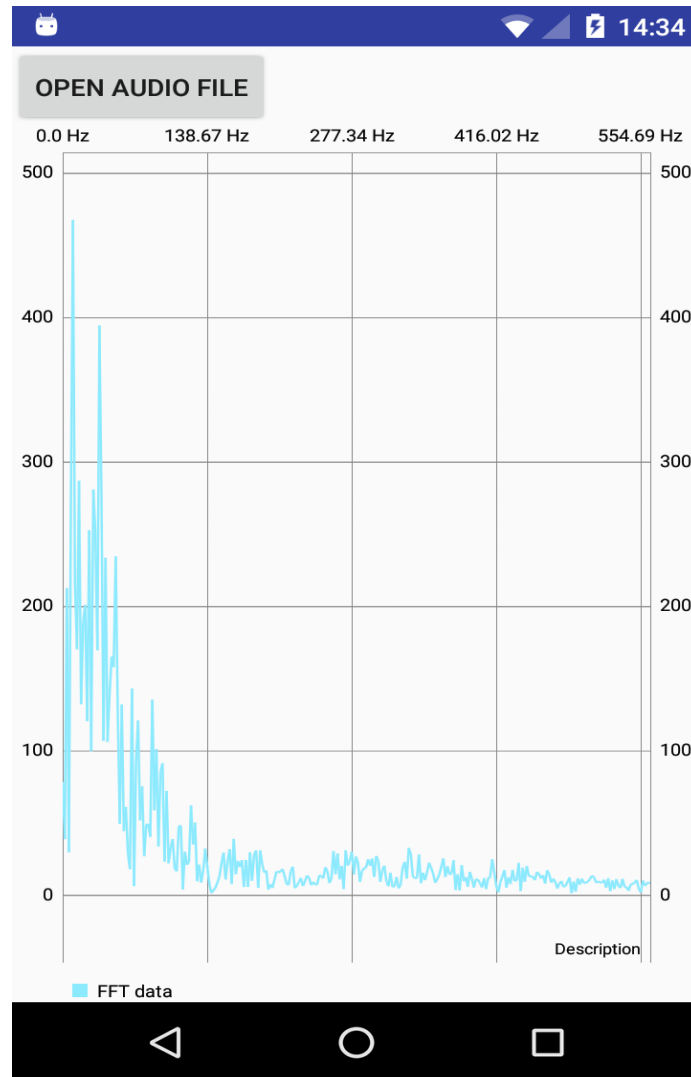


Figure 57 - FFT audio data Activity

The activity plots the FFT data calculated when the stream is stopped and the file downloaded right after.

3.4.4 Audio Waveform (time domain) sound activity

This screen presents a time domain – waveform like form of the audio data, it also accesses the latest audio file downloaded from the server, just like the FFT activity.

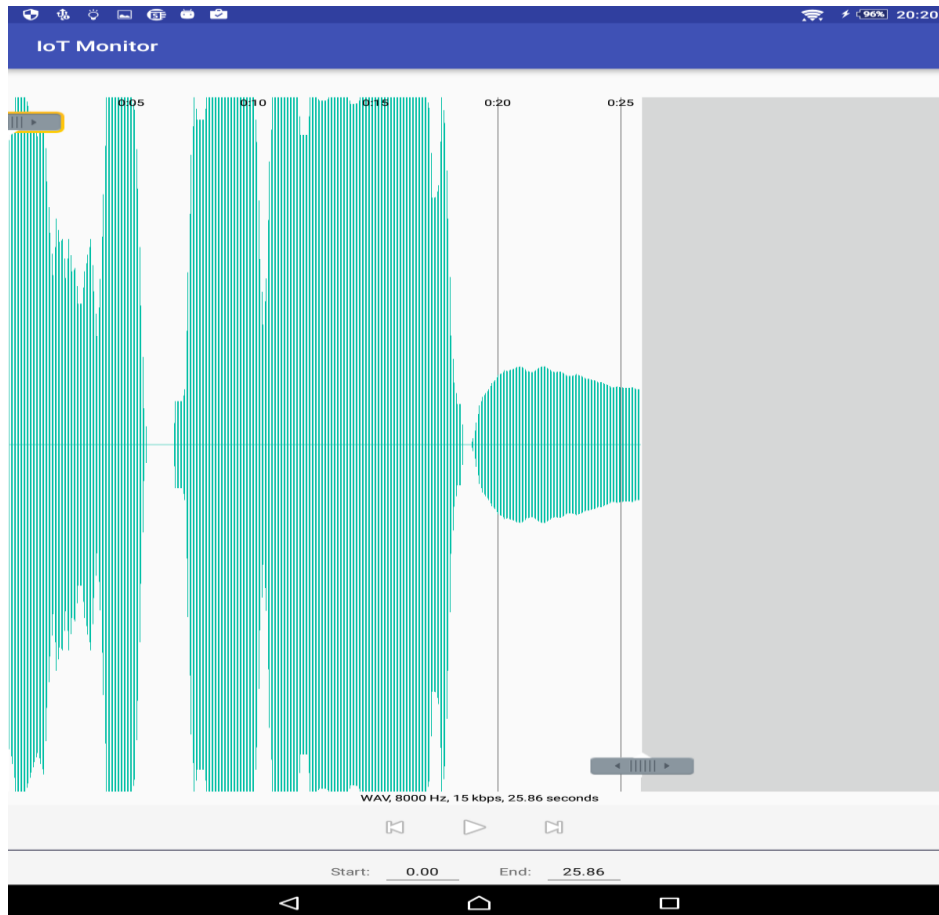


Figure 58 - Audio waveform (time domain)

3.4.5 Mobile Application Code Highlights

To receive the sensors' data, the Main Activity will access the SensorDataParser class that receives and parses the sensors' data to be read in the main activity:

```
@Override
public void getWebResult(String output, String timeStamp, int info) {
    if (info == 0){
        if (temp == null) {
            temp = new Sensor("Temperature Sensor", output+" °C", timeStamp);
            sensorList.add(temp);
            adapter.notifyDataSetChanged();
        }
        temp.setValue(output+" °C");
        temp.setTimeStamp(timeStamp);
        adapter.notifyDataSetChanged();
    }
    else if (info == 1){
        if (hum == null) {
            hum = new Sensor("Humidity Sensor", output+" %", timeStamp);
            sensorList.add(hum);
            adapter.notifyDataSetChanged();
        }
        hum.setValue(output+" %");
        hum.setTimeStamp(timeStamp);
        adapter.notifyDataSetChanged();
    }
    else if (info == 2){
        if (gas == null) {
            gas = new Sensor("Gas Sensor TGS800", output, timeStamp);
            sensorList.add(gas);
            adapter.notifyDataSetChanged();
        }
        gas.setValue(output);
        gas.setTimeStamp(timeStamp);
        adapter.notifyDataSetChanged();
    }
}
```

Figure 59 - Android code snippet of data receiving interface method

This callback interface method receives the values of the sensors and the main activity writes them in the TextViews in the UI. The info variable is just to separate the data, as all the sensors throw the values to this method.

The above-mentioned method is triggered by calling these methods which will create an AsyncTask for each sensor:

```

private void getSensorData(){
    if (tempHandler == null && humHandler == null && gasHandler == null && gpsHandler == null) {
        tempHandler = new SensorDataParser(this, SENSORS_ID);
        humHandler = new SensorDataParser(this, SENSORS_ID);
        gasHandler = new SensorDataParser(this, SENSORS_ID);
        gpsHandler = new SensorDataParser(this, GPS_ID);
        tempHandler.getLatestValue(DataType.TEMP);
        humHandler.getLatestValue(DataType.HUM);
        gasHandler.getLatestValue(DataType.GAS);
        gpsHandler.getLatestValue(DataType.GPS);
    }
    final Runnable r = new Runnable() {
        @Override
        public void run() {
            tempHandler.getLatestValue(DataType.TEMP);
            humHandler.getLatestValue(DataType.HUM);
            gasHandler.getLatestValue(DataType.GAS);
            gpsHandler.getLatestValue(DataType.GPS);
            mHandler.postDelayed(this, REQUEST_DELAY);
        }
    };
    mHandler.postDelayed(r, REQUEST_DELAY);
}

```

Figure 60 - Android code snippet get data method

This method will call the SensorDataParser for each sensor and keep getting the latest value with a delay. The SensorDataParser class has the AsyncTask that fetches the data from the PHP scripts on the cloud server:

```

private String downloadContent (String url) throws IOException {
    InputStream is = null;
    try {
        URL url1 = new URL(url);
        HttpURLConnection conn = (HttpURLConnection) url1.openConnection();
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        conn.connect();
        is = conn.getInputStream();
        return convertToString(is);
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } finally {
        if (is != null){
            is.close();
        }
    }
    return convertToString(is);
}

```

Figure 61 - AsyncTask code snippet

This is part of the AsyncTask and download the data from the files with the link to the PHP scripts passed in the “url” String.

In case of the scripts that return the JSON data to be plotted, the parsing is made in the graphs’ screen activity, showed in the following figure:

```
private void populateChart(HashMap<String, String> values){
    chart.clear();
    test=0;
    ArrayList<Entry> entries = new ArrayList<>();
    ArrayList<String> xAxisTimeStamp = new ArrayList<>();

    for (Map.Entry<String, String> entry : values.entrySet()){
        entries.add(new Entry(Float.parseFloat(entry.getValue()), test++));
        xAxisTimeStamp.add(entry.getKey());
    }

    LineDataSet dataSet = new LineDataSet(entries, "Sensor values");
    dataSet.setDrawCircles(false);
    LineData data = new LineData(xAxisTimeStamp, dataSet);

    chart.setData(data);
    chart.invalidate();
    chart.notifyDataSetChanged();
}

private HashMap<String, String> getHashFromJson(String jsonString){
    HashMap<String, String> jsonHash = new HashMap<>();
    if (jsonString != null){
        try {
            JSONArray values = new JSONArray(jsonString);
            for (int i = 0; i < values.length(); i++) {
                jsonHash.put(values.getJSONArray(i).getString(1), values.getJSONArray(i).getString(0));
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    return jsonHash;
}
```

Figure 62 - JSON data parsing and plotting

These methods, after receiving the data via the AsyncTask interface, will parse the JSON data into a HashMap with the values and the timestamps and then populate a graph with the contents of this HashMap.

In the case of Audio FFT activity, the graph accesses the AudioStream class that calculates the FFT data via the audio file and plots the frequency spectrum and amplitude:


```
private void populateChart(double[] signal, double sampleRate, double binSize){
    chart.clear();
    int test=0;
    ArrayList<Entry> entries = new ArrayList<>();
    ArrayList<String> xAxisFreqs = new ArrayList<>();

    double freqResolution = sampleRate/binSize;
    double res;

    for (int i = 0; i < signal.length; i++){
        res = freqResolution * i;
        entries.add(new Entry((float) signal[i], test++));
        xAxisFreqs.add(res+" Hz");
    }

    LineDataSet dataSet = new LineDataSet(entries, "FFT data");
    dataSet.setMode(LineDataSet.Mode.LINEAR);
    dataSet.setDrawCircles(false);
    LineData data = new LineData(xAxisFreqs, dataSet);

    chart.setData(data);
    chart.invalidate();
    chart.notifyDataSetChanged();
}
```

Figure 63 - Android fft graph code snippet

The method will receive the output signal of the FFT, the sample rate and the bin size and will plot the values accordingly. The x Axis will have the frequency spectrum calculated with the (4) formula and the y Axis will have the amplitude in the signal array.

4 RESULTS

4.1 Field Tests

The tests were conducted in a lake in Campo Grande's garden. After mounting all the hardware in the lab in USV, the resulting system looks like this:



Figure 64 - USV components



Figure 65 - USV with the components installed plus mobile application

Before starting the measurements in the lake, the system was deployed in the water to be ready to go:



Figure 66 – Deploying the USV in the lake



Figure 67 - USV deployed

After deploying the USV in the lake it was time to take a test trajectory, and the trajectory is represented in the following figure:



Figure 68 - USV test trajectory (start red circle; finish yellow circle)

During this trajectory, the USV kept recording data and playing the audio stream, including showing the data in a live format and also playing the audio.

The data acquisition script is started and so is the stream. Also, the live data and GPS trail of markers from the positions of the USV are represented in the following figure:

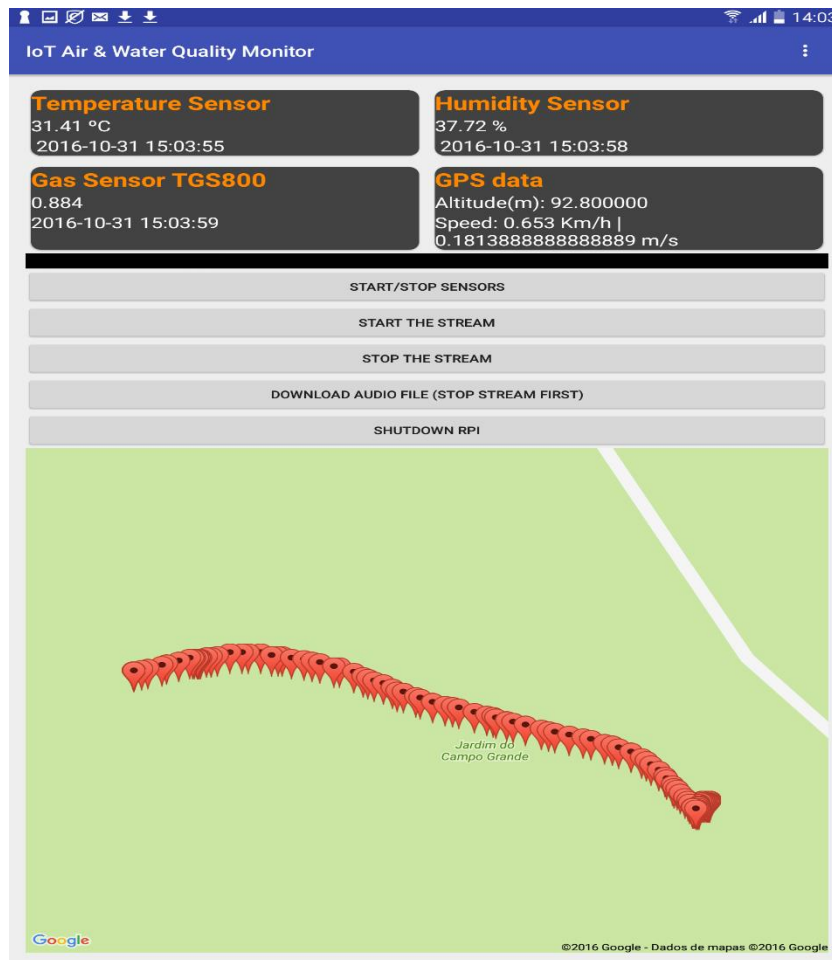


Figure 69 - Main screen after the test

As it shows in the user interface, the trail of markers is traced in the map showing the trajectory of the USV. The GPS device showed a really good precision on providing the coordinates. As the data is being shown in this screen, the stream is also playing at the same time.

After finishing with the tests, it is possible to stop the data acquisition script, the stream and download the resulting audio file, which will allow for time and frequency sound spectrum analysis. In the case of time:

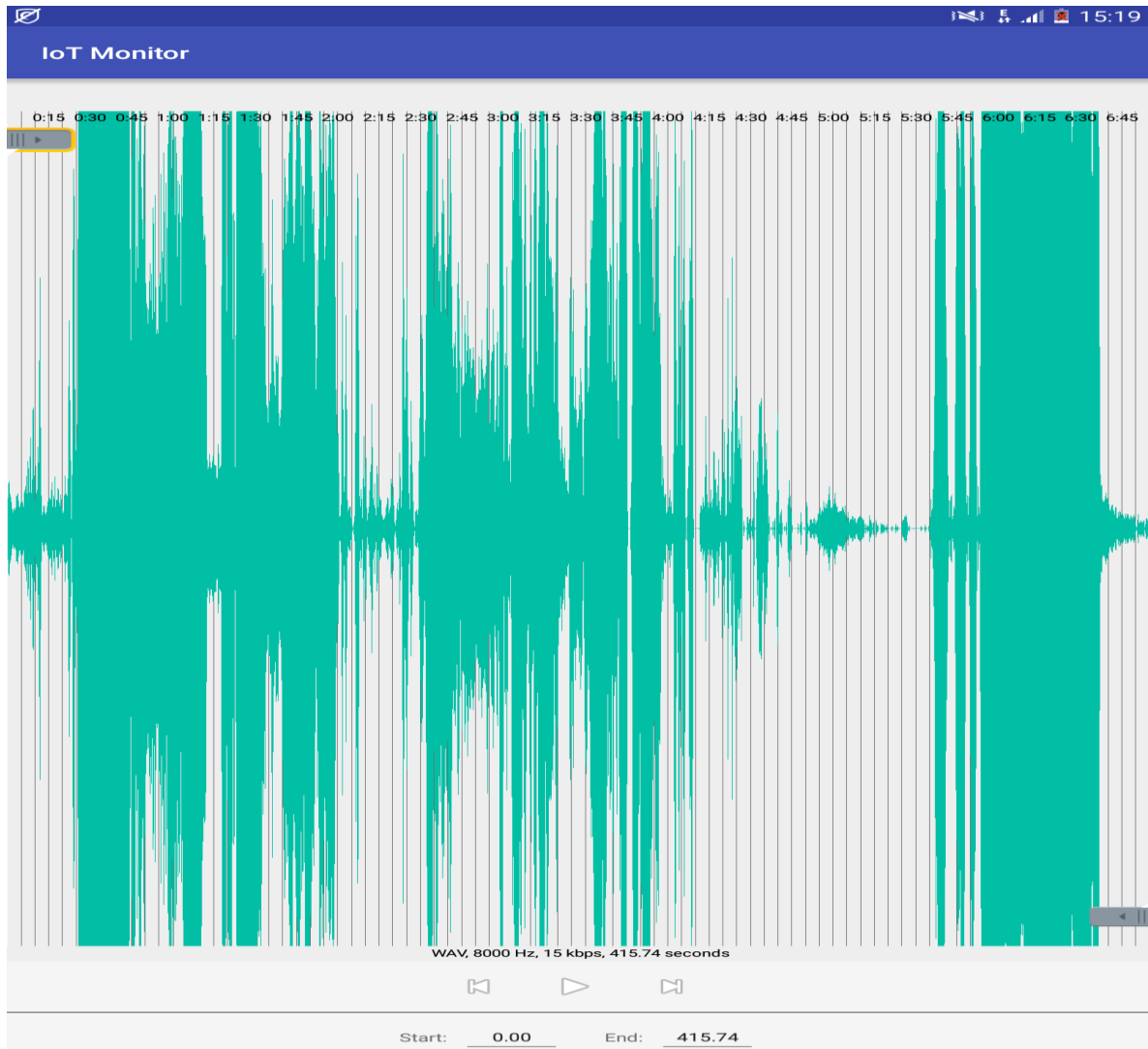


Figure 70 - Audio waveform in the time domain

Now applying the FFT algorithm in the resulting audio file, the result is the following:

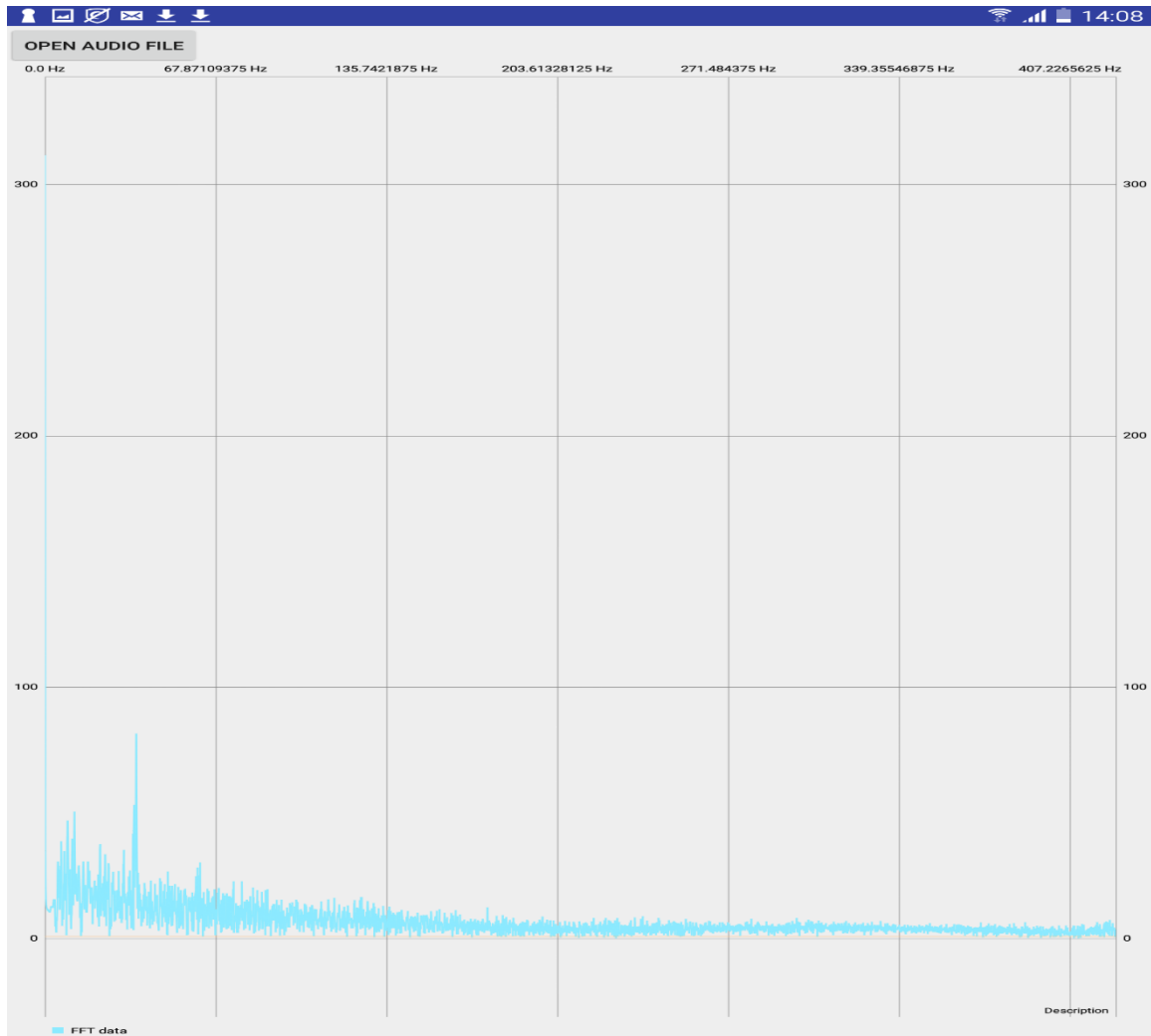


Figure 71 - Audio FFT data from the tests

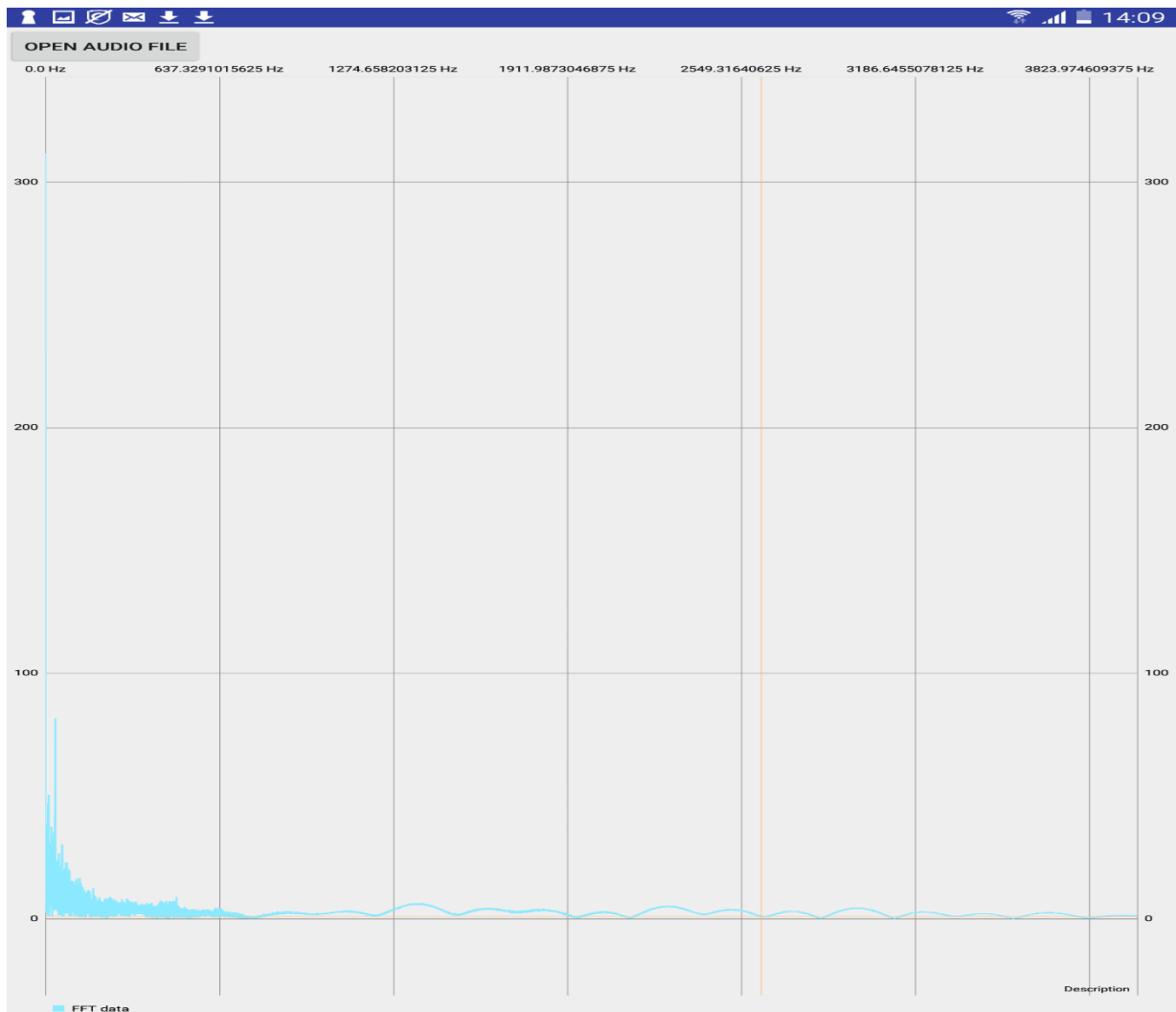


Figure 72 - Audio FFT with wider spectrum

Most of the sound activity is within the lower frequencies (up until ~650 Hz) which shows that there is not much activity in the lake noise wise.

As for the air quality measurements, as the sensors were inside a plastic container and being a bright sunny day it could also heat the plastic, affecting the values in a way. In case of the relative humidity values:

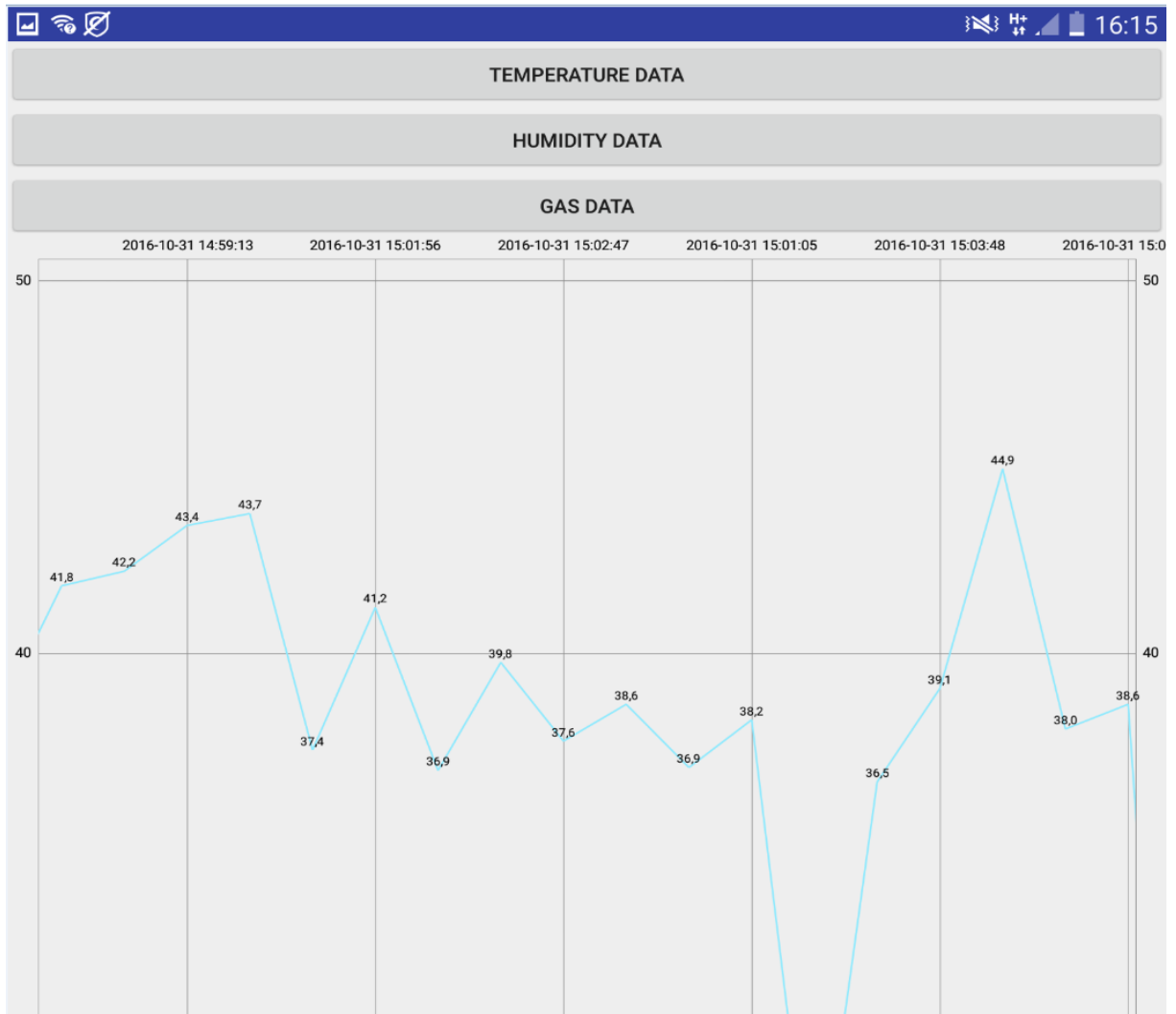


Figure 73 – Relative humidity values (%)

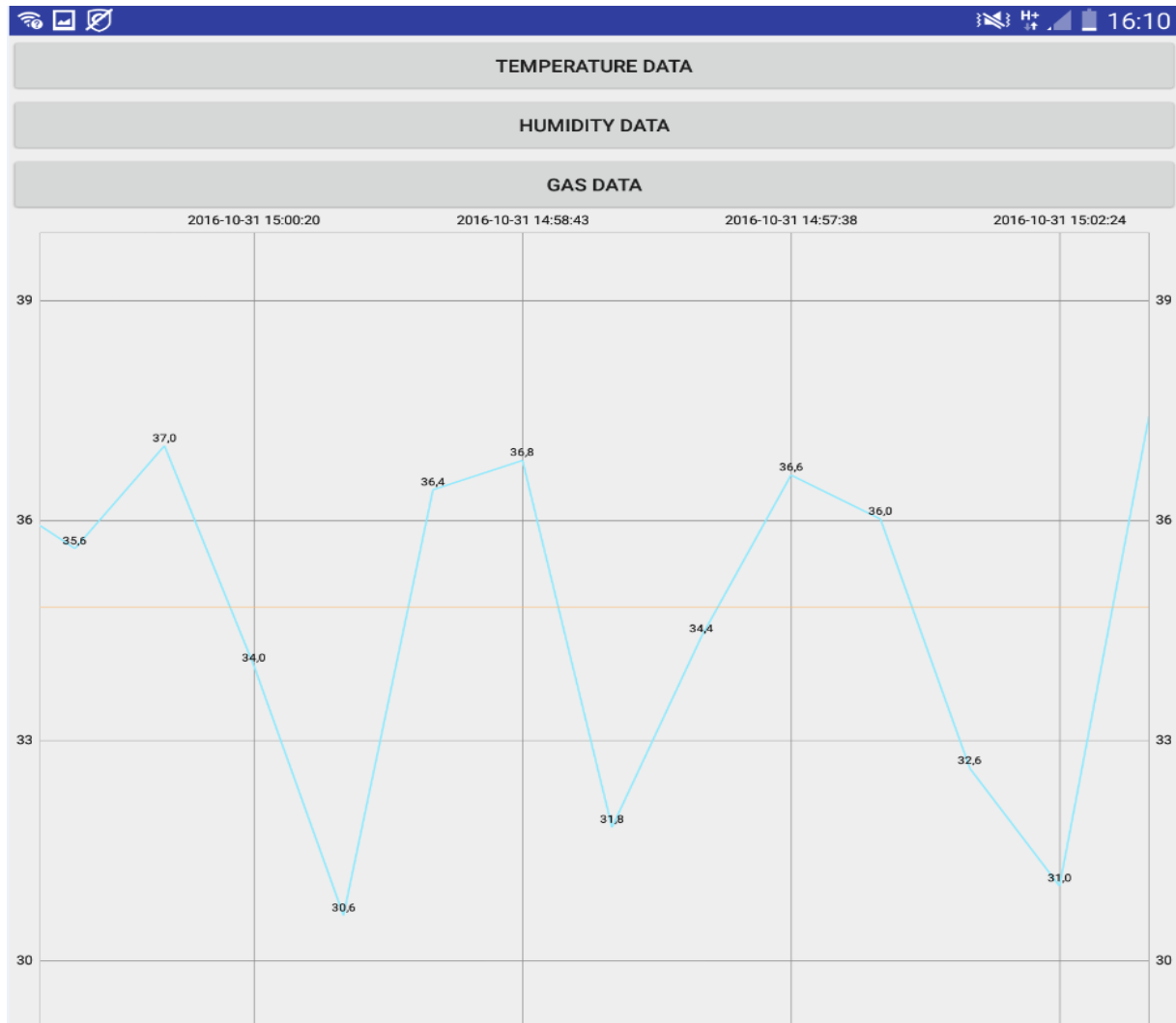


Figure 74 - Temperature values (°C)

The temperature and relative humidity values show a little bit more fluctuations but the values are within the expected range, being a warm sunny day.



Figure 75 - Gas sensor values

The gas sensor values are not affected by the heated plastic as it shows more stable values, going from around ~0.8 to 1 volts showing no alarm as no air contamination is present. It would start to be harmful if the values went above 2 volts.

5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

The goal of this dissertation was to build an environmental monitoring system, more specifically a sensing module to be used with autonomous aquatic drones involved in maritime tasks. So, in short, what was developed in total:

- IoT architecture based on Raspberry Pi 3 platform as the core with air quality sensor measurements,
- Underwater acoustic monitoring architecture associated with a Raspberry Pi computation platform including a software component that provides audio streaming from the client to cloud server to mobile application,
- A Mobile application for data visualization including the underwater acoustic signal time/frequency analysis and Raspberry control.

The system, now concluded, has got three working platforms: the core system of the USV (Raspberry Pi + sensing module), the cloud server with the PHP scripts plus the streaming service and the mobile application with the data visualization UI and streaming playback.

5.2 Future Work

There are still improvements that can be done to provide a better system for environmental monitoring with USVs:

- Extending the water quality measurement sensors' support like turbidity, dissolved oxygen, pH and water conductivity for a wider spectrum of study data,
- Improve the mobile application with a better user interface,
- Development of on-site measurement channel calibration procedures,
- Big data analytics,
- Although some security practices were considered while developing, data encryption was not applied and would be interesting to encrypt the data flows.

REFERENCES

- [1] Velez, Fernando J., Nadziejko, Aleksandra, Christensen, Anders Lyhne, Oliveira, Sancho, Rodrigues, Tiago, Costa, Vasco, Duarte, Miguel, Silva, Fernando and Gomes, Jorge, 2015, Wireless Sensor and Networking Technologies for Swarms of Aquatic Surface Drones. *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*. 2015. DOI 10.1109/vtcfall.2015.7391193. Institute of Electrical and Electronics Engineers (IEEE)
- [2] Zahugi, Emaad Mohamed H., Shanta, Mohamed M. and Prasad, T.V., 2012, Design of Multi-Robot System for Cleaning up Marine Oil Spill. *International Journal of Advanced Information Technology*. 2012. Vol. 2, no. 4.
- [3] Burange, Anup W. and Misalkar, Harshal D., 2015, Review of Internet of Things in development of smart cities with data management & privacy. *2015 International Conference on Advances in Computer Engineering and Applications*. 2015. DOI 10.1109/icacea.2015.7164693. Institute of Electrical & Electronics Engineers (IEEE)
- [4] Spalazzi, Luca, Taccari, Gilberto and Bernardini, Andrea, 2014, An Internet of Things ontology for earthquake emergency evaluation and response. *2014 International Conference on Collaboration Technologies and Systems (CTS)*. 2014. DOI 10.1109/cts.2014.6867619. Institute of Electrical & Electronics Engineers (IEEE)
- [5] Ye, Jiuyan, Chen, Bin, Liu, Qingfeng and Fang, Yu, 2013, A precision agriculture management system based on Internet of Things and WebGIS. *2013 21st International Conference on Geoinformatics*. 2013. DOI 10.1109/geoinformatics.2013.6626173. Institute of Electrical & Electronics Engineers (IEEE)
- [6] Bauer, Harald; Patel, Mark and Veira, Jan; 2014, The Internet of Things: Sizing up the opportunity. *Mckinsey.com* [online]. 2014. [Accessed 12 January 2016]. Available from: http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_internet_of_things_sizing_up_the_opportunity
- [7] Protectplanetoocean.org, Why are oceans important?. [online]. [Accessed 12 January 2016]. Available from: <http://www.protectplanetoocean.org/collections/introduction/introbox/oceans/introduction-item.html>
- [8] Living-rivers.org, Why are natural rivers important?. [online]. [Accessed 12 January 2016]. Available from: <http://www.living-rivers.org/Default.aspx?sifraStranica=64>

- [9] Society, National, Marine Pollution -- Pristine Seas -- National Geographic. *National Geographic* [online]. [Accessed 12 January 2016]. Available from: <http://ocean.nationalgeographic.com/ocean/explore/pristine-seas/critical-issues-marine-pollution/>
- [10] World Scientific Publishing Company, Unmanned Systems. [online]. [Accessed 12 January 2016]. Available from: <http://www.worldscientific.com/worldscinet/us>
- [11] Wikipedia, Uncrewed vehicle. [online]. [Accessed 12 January 2016]. Available from: https://en.wikipedia.org/wiki/Uncrewed_vehicle
- [12] Duarte, Miguel, Costa, Vasco, Gomes, Jorge, Rodrigues, Tiago, Silva, Fernando, Oliveira, Sancho Moura and Christensen, Anders Lyhne, 2016, Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*. 2016. Vol. 11, no. 3, p. e0151834. DOI 10.1371/journal.pone.0151834. Public Library of Science (PLoS)
- [13] Vijayakumar, N and Ramya, R; 2015, The Real Time Monitoring of Water Quality in IoT Environment. 2015 International Conference on Circuit, Power and Computing Technologies. 2015. IEEE Xplore Digital Library
- [14] FUJITA, Isamu and MATSUZAKI, Yoshitaka, 2015, Development of Smartphone-embedded Telemetry Drifting Buoy for Tracking Drifting Oil Patches in the Coastal Water and Field Test. *OCEANS'15 MTS/IEEE Washington*. 2015. P. 1-8
- [15] Waterontheweb.org, Water on the Web | Data |. [online]. [Accessed 12 January 2016]. Available from: <http://www.waterontheweb.org/data/index.html>
- [16] Lakeaccess.org, RUSS - Remote Underwater Sampling Station. [online]. [Accessed 12 January 2016]. Available from: <http://www.lakeaccess.org/russ/>
- [17] BeagleBoard.org - boards, [no date]. *Beagleboard.org* [online]
- [18] Arduino - Products, [no date]. *Arduino.cc* [online]
- [19] Raspberry Pi, What is a Raspberry Pi?. [online]. 2015. [Accessed 29 December 2015]. Available from: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>
- [20] Raspberrypi.org, Raspberry Pi Hardware - Raspberry Pi Documentation. [online]. 2015. [Accessed 29 December 2015]. Available from: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md>
- [21] *Cirrus Logic Audio Card User Documentation*, 2015. [online], pp 1-3. Cirrus Logic, element14.

- [22] Technology, Joseph; SS03-10 Sea-Phone - basic hydrophone - Cetacean Research Technology. *Cetaceanresearch.com* [online]. [Accessed 29 December 2015]. Available from: <http://www.cetaceanresearch.com/hydrophones/ss03-10-hydrophone/index.html>
- [23] SQ26-H1B - basic hydrophone - Cetacean Research Technology, *Cetaceanresearch.com* [online]
- [24] *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*, 2008. [online], pp 1. maxim integrated
- [25] LM35 Datasheet, *ti.com* [online]
- [26] HIH4000 Datasheet, *farnell.com* [online]
- [27] Figaro Gas Sensor TGS2600 Datasheet, *figarosensor.com* [online]
- [28] Figaro Gas Sensors, *jenslabs.files.wordpress.com* [online]
- [29] TGS800 Datasheet, *platan.ru* [online]
- [30] ADS1015 / ADS1115 | Raspberry Pi Analog to Digital Converters | Adafruit Learning System, 2016. *Learn.adafruit.com* [online]
- [31] Adafruit MCP3004/3008 Datasheet, *cdn-shop.adafruit.com* [online]
- [32] Lee, Desmond; Garmin | What is GPS?. *Www8.garmin.com* [online]. 2016. [Accessed 12 January 2016]. Available from: <http://www8.garmin.com/aboutGPS/>
- [33] Wikipedia, Android (operating system). [online]. [Accessed 12 January 2016]. Available from: https://en.wikipedia.org/wiki/Android_%28operating_system%29
- [34] Android Studio Overview | Android Developers. *Developer.android.com* [online]. [Accessed 12 January 2016]. Available from: <http://developer.android.com/tools/studio/index.html>
- [35] iOS - iOS 10, *Apple (Montenegro)* [online]
- [36] Xcode 8 - What's New - Apple Developer, *Developer.apple.com* [online]
- [37] Swift - Apple Developer, *Developer.apple.com* [online]
- [38] About Objective-C, *Developer.apple.com* [online]
- [39] Windows 10 Mobile, *Windows Central* [online]
- [40] IDC: Smartphone OS Market Share, *www.idc.com* [online]

- [41] Comparison of mobile operating systems, *En.wikipedia.org* [online]
- [42] Danielyan, Edgar, 2002, IEEE 802.11 - The Internet Protocol Journal. 2002. Vol. 5, no. 1.
- [43] IEEE 802.11, *En.wikipedia.org* [online]
- [44] IEEE 802.11ac, *En.wikipedia.org* [online]
- [45] 3gpp.org, UMTS. [online]. [Accessed 12 January 2016]. Available from: <http://www.3gpp.org/technologies/keywords-acronyms/103-umts>
- [46] 3gpp.org, HSPA. [online]. 2016. [Accessed 12 January 2016]. Available from: <http://www.3gpp.org/technologies/keywords-acronyms/99-hspa>
- [47] 3gpp.org, LTE. [online]. 2016. [Accessed 12 January 2016]. Available from: <http://www.3gpp.org/technologies/keywords-acronyms/98-lte>
- [48] 3gpp.org, LTE-Advanced. [online]. 2016. [Accessed 12 January 2016]. Available from: <http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced>
- [49] Anacom.pt, 2014, Avaliação da qualidade dos Serviços de Voz, Dados e Cobertura Radioelétrica GSM, UMTS e LTE, nos principais Aglomerados Urbanos e Eixos Rodoviários de Portugal Continental. [online]. 2014. [Accessed 12 January 2016]. Available from: http://www.anacom.pt/streaming/RelatorioPortContinQoS2014.pdf?contentId=1363038&field=ATTACHED_FILE
- [50] Mikhaylov, Konstantin and Tervonen, Jouni, 2012, Evaluation of Power Efficiency for Digital Serial Interfaces of Microcontrollers. *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*. 2012. DOI 10.1109/ntms.2012.6208716. Institute of Electrical and Electronics Engineers (IEEE)

ANNEX

Paper submitted and accepted for presentation in EPE 2016 Iasi, Romania:



Metrology and Measurement Systems:

[Geometry Influence on the Precision of Light Flux Measurement with Ulbricht Integrating Sphere](#)
Cătălin-Daniel Gălăţanu

[Interconnection of Medical Data Acquisition Systems](#)
Lucian Nita

[IoT Enabled Aquatic Drone for Environmental Monitoring](#)
João Matos, Octavian Postolache

[Java Technology Used To Facilitate Measuring Activity Performed By Students In Laboratory](#)
Marius Branzila, Constantin Sarmasanu, Alexandru Mihai

[Modern Water Flowmeters, Differential Pressure Flowmeters](#)
Gabriel-Constantin Sârbu

IoT Enabled Aquatic Drone for Environmental Monitoring paper:

IoT Enabled Aquatic Drone for Environmental Monitoring

João Matos⁽¹⁾, Octavian Postolache⁽¹⁾

⁽¹⁾ISCTE-Instituto Universitário de Lisboa, ISCTE-IUL, Instituto de Telecomunicações,
Avenida das Forças Armadas, 1649-026 Lisboa
email: jrbms@iscte.pt; opostolache@lx.t.pt

Abstract— The article describes a monitoring system based on Raspberry Pi platform and a multichannel sensing module associated with water quality and air quality measurement parameters. Thus the temperature, conductivity, relative humidity and gas concentration are measured as was as the underwater acoustic signals. The data is stored on the memory of the drone's computational platform, and synchronized with a remote server database. Advanced data processing algorithms were implemented on the server side. Additionally, a mobile application was developed to be used by people working in the field for data visualization and statistical analysis.

Keywords—Drones; USV; Sensors; Hydrophone; Raspberry Pi; Streaming.

I. INTRODUCTION

The Internet of Things is an already known term nowadays and it is becoming bigger and bigger overtime with all sorts of sensors and systems being developed to help people ease up their lives. The number of connected devices continues to grow worldwide but also the diversity and the applications in the real world are immense, making it an appealing industry to work on [1]. A lot of companies are working on new devices and new solutions to deal with the growth of the connected devices, it is a massive growing industry. It possesses the power to transform every single environment such as agriculture, transportation, manufacturing, smart houses even entire cities. Companies are working on making new devices for every possible scenario but also improving communication protocols as well as security. It is estimated that around 2015 10 billion devices were connected and by 2020 will be connected 20 to 30 billion devices, as costs continue to drop and demand continues to grow [2].

Rivers and estuaries are incredibly important, they are a source of fresh water and nutrients required by animals and human activity, but with the development of society and industrial demands, the pollution of rivers have become a problem to be concerned about. Food production and pollution have been putting stress within these natural resources and it is necessary to create systems to help maintain the good health of rivers. Other factor that should not be ignored is the noise caused by human activity around the rivers but also by the animals themselves, it is interesting to see the relation of these two and see if outside noise can disrupt natural activities from the rivers [3].

With today's advancements in technology, it became possible to address a larger number of problems regarding accessibility and also monitoring and interacting more thoroughly with systems. This work presents the design and implementation of a system that can be used for water quality monitoring, the system being designed especially for monitoring tasks on rivers and estuaries. The capabilities of the system is based on the usage of unmanned surface vehicle (USV) that helps on maritime monitoring tasks using multi-drone swarms [4] for extended spatial resolution. The system may detect such as cleaning oil spill [5], or for general environmental condition monitoring.

The Internet of Things help to solve a lot of problems in different fields: in smart cities, IoT applications are related with parking issues, noise, traffic, illumination monitoring [6]; emergency systems for earthquakes [7]; precision agriculture applications in culture process optimization [8]. IoT are used to deliver information from the sensors and to the actuators.

The goal of the paper is to present the aquatic drone setup equipped with a Raspberry Pi that is connected to an array of sensors for air and the water quality monitoring. For air quality measurements, temperature, humidity and gas sensors were considered. For water quality monitoring water temperature and conductivity sensing channels were implemented.

Several tasks were carried out during the system implementation. Thus the sensors interfacing with Raspberry Pi platform was considered, different signal conditioning modules were implemented. Referring software Python scripts were developed in order to read the values from the sensors and to write them to a local database. A server setup was carried out to receive the data from the local database implemented on Raspberry Pi level using database replication, in which the server database will always update whenever there are changes in the sensor readings. In order to be possible to access and visualize the data in an appropriate GUI an android application was developed. The applications fetch the data from the server database and provide live readings for the system user.

II. RELATED WORK

The BioMachines Lab in Institute of Telecommunications in ISCTE-IUL, Lisbon is developing a swarm of aquatic drones designed to perform maritime tasks in the sea based on

a Wi-Fi ad-hoc network architecture. Each drone is equipped with a compass, GPS and an Raspberry Pi 2 unit with a Wi-Fi interface and they form a distributed network (controlled by an artificial neural network based controller) without a central coordination, they keep broadcasting their position to the drones in their neighborhood every second. The neural network performs actions depending on what it receives, the sensor readings feed the neural network and then it will control the drone based on the information received [9].

Reported research at the International Conference on Circuit, Power and Computing Technologies (ICCPCT) 2015 consisted of developing of a static system that includes Raspberry Pi B+ model connected to an extra sensors such as: temperature sensor. The Raspberry is part of an IoT module that will send the data to a cloud server, the cloud server then sends the data to a web server with a user interface [10].

In the United States, a project called the Water on the Web (WOW) gathers information from lakes and rivers across the different states of the country, it uses a RUSS (Remote Underwater Sampling Station) which is a floating platform equipped with conductivity, dissolved oxygen, pH, temperature and turbidity sensors and it uses cellular networks to transmit the data to the website [11].

This project resembles in a way to all of these three projects and will sort of connect them together, it is not as thorough as some of them but targets the main subjects.

Drones have come a long way and have all sorts of applications and come in different shapes, UAVs (aerial), USVs (water surface), UUVs (underwater) and UGVs (on the ground) [12][13]. The UAVs are perhaps the most common with their vast application in military and the continuous investments in that way [14]. Nowadays the drone market is targeting more domestic and commercial usage like helping weather forecasting [15], agriculture, movie industry, real estate, local law enforcements, the construction business [16], and much more that can be explored. All of this presents a new large potential to dramatically change people's lives, whole industries and also introducing more privacy concerns and security [17].

III. SYSTEM DESCRIPTION

The developed system based on Raspberry pi platform is characterized by a hardware and software parts. The hardware part of the system is represented in the Figure 1:

The analog sensors are connected to the Raspberry Pi to an ADC board based on ADS1015. The hydrophone channel requires special analog to digital conversion and interfacing capabilities thus an USB Audio Card was employed and connected to a Raspberry USB port.

The software component part and the hardware software relations are presented in Figure 2.

As it is presented in the Figure 2 the Raspberry has a local database that is fetching the data from the sensors, expressed by water and air measured parameters and the respective

timestamps. The local database implemented at the Raspberry pi level is configured to be replicated at the remote server's side database, of which acts like a slave and copies the data from the master 1:1, every change done in the master is replicated by the slave. For the server side a set of PHP scripts were developed, the scrips being accessed by the mobile application that retrieve the sensors' stored data.

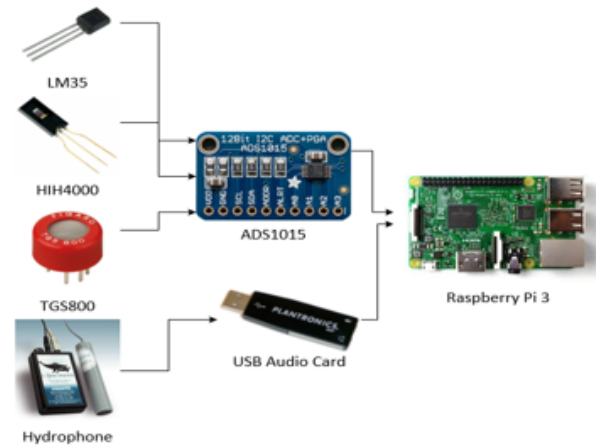


Figure 1 - Hardware System Architecture

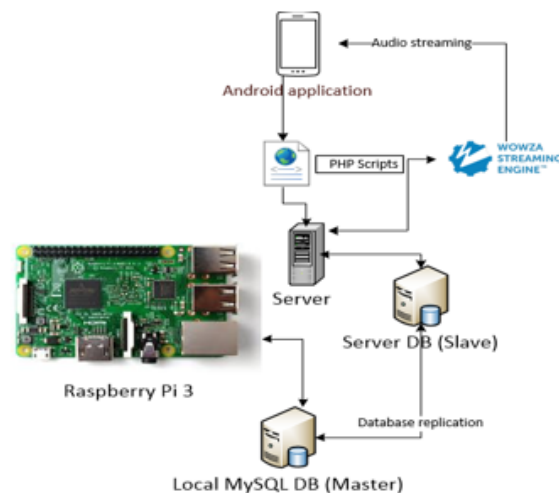


Figure 2 - Software System Architecture

The audio streaming is handled by a Wowza streaming service [18] installed in the server. The service Wowza Streaming Engine facilitates the streaming from the Raspberry Pi to the server and from the server to the mobile application.

As for communication is concerned, a 3G/4G connection is used as it is more viable due to the distances with the drone, being easier to provide internet connection to the Raspberry Pi via a cellular receiver.

IV. HARDWARE AND CONNECTIONS

In this section, the details of the used hardware are provided focussing in acquisition modules, sensors and processing units.

A. Acquisition modules

Since the Raspberry Pi does not includes analog inputs additional hardware is required to acquire the signals from analog sensors. An ADC module (Adafruit model) characterized by I2C communication interface was connected directly to the Raspberry through the GPIO ports with proper configuration. The ADC board is powered by 5V and has 4 analog input (A0-A4) that are used by the analog sensors. Using the python library provided by Adafruit the acquisition control is carried out. Using the developed scrips, the system starts getting information from the sensors' channels.

The Plantronics USB audio adapter features a C-Media chipset which does not required any additional drivers presents full support for arm-based Linux distributions, making it ideal to work with a Raspberry Pi. It allows the acquisition the signals delivered by hydrophone measurement channel.

B. Sensing and conditioning circuits

The LM35 is a compact temperature sensor that can be powered by 5V, it provides an accuracy of 0.5°C for the -55°C to 150°C specified measurement range. The sensor is already calibrated directly in Celsius and it provides an output voltage linearly proportional to the centigrade temperature [20].

The temperature sensor is powered with 5V from the Raspberry and the output is connected directly to the ADC module without additional conditioning required. The implemented voltage to temperature conversion is given by:

$$\text{Temp}(^{\circ}\text{C}) = \text{Vout} - 100 \quad (1)$$

where Vout is the output voltage of the LM35.

This humidity sensor provides voltage values that might be converted in relative humidity (RH) values [21]. The implemented relation to extract the RH values is

$$\text{RH}(\%) = \frac{\text{Vout} - 0.16}{0.0062} \quad (2)$$

where Vout is the output voltage of the sensor and Vsupply is 5V.

The TGS800 gas sensor from Figaro is recommended for air quality control measuring general air contaminants such as: carbon dioxide, ammonia, methane, ethanol and hydrogen through their gas concentrations in ppm. The sensor is characterized by a heater; the applied voltage VH on the heater is responsible for heating up the sensor making it sensitive to the pollution compounds. An additional circuit includes the sensor and a reference resistance. The circuit is powered by 5V and the output pin connects to the ADC board [22].

The ADC module presents a limitation due to being only 12-bit, both humidity and gas sensors easily saturated with the increase of output voltage which would reach the limit of 4.096V from the ADC. In order to solve this problem, a voltage divider was applied to attenuate the output voltage in half and then compensated back at a software level.

V. COMPUTATIONAL PLATFORM

The Raspberry Pi 3 is the core of the system; it deals with all the important data input and output. It receives data from all the analog sensors and the hydrophone and saves the data in a local database and replicates to a remote server and also streams the audio received by the hydrophone.

Several characteristics of Raspberry Pi are: possesses an ARM CPU, graphics capabilities, USB ports, Ethernet port,

HDMI port, programmable ports (GPIO) and a combined audio and composite video jack. Its characteristics made it ideal for this application. Thus it makes it ideal to use in the environment of this project, it is possible to connect all the sensors needed and to communicate with a server to keep uploading data. It also does not require much power (5V, ~1A), allowing it to be attached to a rather small battery for mobile purposes [23][24]. The operating system selected was the Raspbian, the official Raspberry Linux distribution, as it offers official support for specific Raspberry functionalities and not disregarding the advantages of having a Linux kernel.

VI. SOFTWARE

The software component of the system includes embedded software and server software several details regarding backend and frontend functionality within the Raspberry Pi, databases, remote server and the mobile application are described.

A. Embedded platform software

Several scripts were described for the Raspberry Pi. The developed software allows the system to work flawlessly. The used software technologies were Python, MySQL and FFmpeg.

A Python scripts were developed for data acquisition thus Raspberry is able to obtain the data from the measurement channels including sensors. The Adafruit's ADC library was used in this case.

The sensor data is uploaded to the local MySQL database installed in the Raspberry pi SDcard. The script starts by executing a thread for each sensor with a user inputted reading interval; it then keeps reading the data from the ADC module every X seconds and uploading to the database at the same time. At this stage the script only ends with a SIGINT signal sent from the keyboard combination CTRL+C which terminate all the existing threads.

The MySQL database has a straightforward design, a table is created for each sensor and their fields are: id, value and timestamp. This allows the data storage associated with every sensor reading and the respective time of when the reading action was carried out.

To be able to stream the audio from the hydrophone, the FFmpeg software was installed to allow a streaming channel to be made to the server. The FFmpeg receives the audio from the ALSA input microphone channel, which is the general software in Linux to control all the audio/video devices and provides drivers in order for the devices to work in Linux. It then captures the microphone data and creates an RTMP channel that publishes the stream to the remote server.

B. Server side software

The server houses a couple of PHP scripts to allow the mobile application to access the sensors data. A total of 4 scripts were developed, three of them will return all of the saved data from each sensor in a JSON format and the last script will return the latest recorded value for each sensor, depending on the "GET" variable is passed by the mobile application. If a GET string "temp" is passed, the script returns the latest temperature value recorded the same goes for the "hum" and "gas" strings for their respective sensors. The purpose of this last script is to allow the mobile application to keep getting the latest value when it desires to show live readings with a certain interval.

In order of the streaming to work, it is also necessary to configure the server to act as the distributor of the audio stream. It receives a stream publication from the FFmpeg software in the Raspberry Pi and the mobile application will subscribe to that same stream via the same RTMP link. For all this to work, a streaming service from Wowza, the Wowza Streaming Engine was installed in the server to facilitate the streaming process. Wowza generates a link where it is allowed to publish streams with almost any audio/video streaming protocol and at the same time allows for subscriptions that lets a client software to play the audio/video, in this case, the mobile application.

C. Android Application

The android application is at this moment in an early stage, it has a very simple UI that shows the sensors latest reading values in Card Views. It reads the data through connecting to the remote server via HTTP and executing the PHP scripts that will return the data.

VII. PRELIMINARY RESULTS AND DISCUSSION

The implemented system was partially tested by components before installation on a USV. The sensing channels were tested one by one in the laboratory conditions.

For the temperature and relative measurement case the acquired and processed values are presented in the figure 3.

As expected, the values do not vary too much as it is an indoor measurement during the night, temperature was always around ~26°C and humidity always around ~39%.

As for the gas concentrations, the following figure shows the sensor being turned on for a couple of hours and it can be seen that the sensor takes around 2-3 hours to calibrate.

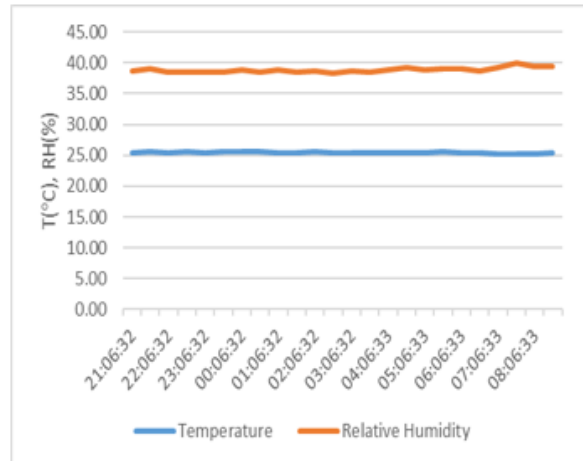


Figure 3 – Temperature(T) and Relative Humidity (RH) Measurements in Indoors conditions

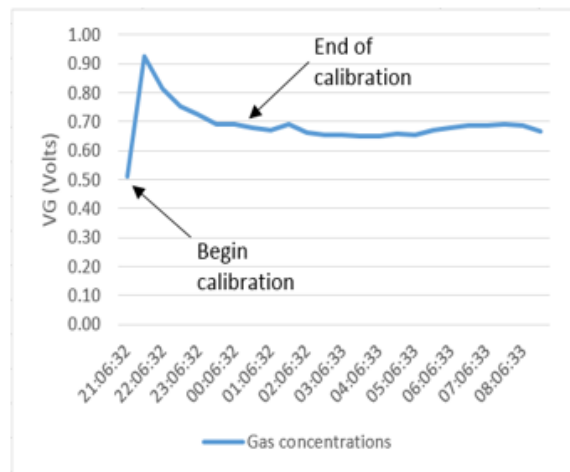


Figure 4 – Gas Sensor Output Voltage (VG) evolution according with indoor gases' concentrations conditions

In this case the ethanol concentration was put really close to the sensor, it explains the high peak value. As the ethanol is removed from the air, the sensor takes time to adjust to normal levels as the rest of the gas inside the sensor evaporates.

Regarding the pollution event detection capabilities of the implemented system an ethanol pollution event was induced the broad band gas sensor response being presented in Figure 5.

Several tests were also performed using the hydrophone mobile software component being designed and implemented to extract information about underwater audio spectral components. In figure 6 is presented the GUI of the underwater acoustic spectrum.

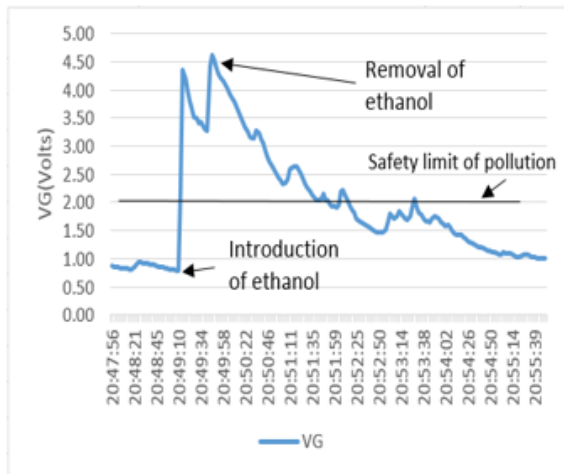


Figure 5- Gas Sensor Output Voltage (VG) evolution with the presence of ethanol

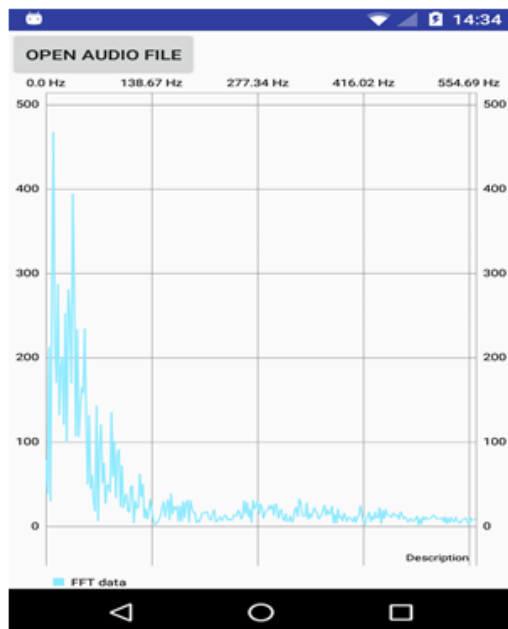


Figure 6 – Spectral analysis of audio signal provided by hydrophone implemented in the mobile APP

In the mobile application case, the GUI associated with on-line monitoring of air and water quality parameters is presented in Figure 7. This is the main screen of the mobile application, where the measured values delivered by sensors are displayed “in cards” and are updated in real-time. The capabilities of the implemented system on pollution event detection were also tested. Thus an ethanol pollution event

was induced and the broad band gas sensor response is represented in Figure 5 .

The mobile application receives the data from the remote server via the implemented PHP scripts. Based on the remote server scripts the signal data after processing can be transferred to the Raspberry Pi.



Figure 7 – Air & Water Quality Mobile APPs GUI

VIII. CONCLUSIONS AND FUTURE WORK

The goal of this paper is to provide information about UAV prototype implementation for water and air quality monitoring. The preliminary data underline the capabilities of the implemented system to provide information about water and air quality for an extended area. Future work will be focussed on mobile application test and improvement regarding functionalities and robustness. On the software side, the mobile application is also developed to provide data analysis and audio streaming. Additionally, field tests related to water and air quality monitoring together the implementation of measurement channel calibration procedures that will assure the appropriate measurement accuracy for the system will be carried out.

REFERENCES

- [1] O'Brien, H. Michael, 2016, The Internet of Things. *Journal of Internet Law*. 2016. Vol. 19, no. 12, p. 1-20.
- [2] Bughin, Jacques, Chui, Michael and Manyika, James, 2015, An executive's guide to the Internet of Things. *McKinsey Quarterly*. 2015. No. 4, p. 92-101.
- [3] Wang, Biao, Lu, Shi-qiang, Lin, Wei-qing, Yang, Yi-fan and Wang, Dao-zeng, 2016, Water quality model with multiform of N/P transport and transformation in the Yangtze River Estuary. *Journal of Hydrodynamics*. 2016. Vol. 28, no. 3, p. 423-430.
- [4] Velez, Fernando J., Nadziejko, Aleksandra, Christensen, Anders Lyhne, Oliveira, Sancho, Rodrigues, Tiago, Costa, Vasco, Duarte, Miguel, Silva, Fernando and Gomes, Jorge, 2015, Wireless Sensor and Networking Technologies for Swarms of Aquatic Surface Drones. *2015 IEEE 82nd*

- Vehicular Technology Conference (VTC2015-Fall)*. 2015. DOI 10.1109/vtcfall.2015.7391193. Institute of Electrical & Electronics Engineers (IEEE)
- [5] Zahagi, Emaad Mohamed H., Shanta, Mohamed M. and Prasad, T.V., 2012, Design of Multi-Robot System for Cleaning up Marine Oil Spill. *International Journal of Advanced Information Technology*. 2012. Vol. 2, no. 4.
- [6] Burange, Anup W. and Misalkar, Harshal D., 2015, Review of Internet of Things in development of smart cities with data management & privacy. *2015 International Conference on Advances in Computer Engineering and Applications*. 2015. DOI 10.1109/icacea.2015.7164693. Institute of Electrical & Electronics Engineers (IEEE)
- [7] Spalazzi, Luca, Taccari, Gilberto and Bernardini, Andrea, 2014, An Internet of Things ontology for earthquake emergency evaluation and response. *2014 International Conference on Collaboration Technologies and Systems (CTS)*. 2014. DOI 10.1109/cts.2014.6867619. Institute of Electrical & Electronics Engineers (IEEE)
- [8] Ye, Jiuyan, Chen, Bin, Liu, Qingfeng and Fang, Yu, 2013, A precision agriculture management system based on Internet of Things and WebGIS. *2013 21st International Conference on Geoinformatics*. 2013. DOI 10.1109/geoinformatics.2013.6626173. Institute of Electrical & Electronics Engineers (IEEE)
- [9] Duarte, Miguel, Costa, Vasco, Gomes, Jorge, Rodrigues, Tiago, Silva, Fernando, Oliveira, Sancho Moura and Christensen, Anders Lyhne, 2016, Evolution of Collective Behaviours for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*. 2016. Vol. 11, no. 3, p. e0151834. DOI 10.1371/journal.pone.0151834. Public Library of Science (PLoS)
- [10] Vijayakumar, N and Ramya, R, 2015, The real time monitoring of water quality in IoT environment. *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*. 2015. DOI 10.1109/iccpct.2015.7159459. Institute of Electrical & Electronics Engineers (IEEE)
- [11] Waterontheweb.org, Water on the Web | Data | [online]. [Accessed 13 July 2016]. Available from: <http://www.waterontheweb.org/data/index.html>
- [12] DiNardo, P.B. and Andrews, S.B., [no date], Unmanned systems initiative: supporting immediate warfighter needs and army transformation. *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication*. 2005.. DOI 10.1109/roman.2005.1513765. Institute of Electrical & Electronics Engineers (IEEE)
- [13] Patterson, Mark R. and Patterson, Susan J., 2010, Unmanned systems: An emerging threat to waterside security: Bad robots are coming. *2010 International WaterSide Security Conference*. 2010. DOI 10.1109/wssc.2010.5730271. Institute of Electrical & Electronics Engineers (IEEE)
- [14] Magnuson, Stew, 2016, Military beefs up research into swarming drones. *National Defense*. 2016. Vol. 100, no. 748, p. 22.
- [15] Elston, Jack, Argrow, Brian, Stachura, Maciej, Weibel, Doug, Lawrence, Dale and Pope, David, 2015, Overview of Small Fixed-Wing Unmanned Aircraft for Meteorological Sampling. *Journal of Atmospheric & Oceanic Technology*. 2015. Vol. 32, no. 1, p. 97-115.
- [16] Unmanned Aircraft Systems Overview, 2016. *Congressional Digest*, Vol. 95, no. 6, p. 2-32.
- [17] Rao, Bharat, Gopi, Ashwin Goutham and Maione, Romana, 2016, The societal impact of commercial drones. *Technology in Society*. 2016. Vol. 45, p. 83-90. DOI 10.1016/j.techsoc.2016.02.009. Elsevier BV
- [18] Media Server Software | Wowza Streaming Engine. *Wowza.com* [online]
- [19] ADS1015 / ADS1115 | Raspberry Pi Analog to Digital Converters | Adafruit Learning System. *Learn.adafruit.com* [online]
- [20] LM35 Precision Centigrade Temperature Sensors. *Ti.com* [online]
- [21] HIH-4000 Series Humidity Sensors. *farnell.com* [online]
- [22] Figaro Gas Sensors. *jens.files.wordpress.com* [online]
- [23] Raspberry Pi, What is a Raspberry Pi? [online]. [Accessed 12 July 2016]. Available from: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>
- [24] Raspberrypi.org, Raspberry Pi Hardware - Raspberry Pi Documentation. [online]. [Accessed 12 July 2016]. Available from: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md>