Indoor Drones
Ostap Kozak
ostap.kozak@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract
Drones are becoming more accessible to everyone. From small drones that are used for fun, to a more sophisticated drones that are used in development, education and in many fields in industry like cinematography and agriculture. Drone development has been a restricted area for people with little knowledge of programming and electronics. The aim of this project is to create a drone development environment for laboratory IT classes at Instituto Superior Técnico, with a given drone; an obstacle avoidance system with the development environment; and an introductory lab work.

This project starts with a research into current open source development tools and obstacle avoidance systems, followed by a research of existing drone, from which the AR.Drone 2.0 is selected for this project. The proposed development environment is primarily composed of the JavaScript and Python programming languages, Node.js framework and ar-drone and gpiozero libraries. A Raspberry Pi Zero W is used to give the ability of adding new components to the drone. The obstacle avoidance system is implemented with 4 ultrasonic sensors. The tests showed that the drone could only avoid obstacles that were in almost direct path with the ultrasonic sensor. This is due to the small detection angle (15°) of the ultrasonic sensor. To introduce the students to the drone’s development environment, a lab work was developed, with the goal of setting up the drone, install the necessary tools, and to implement a simple distance measurement system with the ultrasonic sensor mounted on top of the drone.

Keywords: Drone, AR.Drone 2.0, Development Environment, Obstacle Avoidance

1. Introduction
Drones provide a great opportunity for learning. Teachers can use drones to ignite student’s imagination and improve learning engagement. With drones, students can explore different branches of computer science, physics and mathematics.

Indoor drone’s project aims to develop a drone development environment for IT laboratory classes at Instituto Superior Técnico. Development environment provides an interface and a convenient view of the development process which includes writing code, testing and packaging the build for deployment. The drone development environment can be used as an introduction for programming and electronics as well as a platform for advanced topics such as robotics and real-time systems.

2. Objective
A drone development environment should allow the creation of new applications for the drone both in software and hardware. Specific tools and/or APIs should be provided or developed if necessary, to create the development environment. A research into the available drones should be conducted to select the drone to be used as a platform for app development.

With the selected drone and developed drone development environment, the drone should be equipped with some type of sensors to detect and avoid obstacles. At the end, an introductory lab exercise must be developed to introduce the students to the developed drone’s developing environment.

3. Related Work
In this section a research of development tools for drones and obstacle avoidance systems is conducted in order to gain a deeper understanding and insights on the subject.

3.1. Development tools
There are three popular open source development tools for drones available at the moment of this writing. They are: 3DR DroneKit, Dronecode and ArduPilot. All of them rely on a flight controller board where they are installed. The flight controller is responsible for controlling flight related functions of the drone like precise hovering, flight stability, motor speed, etc. . . . The most popular flight controllers are: Pixhawk, Intel Aero and Qualcomm
Snapdragon. The 3DR DroneKit is made exclusively for the 3DR Solo drone, but its production is now discontinued. The native integration with the 3DR Solo made the drone very popular and powerful because of the simple and complete app development environment and cloud support for the developed applications. These three open source developing tools allow the development of applications for different platforms (Windows, Linux, Mac OS, Android and iOS) using popular programming languages such as Java, Python, C and Objective-C. The supported flight controllers by these three tools are more powerful and complex than what is normally found on regular cheap consumer drones. Thus, they are normally used by professional developers for very specific applications.

3.2. Obstacle Avoidance Systems

Obstacles can be detected with cameras or distance sensors like radar, ultrasound, infrared, laser, etc... One way of detecting obstacles is by inferring space depth from images of a monocular camera [1]. This approach works by detecting relative size changes of the objects in an image, signaling whether the object is getting closer or farther away from the camera. When the size of the object increases at each frame, means that we are getting closer to the object and are in the presence of an obstacle. However, detecting obstacles with a camera is challenging, since there is no motion parallax and optical flow is low for low resolution cameras. In biological vision systems the depth is sensed through motion parallax, monocular cues and stereo vision. This approach is limited to simple obstacle configuration environments. When for example the environment is packed with sharp and small objects, this approach fails to detect the obstacles. Also, cameras depend on external light sources to illuminate the environment, making this approach less reliable in poor light conditions.

To enhance the reliability of the obstacle detection systems, a mix of different types of sensors can be used, as in [2], where custom build 3D laser scanner, 2 stereo camera pairs and ultrasonic sensors were used to detect and avoid obstacles with a drone. The drone with this setup was able to autonomously navigate in dynamically packed environment, and detect sharp and small objects. This system has a big advantage over the single camera setup in obstacle detection accuracy, but as a drawback, it requires a big and powerful drone to carry all of the equipment and a strong processing capacity to process and combine data from the array of sensors.

Laser and radar sensors are expensive, but a precise distance measurement to an obstacle can be achieved by using just and array of cheap ultrasonic sensors, as in [3], where the HC-SR04 ultrasonic sensors were attached to a Parrot Bebop drone to study Kalman filter implementation. The results showed that with ultrasonic sensors and Kalman filter, the error margin of the measured distance values was substantially low. Perhaps the best and simplest obstacle detection and avoidance system is a work of [4]. The system is used in an autonomous flying quadcopter for indoor application, as part of the AQopter18 project at the department of Aerospace Information Technology (University of Wurzburg). The obstacle detection and avoidance approach is depicted in Fig.1.

![Figure 1: Concept for obstacle detection and collision avoidance](image)

The modules are implemented independently from one another. The ultrasonic and IMU raw data are filtered, fused together and sent to obstacle detection module. The collision avoidance module uses that data as well as remote data from a computer or RC controller. The produced data is then used to control the drone’s behavior. In the obstacle detection module, redundant ultrasonic sensors were added to increase sensor reliability and detection resolution, since the ultrasonic sensors have a small dihedral detection angle. A total of 12 ultrasonic sensors were used to obtain a 360 degree view, with 3 sensors for each direction.

The behavior of the obstacle avoidance module is described with a state-machine with 3 states. When the drone is not in the autonomous collision avoidance mode, its state is 0. When the autonomous mode is activated, the drone switches between states 1, 2 or 3, depending on the distance that it is from the obstacle. If there are no obstacles nearby, the drone is in state 1. If an obstacle is detected in the close zone, the drone switches to state 2, and performs pre-evasive maneuvering like reducing the approaching speed and adjusting pitch or roll angle toward the obstacle. If the drone is in dangerous zone, it switches to state 3, and the distance is controlled with PID controller, preventing the drone from approaching the obstacle. This state machine is necessary for every direction. The downside of the simultaneously using 12 ultrasonic sensors is the disturbance that each
sensor experiences from others. More sensors means more noise and errors. The solution to this problem consists in a trade-off between sampling time and accuracy, by using 3 groups of 4 sensors, and activating those sensors with a 90 degree shift angle between them at the same time. Another drawback is that the rotation of the quadcopter influences the ultrasonic measurements, and for that reason readings from the IMU and the gyroscope are used to correct the measurements from the ultrasonic sensors.

Under good circumstances the system was able to reproduce its surrounding environment with an accuracy of a few centimeters. The sensors could also detect a wall through smoke, which is a substantial advantage over the optical sensor systems. However, the ultrasonic sensor could not detect soft surfaces like foam material. The system was able to detect and avoid obstacles, however given the nature of the ultrasonic sensors used, they were only capable of reliably measure distances up to 250 cm. Greater distances and problematic surfaces were not detected, or if detected did not had the necessary reliability. In order to improve the system, it is necessary to improve ultrasonic sensors and fuse them with other systems like infrared sensors, since the ultrasonic sensors alone cannot detect all surfaces.

3.3. Conclusions and thoughts
The researched drone development tools are designed for specific flight controller boards. They support different OS’s and simplify the development process, but the supported flight controllers are expensive and do not with widely popular consumer drones. The Dronecode and ArduPilot support one popular drone, the Parrot Bebop, but they only add and support limited functionalities like flight planning and sensor readings. Other than that, there is not much one can do to add more complex functionalities to the drone, since the drone’s flight controller board is locked and does not allow any hardware expansion. Due to these limitations, these developing tools are not suited for this project. In terms of obstacle avoidance systems, the laser and radar based systems are very expensive, therefore are not considered for this project. Camera and ultrasonic sensor systems on the other hand are great, cheap and easy to use. Camera system has the disadvantage of requiring a good environment lightning condition to accurately operate in. Also, to implement a reliable and accurate system, requires a powerful processing capacity, something that only the most expensive drones have. Ultrasonic sensors are the perfect choice for this project, as they are accurate and simple to implement and use. The work of [4], from the Aerospace Information Technology department of the University of Wurzburg is a perfect starting point for implementing an obstacle avoidance system for this project.

4. Drone Research
In this section a research into 9 different consumer drones is conducted in order to select the one that best fits the project requirements. The drone is required be safe to fly indoors, without causing injuries to others; expandable with additional components, like for example new sensors and have area to place those components; have a flight time of at least 10 minutes and must have an API and/or SDK to allow some sort of communication with other devices. The price is also an important factor to consider, since the lower the price is, the more easy it is to provide the drone to students.

4.1. Parrot Mambo
The Parrot Mambo is small and lightweight racing mini-drone mainly made to be used indoors. It has a LEGO brick-like connection on top of the drone’s body, where the accessories interface with the drone.

The drone is controlled with a phone app or with a Bluetooth controller. The flight time goes up to 9 minutes. In terms of development, the drone is supported by Swift Playground app, which allows the creation of small scripts on an iPad using the Swift programming language [5]. The Simulink graphical programming environment supports the development of flight control algorithms for the drone, which are then deployed wirelessly over Bluetooth to the drone. Python and JavaScript programming languages can as well be used to communicate with the drone, and create custom apps. There is an official SDK [6] that allows the development of new features for the drone. It helps to connect, pilot, receive stream, save and download photos and videos, send and play autopilot flight plans and update the drone. The SDK provides libraries for Unix system, Android and iOS. It also comes with a drone simulator called Sphinx, which is intended to help test applications before flying with the actual drone.

4.2. Parrot Bebop 2
The Parrot Bebop 2 is mainly used to fly outdoors. It reaches 25 minutes of flight. It is controlled over Wi-Fi with a FreeFlight 3 smartphone app or a wireless controller. When controlled over Wi-Fi it can fly up to 300m of range, and 2km with Skycontroller. The built in GPS gives autopilot functionalities in which the user can pin point on the map the coordinates that the drone should follow, and the drone will automatically fly to the selected location. The drone has numerous safety systems, like for example the propeller engine emergency cut-of in case of contact with an object, altitude and perimeter
limitation, defined by the user and an automatic return home feature.

In terms of development tools, there is a SDK for developers to develop new functionality for the drone. There are also open source projects like Paparazzi UAV, Dronecode and ROS that are compatible with the Bebop 2.

4.3. DJI Tello
The DJI Tello is mainly used to fly indoors due to its small size and protective bumpers around propellers. The drone is controlled with a smartphone app via Wi-Fi but can also be controlled with a Bluetooth/Wi-Fi controller. There is an official SDK [7] to communicate with the drone, as well as an educational app called Scratch and Node-RED, that uses visual block-based programming language to program the drone. The SDK gives access to: speed, battery, flight time, height, temperature, IMU attitude data, barometer, acceleration and the drone’s distance to the Wi-Fi controller.

The DJI Tello is equipped with Movidius Myriad programmable video accelerator core and 2 general purpose core VPU’s. It has a downward IR sensor that helps with a smooth takeoff and landing as well as palm lunch and downward obstacle avoidance.

The drone has a large hacking community behind, dedicated to reverse engineer its secrets. There is an unofficial forum, where many developers and investigators share their projects and hacks done with the drone. For example, developers have created a new driver for Gobot [8] (framework using the Go programming language for robotics, physical computing, and the IoT) that enables one to explore the drone’s functionality, and combine it with the other capabilities of the Gobot framework, such as controlling the drone with a PlayStation joystick.

4.4. MakeBlock Airblock
The MakeBlock Airblock is a modular and programmable hexacopter drone. It can be transformed among multiple forms like a drone, a hovercraft, integrated to a LEGO construction or attached with other objects. It has a magnetic connections that enable the creation of different kinds of formations. The goal of the drone is to inspire and introduce kids into STEM education. The drone itself consists of six rotor blades and one master module that links the 6 modules with the propellers with magnetic connectors. Each module is wrapped in special foam to protect the electronics from damage during a crash and also to protect the user from being injured by propellers.

It comes with three integrated LED lights, which are individually controllable; an ultrasonic sensor for altitude measurements; a gyroscope for measuring and maintaining orientation; and a Bluetooth module for connectivity. The drone is programmed with the Makeblock graphical programming software. There are no currently available SDKs or APIs to program the drone other than by using the Makeblock app. The drone is controlled with a smartphone via Bluetooth.

4.5. Syma X26
The Syma x26 is a small obstacle avoiding drone. It uses four IR sensors to detect obstacles, but can only detect them at low speeds and has difficulty detecting dark objects and reflective surfaces. The drone comes with a traditional drone functionalities such as on-key take-off/landing, auto-hovering and flips. The battery of 380 mAh provides 6 minutes of flight and takes 90 minutes to recharge. It can only be controlled with the provided controller, which gives a controlling range of about 30 meters. There are currently no SDKs or APIs that would allow any development for this drone.

4.6. DJI Spark
The DJI Spark is a small drone for indoor and outdoor use. The drone is always aware of its environment, which allows it to safely take-off, land and avoid obstacles when flying.

The drone is able to stream 720p footage in real time up to 2km with a maximum video bit rate of 24 Mbps. It is controlled with a smartphone over Wi-Fi connection, which gives a range of about 100m, or with a remote controller that also uses the smartphone, and adds more precise controls and efficient antennas that increase the range up to 2km. DJI only provides a mobile SDK [9] for the DJI Spark, which means that the drone development is limited to only mobile platforms (Android and iOS).

4.7. Air Selfie 2
The Air Selfie is a pocket-size camera drone. The body of the drone is made of aeronautical anodized aluminium case, which is very robust and lightweight. It features a 12 MP wide-angle (81 degree) camera. The drone is controlled with a smartphone over Wi-Fi, and has a vertical camera and altitude sonar sensor for hovering stability, to take perfectly stable photos. There is no APIs or SDKs available for this drone.

4.8. Micro Drone 3.0
The Micro Drone 3.0 is a small customizable drone. It combines expendable modular technology, sensor-assisted flying and a camera supported by a micro-gimbal. The drone can be controlled with a smartphone over Wi-Fi or with a radio controller over 2.4 Ghz radio signal. The drone has a modular snap-on design used to connect
the camera module. There is no information available at the moment if this modular design can be used to develop custom modules to connect with the drone. The drone allows to use different propeller sizes. It is programmed with three settings to adjust to the different body and propeller sizes. It also supports custom made body frames. There are currently no SDKs or APIs that would allow any development for this drone.

4.9. Parrot AR.Drone 2.0
The AR.Drone 2.0 was designed to be used indoors and outdoors. The drone has a foam around its body to protect it against collisions, when used indoors. When used outdoors, the protective foam can be taken off. The drone is controlled with a smartphone app AR.FreeFlight, and all the communication is done over Wi-Fi. The drone features an ARM Cortex A8 1Ghz 32-bit processor, Linux OS and a USB 2.0 port used to save video recording to a USB pen.

The AR.Drone 2.0 was used in NodeCopter.js event, were developers around the world teamed up to program and hack the drone using the JavaScript language with Node.js framework. There is a lot of documentation, APIs and SDKs available for the drone, both official and from developers from around the world, that enable to take full control of the drone. According to this [10] forum thread, the drone can lift comfortably 200g of additional payload weight.

4.10. Thoughts and Conclusions
All of the researched drones do not allow hardware expansion because their manufacturers made their hardware proprietary. Some of the manufacturers provide SDKs and APIs to expand drone's possibilities in software.

The Parrot Mambo has practically no available space to add new component. It's LEGO brick-like connection could enable the addition of new components, but they would be limited to only 4g. The officially supported accessories are not very energy consuming devices. They are just small motors and don't drain much power from the drone. Also, there is no information on the voltage that the LEGO brick-like connector supplies, thus we cannot assume that the connector could even be capable of supplying enough power to those components.

The Parrot Bebop 2 lacks protection bumpers for its propellers. The motors used in the drone are very powerful and are capable of causing serious damage when in contact with objects or people.

The modularity of the Airblock drone is very interesting and it would be interesting and could be incorporated in various projects. But there are no APIs or SDKs that would allow its development.

The Syma X26 is also a very limited drone in terms of available developing options. There are no APIs or SDKs for this drone that could be used for new application development.

The DJI Spark is a great and complete drone. Due to its size, it is very easy to use it indoors and outdoors at the same time. The big disadvantage of the Spark drone is the fact that all of the sensors and hardware are closed for modifications and expansions, and the fact that DJI only provides a SDK for mobile development. Any app developments are limited to mobile platforms.

The AirSelfie drone has an interesting flat surface that would allow an easy placement of some external hardware like sensors. But the drones motors are not powerful enough to lift any additional payload. Also, there is no APIs or SDKs available for this drone that would allow any new application development.

The Micro Drone 3.0 hardware and firmware are locked and there are no APIs available to communicate with the drone.

The Parrot AR 2.0 drone is perfectly fit for this project. It comes with a foam around that protects the drone against collisions. Due to the drone's body design, it is more robust to the potential damages, and the students can easily and safely learn to fly and work with the drone without worrying for their and others safety. The drone has a large body that can accommodate additional hardware and carry without any problem 200g of additional payload. There is probably more documentation, hacks, applications and experiments with this drone, then any other drone ever produced. The students that are not familiar with the drone can easily learn to fly it, and get right on developing new applications and functionalities.

5. Architecture
The proposed drone development environment architecture is depicted in Fig.2.

The architecture is composed of 4 elements:

- **Drone**: The drone receives maneuver commands over Wi-Fi from the Raspberry Pi or the PC. The drone can only be controlled by one device at a time. The sole purpose of the drone is to accept commands from other devices. The drone cannot be programmed with custom logic.

- **PC**: The user uses the PC to send maneuver commands to the drone. It also acts as a server that listens for any specific data and acts upon it. If the PC is used to send maneuver commands to the drone, then any other device that wishes to also send maneuver commands to the drone, must do it through the PC.
Raspberry Pi: The Raspberry Pi communicates with the drone and the PC. Since the drone does not allow for any new components to be attached to its motherboard, the Raspberry Pi acts as a connection hub to where electronic components are attached to.

Sensors: The sensors are used to augment the drone’s functionalities. Depending on the application, the sensors can be mounted on the drone or be used in other ways. The only limitation is the extra weight that drone is capable of carrying.

All the communication between the drone, PC and the Raspberry Pi is done through Wi-Fi.

5.1. Drone
From the conducted drone research, the Parrot AR.Drone 2.0 was selected for this project.

The AR 2.0 drone has a SDK that enables developers to create new applications for iPhone, iPad, iPod touch, Android devices and PC. The SDK is available at [11]. The official AR.Drone 2.0 developer guide [12] is also available and describes in detail all the functionalities of the drone, how to use the SDK and the drone communication protocols.

The SDK is composed of: AR.Drone 2.0 Library, which provides the APIs needed to communicate and configure the AR.Drone 2.0; AR.Drone 2.0 Tool library, which provides a drone client, where the developers only have to insert their custom code; AR.Drone 2.0 Control Engine library, which provides a control interface for remotely controlling the AR.Drone 2.0 from an iOS device; Code samples for how to control the drone from a Linux computer; Source code for iOS and Android AR.FreeFlight 2.0 applications.

5.2. Raspberry Pi
The Raspberry Pi Zero W is used in this project to communicate with the sensors, drone and the PC. It is small (30 x 65 x 5 mm), has a Wi-Fi module and the whole board only weighs 9 g.

5.3. Programming Languages
New applications for the AR.Drone 2.0 can be developed in JavaScript, Python, C and Java.

JavaScript
The Node.js runtime environment has to be used to enable the execution of JavaScript code outside of the browser. On top of that, the ar-drone library has to be used to be able to communicate with the drone. The ar-drone module exposes a high level client API to communicate with the drone.

Python
The development in Python is accomplished with the python-ardrone library.

C
The official SDK is written in the C programming language. The SDK provides a API for desktop and mobile development (Android and iOS).

Java
The drone app development with Java can be done with two libraries: Autonomous4j and javadrone. Both enable to take full control of the drone, but the last one is also compatible with Android platform.

6. Implementation
In this section, the implementation of the proposed drone development environment architecture is conducted.

6.1. Programming Language
The JavaScript is selected as the main programming language of the drone development environment. With Node.js framework, JavaScript code can be executed in Window, macOS and Linux platforms. Most of the modern web development is done with the JavaScript, therefore it is a popular programming languages with a large developer community behind and with a large number of open source libraries. It is very simple and easy to learn for newcomers. The Johnny-Five library is a popular choice for robotics and IoT projects. And the TensorFlow.js library is a must for machine learning projects.

The Python programming language is selected to communicate with external components connected to the Raspberry Pi’s GPIO pins.
6.2. Software
The developing environment for this project is primarily composed of Node.js framework on top of which runs JavaScript code. There is a library for Node.js called ar-drone that implements the networking protocols used by the AR.Drone 2.0. The ar-drone is used to make the computer and the Raspberry Pi Zero W to communicate with the AR.Drone 2.0. The gpiozero library is used for the Raspberry Pi Zero W to communicate with the components connected to its GPIO pins.

6.3. ar-drone library
The ar-drone is a JavaScript library for Node.js that implements the networking protocols used by the AR.Drone 2.0. It allows any device running Node.js to easily communicate with the drone over Wi-Fi.

The ar-drone library provides a high level Client API to communicate with the AR.Drone 2.0. Under the hood, the library sends flight commands to the drone UDP on port 5556, 30 times per second to guarantee a smooth user experience.

The library also gives access to the drone’s navigation data (like its status, position, speed, motor's speed, etc.). By listening on the navdata stream on UDP port 5554, the drone sends the navigation data approximately 15 times per second in demo mode, and 200 times per second in full mode. The demo mode contains reduced navigation information, whereas the full mode contains all the navigational data, which is useful for debugging. Other available events: landed, hovering, flying, landing, batteryChange and altitudeChange.

6.4. gpiozero library
The gpiozero library is an interface to the Raspberry Pi's GPIO pins. It provides a range of different component interfaces that facilitate communication with different types of electronic components. The gpiozero library uses the BCM pin numbering for the GPIO pins. The full library documentation is available at https://gpiozero.readthedocs.io.

6.5. Raspberry Pi
If the Raspberry Pi is connected to a monitor, keyboard and mouse, the development environment setup is actually pretty simple. After installing the Node.js and ar-drone library, one can use whatever developing editor he/she desires, like for example: Atom or Visual Studio Code. When there is no monitor connected to the Raspberry Pi, there are primarily 2 ways to interact with the Raspberry Pi: VNC or ssh connection.

The VNC is a graphical desktop sharing system, that allows to control one machine from another. The ssh remote development is accomplished with the remote development extension for the Visual Studio Code editor. It allows to connect to the Raspberry Pi over SSH and write code in the main computer and execute and debug the code at the Raspberry Pi.

7. PC Drone Control
The AR.Drone 2.0 accepts commands via Wi-Fi communication. The drone creates a WLAN to which any Wi-Fi enabled device can connect to. The PC runs a Node.js application that interacts with the drone using the ar-drone library. Every 30 milliseconds, the library sends maneuver commands to the drone over UDP protocol on port 5556. This is because according to the drone’s SDK guide, to provide an optimal user experience the messages must be sent to the drone at an interval of 30 milliseconds.

8. Raspberry Pi Drone Controller
Raspberry Pi is a small computer, which means that the same control script written in JavaScript for PC, will also work in Raspberry Pi. The Raspberry Pi must be connected to the drone’s WLAN network, to be able to send the commands.

9. Raspberry Pi and PC communication
The Raspberry Pi communicates with the PC over Wi-Fi. The system architecture is depicted in Fig.3. There are two main components in this system: PC and Raspberry Pi. The PC is the server that just receives the data and acts upon it. The server is built with Node.js and Express web framework. The Raspberry Pi just sends data to the PC over HTTP protocol.

10. Obstacle Avoidance
In this section an obstacle avoiding system is implemented, using the developed drone development environment. It starts by defining the obstacle avoidance system architecture, followed by it’s implementation.

10.1. System Architecture
The obstacle avoidance system architecture (Fig.4) is composed primarily of the Raspberry Pi...
Zero W, AR.Drone 2.0, PC and four HC-SR04 ultrasonic sensors.

The AR.Drone 2.0 receives manoeuvre commands from the PC. It creates a WLAN, to which the Raspberry Pi Zero W and the PC are connected.

The Raspberry Pi Zero W has four HC-SR04 ultrasonic sensors connected to it. It reads distance values from ultrasonic sensors and sends them to the server (PC). The Raspberry Pi Zero W and the four ultrasonic sensors are placed on top of the drone, with one ultrasonic sensor for each direction (left, right, front and back).

The PC is the server that receives distance values of the ultrasonic sensors from the Raspberry Pi Zero W. It also sends the manoeuvre commands to the AR.Drone 2.0, having the user to input these commands through a keyboard. The obstacle avoidance logic is located on the server. While the server is constantly receiving distance values from the Raspberry Pi Zero W, at the same time, it decides if any actions are necessary to prevent the drone from crashing into an obstacle.

The flow of the events happens in the following order: The Raspberry Pi Zero W reads the distance values from 4 ultrasonic sensors (1). These distance values are sent to the server. The server constantly checks the distance values received, and decides if any evasive actions are necessary to avoid the drone from crashing into an obstacle (2). At the same time, the server accepts manoeuvre commands from the user through the keyboard and sends them to the drone. The drone is constantly waiting for commands from the server. When it receives them it acts immediately on them.

10.2. Evasive maneuvers
The evasive maneuvers are based on a notion of distance zones. There are 3 distance zones defined: safe, danger and critical, Fig.5. Each zone is based on the distance that the drone is to the obstacle. The obstacle avoidance system is best described with a state machine, Fig.6. The drone transitions between 3 states (Safe, Danger and Critical), depending on the measured distance to the nearest obstacle. The drone is in Safe state when the drone’s distance to the obstacle is greater than $b + c$. The drone is in danger state when the distance to the obstacle is greater than $c$ but lesser than $a$. In this state, the drone gradually decreases its speed towards the obstacle. The drone is in critical state when the distance to the obstacle is lower than $c$. In this state the drone maneuvers in the opposite direction of the initial movement. This system is applied for each sensor direction.

11. Lab Work
In order to introduce the students into the drone development environment, a lab work was developed. The lab works starts by introducing the students guided through the installation of the Node.js framework, ar-drone library and configuration of the Raspberry Pi Zero W, in order to be able to communicate with the AR.Drone 2.0. As an exercise, the students are asked to connect the ultrasonic sensor to the raspberry pi zero w and mount them on top of the drone to make it measure the distance to a wall.

12. Evaluation
12.1. Flight Time
To measure the flight time in free flight mode, 5 test runs were performed with and without the foam protector. The foam protection weighs 40 g. The average flight time with protective foam was 11.25 minutes, whereas without foam protective was 9.57 minutes. This corresponds to a 14.93% reduction
in flight time when the protective foam is used. The advertised flight time is 12 minutes which is almost the time that was obtained when tested without the protective foam.

12.2. Obstacle Avoidance Evaluation
To measure the obstacle avoidance performance, a series of tests were performed in 6 x 4 x 3 m room. The system’s ability to avoid crashing into the obstacle was tested, and the system was stressed with different environment conditions. Mainly the different obstacle positions were tested to understand the system’s detection accuracy.

The first test set tested how the obstacle position influences the obstacle detection accuracy. The test results showed that the obstacles could have been only detected when the obstacle was in direct sight with the sensors. This is because the HC-SR04 ultrasonic sensor has a 15 degree detection angle. And because only four ultrasonic sensors were used, they did not cover the full 360 degree angle.

The second test, tested if the drone could navigate from point A to point B, in room layout defined in Fig.7, without crashing into a wall. The user conducting the test directed the drone in a collision course with a wall along the way to point B.

The drone was able to navigate from point A to point B 1 out of 5 times. This represents a success rate of 20%, which is considered to be not acceptable value. The times when the drone failed to avoid the obstacle was because the obstacle was not in the 15 degree angle detection zone of the ultrasonic sensors.

The components used to implement the obstacle avoidance system totaled 124 g of additional weight that the drone had to lift. From the 5 test flights conducted and with the Raspberry Pi draining battery from the drone’s internal battery, the drone stayed on average 7.02 min in the air. This represents a 37.6% reduction in flight time compared to the flight time without a protective foam, and 26.6% compared to the flight time with a protective foam, without any additional weight.

13. Conclusion
This project aimed to develop a drone development environment by using a commonly available and affordable drone, to be used by students and teachers in IT laboratory classes at Instituto Superior Técnico. With the developed drone development environment, a drone obstacle avoidance system had to be developed.

A research into the currently existing tools and platforms for drone development, commercial drones and obstacle avoidance systems was conducted. The available developing tools for drones (3DR DroneKit, Dronecode and ArduPilot) are mostly available for more expensive drones, and drone enthusiasts that build their own drone platforms with custom components and expensive modules. Thus, none of those tools were used in this project. After researching 9 drones, the AR.Drone 2.0 was selected to be using in this project due to its large developing community that has developed a large number of tools, libraries and projects for the drone. For the big available area to place any new components and to carry an astonishing amount of 200g of additional weight, and also for the protective foam around its propellers, to protect the users from being injured and the objects around it from being damaged.

From the research of different obstacle avoidance systems, was learned that there are different types of sensors that can be used to measure distance to the obstacles. Primarily there are IR, laser, radar and ultrasound sensors. The laser and radar sensor are very precise in measuring the distances, but are very expensive to use. On the other hand the IR and ultrasound sensors are cheap, but are not as precise as the IR and laser sensors. The ultrasound sensor in comparison with the IR sensor is more reliable for this project because it works in more environments, is not influenced by the light conditions, as opposed to the IR sensor, and has a large detection angle and measuring distance.

A drone development environment was developed having the hardware and software development in mind. In the software aspect, the development is accomplished using the JavaScript programming language with the Node.js framework. The Node.js allows the execution of JavaScript code outside of the browser. The already available ar-drone library for the Node.js framework was selected to communicate with the AR.Drone 2.0. This library implements the communication protocol of the drone and exposes a high level API client to facilitate the communication with the drone. In terms of hardware development, it is not possible to simply add new components to the drone’s motherboard and make it work seamlessly. For that reason the Raspberry Pi Zero W board is used, to
where any new hardware components can be connected to its GPIO pins. The Raspberry Pi board has a Wi-Fi module which also allows it to communicate with the drone using the Node.js framework and ar-drone library. The communication with new hardware components is done with the gpiozero, written in python programming language, and provides a set of interfaces and abstraction layers to facilitate the communication with low level hardware components.

The obstacle avoidance system was developed using the proposed drone development environment. The main idea of the obstacle avoidance system is the notion of distance zones. There are 3 distance zones defined: safe, danger and critical. Each one represents the distance that the drone is to an obstacle, in four directions (front, left, right and back). In total, 4 ultrasonic sensors were used to measure the distances. The distance values were then sent to the server by the Raspberry Pi Zero W. The server maintains a direct communication with the drone, and controls it when evasive manoeuvres are necessary. The user has access to the server, that uses it to send manoeuvre commands to it, through a keyboard.

From tests that were conducted, the drone was only able to avoid obstacles that were in direct sight with the sensors. The ultrasonic sensor has only 15 degree detection angle, which is very small and does not allow the full 360 degree view angle. When tested in packed environment and with obstacles not in sensors detection angle, the drone crashed into them. To overcome this limitation, more sensors are needed to be used.

To introduce the students to the developed drone development environment, a lab work was developed. It is primarily structured to introduce to the development tools and software, how to set everything up and in the end the student is asked to be connected an ultrasonic sensor to the raspberry pi zero w, place them on top of the drone and measure distances to a wall.

References


