FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Quad-copter platform for civil applications

Gustavo Pinho Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Rosaldo Rossetti

Co-supervisor: Eng. Lúcio Sanchez Passos

Co-supervisor: Eng. Zafeiris Kokkinogenis

June 16, 2014

Quad-copter platform for civil applications

Gustavo Pinho Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Professor Carlos Manuel Milheiro de Oliveira Pinto Soares
External Examiner: Professor Artur José Carneiro Pereira
Supervisor: Professor Rosaldo Rossetti

June 16, 2014

Abstract

Unmanned aerial vehicles (UAVs) are gaining the attention of researchers around the world, due to their maneuverability and performace in both indoor and outdoor environments.

This dissertation documents the specification, implementation and test of a modular, extensive and flexible architecture to autonomous UAVs. It aims to provide a testbed for researchers where they can rapidly prototype the application of new methods without the effort of reimplementing all the infrastructure.

The document contains the background in the relevant topics and related works reviews, a detailed description of the approach taken, tests and the consecutive results and also some conclusions about the final solution.

The proposed architecture consists of a hybrid design with three layers where each layer has a different level of cognition, different functions and different requisites. The method was implemented and tested in a quadcopter.

The robot was designed with indoor civil applications in mind, however because of resources constraints only simple maneuvers were considered. Nonetheless with more time the implementation could be extended to allow for actual applications like surveillance or exploration.

The report finishes concluding that although the project was very changelling important steps to achive autonomous indoor flight were taken. Also the outcomes of the project resulted in a software stack that is already being applied within other research groups with no prior connection to this disseration.

Keywords: Robotics; Agents; Quadcopter; Localization; System Architecture

Resumo

Quadcopter para aplicações civis

Veículos Aéreos Não Tripulado (VANT) estão a ganhar a atenção de investigadores à volta do mundo, devido a sua manobrabilidade e desempenho em ambiente interiores e exteriores. Esta dissertação documenta a especificação, implementação e teste de uma arquitectura modular, extensiva e flexível para VANT autónomos. Esta tenta providenciar uma ferramenta de teste para investigadores onde estes possam rapidamente testar novos métodos sem o esforço de reimplementar toda a estrutura.

O documento contém uma pesquisa de conceitos nos tópicos relevantes e uma revisão dos trabalhos relacionados, uma detalhada descrição do método utilizado, testes e consecutivos resultados e também algumas conclusões sobre a solução final.

A arquitectura proposta consiste num desenho híbrido com três camadas onde cada uma contem diferentes funções, diferentes requisitos e diferentes níveis de cognição. O método é depois implementado e testado num *Quadcopter*.

O *robot* foi desenhado com aplicações civis de interior em mente, contudo devido a limitações de recursos apenas manobras simples foram consideradas. No entanto com mais tempo a implementação poderia ser estendida para permitir verdadeiras aplicações como vigilância ou exploração.

O relatório termina concluindo que apesar do projeto ter sido muito desafiante passos importantes para atingir voo autónomo no interior foram dados. Também os produtos paralelos do projecto resultaram num conjunto de aplicações em *software* que estão já a ser utilizadas dentro de outros grupos de investigação sem qualquer ligação inicial com esta dissertação.

Palavras-chave: Robótica; Agentes; *Quadcopter*; Localização; Arquitectura de Sistemas

Acknowledgements

First and foremost I would like to thank my co-supervisors, Lúcio Sanchez Passos and Zafeiris Kokkinogenis and my supervisor Professor Rosaldo Rossetti, who never abandoned me and shed tears and blood almost as much as myself.

Also, I would like to remark the important companionship provided by Rúben Veloso, other dissertationist at my laboratory, which never refused to help even when his own schedule was tight.

There is an enormous list of persons which without who this project would have been impossible, among them:

- Francisca Melo Lopes Barreto
- Professor Armando Luís Sousa Araújo
- Professor Joaquim Gabriel Magalhães Mendes

Finally I would like to thank both my friends and family for all their support. In spite of spending more time destroying than helping they kept me happy and motivated.

Gustavo Pinho Oliveira

"Opportunity is missed by most people because it is dressed in overalls and looks like work"

Thomas Edison

Contents

1 Introduction **1**
- 1.1 Research Scope and Motivation . . . 1
- 1.2 Research Problem, Aim and Goals . . . 2
- 1.3 Dissertation Structure . . . 4

2 Literature Review **5**
- 2.1 Background . . . 5
  - 2.1.1 Autonomous Agents . . . 5
  - 2.1.2 Mobile Robotics & Autonomous Vehicles . . . 8
  - 2.1.3 Quadcopter Model . . . 9
  - 2.1.4 Localization and State . . . 11
- 2.2 Related Works . . . 15
  - 2.2.1 Quadcopters research and development . . . 15
  - 2.2.2 Agent design applied to autonomous vehicles . . . 18
- 2.3 Summary . . . 19

3 Methodological Approach **21**
- 3.1 Problem Formalization . . . 21
  - 3.1.1 Assumptions . . . 22
  - 3.1.2 Functional Requirements . . . 23
- 3.2 System Overview . . . 24
  - 3.2.1 Reactive Layer . . . 24
  - 3.2.2 Executive Layer . . . 26
  - 3.2.3 Deliberative Layer . . . 30
- 3.3 Summary . . . 31

4 Test & Results **33**
- 4.1 IR sensors integration . . . 33
- 4.2 Localization . . . 34
- 4.3 Jason . . . 35
- 4.4 Summary . . . 37

5 Conclusion **39**
- 5.1 Final Remarks . . . 39
- 5.2 Further improvements . . . 41
- 5.3 Future works . . . 41
- 5.4 Lessons Learned . . . 41

References **43**

A Class Diagrams **49**
A.1 UAVTalk Library 49

I Spreadsheets **51**
I.1 GP2Y0A02YK Infrared Sensor 51
I.2 LV-MaxSonar-EZ0 Sonar Sensor 55

List of Figures

2.1 Minimum spanning for quadcopter classification 9
2.2 Coordinate systems and forces/moments acting on the quadcopter 10
2.3 Correlation between rotor blades speed and quadcopter motion 11
2.4 Kalman filter overview 13
2.5 Representation of the Sampling Importance Resampling (SIR) algorithm 14

3.1 Proposed Approach Architecture 25
3.2 UAV stabilization PID 26
3.3 Infrared voltage divider 27
3.4 Taulabs GCS input configuration 28
3.5 Sonar signal inverter 29
3.6 Jason agent reasoning cycle 31

4.1 Localization algorithm run with robot stopped in the center of the environment . 34
4.2 Localization algorithm performance comparison 35
4.3 CPU usage running Jason 36
4.4 Heap memory usage running Jason 36

A.1 UAVTalk Library class diagram 50

LIST OF FIGURES

List of Tables

2.1 Quadcopter punctual projects advantages and disadvatages 16
2.2 Quadcopter frameworks comparison 18

3.1 List of parts and cost used in building the UAV 25
3.2 UAVTalk message composition 30

4.1 IR sensors test results 33

5.1 Summary SWOT analysis 40

Abbreviations

2D	2-Dimensional
3D	3-Dimensional
AI	Artificial Intelligence
BB	Beaglebone
BDI	Belief-Desire-Intention
DAI	Distributed Artificial Intelligence
FMA	Flying Machine Arena
GCS	Ground control Station
HTN	Hierarchical Task Network
IMU	Inertial Measurement Unit
IR	Infrared
IPC	Inter-Process Communication
JVM	Java Virtual Machine
PID	Proportional-Integral-Derivative
MAS	Multi-agent systems
MaSE	Multi-agent Systems Engineering
ROS	Robot Operating System
SAS	Single-agent systems
SIR	Sequential Importance Resampling
SWOT	Strengths, Weaknesses, Opportunities, and Threats
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

The following chapter introduces general aspects from the developed work aiming to clarify its context, problem statement, and main goals. Moreover, it presents the contribution and document structure so it might ease the work of the reader.

1.1 Research Scope and Motivation

In the recent years advances in robotics supported the proliferation of Unmanned Aerial Vehicle (UAV)-based solutions that span from military to civil application. Some known real-world usage of such technology are applications such as aerial recognition, search-and-rescue, industrial monitoring missions among others. For instance, the Predator and Reaper, two drone built by General Atomics, which were used by the United States Air Force to recognition and combat over several countries[1]. A more pacific application of UAVs is monitoring agriculture as done by the company AGX Tecnologias that developed several configuration of aerial vehicle to map different varieties of plantations [2].

A particular configuration of UAV that became popular during the last years is the one of vertical land/takeoff, known as multirotor vehicles. There are also real-world applications for multirotors and, to cite a few of many other, we highlight the first "arrest" made by a multirotor that happened in the United Kingdom. It was a curious case where the car thief hided in the brushes and police officers were able to arrest him due to an on-board camera in the device[3]. Also, Amazon.com Inc., the world's largest online retailer, announced their Prime Air service which is a new shipment system where a multirotor delivers packages to customers[4]; even though the

[1] http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104470/mq-9-reaper.aspx

[2] http://www.agx.com.br/n2/pages/index.php

[3] http://www.dailymail.co.uk/news/article-1250177/Police-make-arrest-using-unmanned-drone.html

[4] http://www.amazon.com/b?node=8037720011

company states that the service will be only available in some years, it confirms that the interest in multirotors has spread through different domains.

Although the proliferation of applications using multirotor vehicles, they are still constrained by human intervention in activities as piloting the vehicle. Indeed an operator is necessary to remotely control them. This fact in many cases can result in high maintenance and deployment costs particularly speaking in the industrial domain applications. Some applications implement an autonomous flight mode, however the autonomy here is intended as a simple path planning through several given points.

Thus, one of the great challenges is how to provide and embed into a multirotor vehicle real autonomous decision-making capabilities. This is true in many application where the environment is dynamic and short term decisions need to be made to accomplish the designed mission. Another kind of situation where decision-making capabilities are desired is when cooperation among vehicles is necessary to pursue a common goal. To that sense the agent and multi-agent system paradigms are of a great importance and inspiration.

An envisioned scenario of some years of research from now is to imagine having a robot that can pick up valuables and perhaps save people when there is a fire in your house. As far as it seems from our current reality, we believe that it is not far from happening due to the rapidly advance of microelectronics and artificial intelligence. Thus, this work intends to give the first step towards an aerial vehicle equipped with autonomous decision-making and reasoning capabilities that controls and adapts itself to new environmental conditions to be applied in civil scenarios. Examples like the ones given above drive researchers around the world in enhancing this machines with more capabilities and skills.

1.2 Research Problem, Aim and Goals

Even though nowadays the design and implementation of an UAV unit imposes less effort to researchers, the necessity to provide vehicles some degree of autonomy and reasoning is challenging and imposes many issues; such as how much intelligent the UAV needs to be and how to embed high costly cognitive algorithms in quadcopter while responding in a timely fashion to execute maneuvers. Developers need to consider that some applications are more demanding than others and thus the architectural frameworks behind these systems should adapt themselves to fulfill these tasks. To address these design issues, flexibility and modularity are inherent requirements for such architectures. That is, they should effortlessly adapt to changing conditions and each of its component should be transparent to others easing its usage in different multirotors configurations.

From the previous discussion, the problem statement is *the lack of software architectures that properly supply modularity features when implementing high-level reasoning capabilities to multirotors vehicles*. Although the multirotors with autonomous features is a wide research area, this work is limited to map information from lower level controllers and sensors to a more deliberative layer.

Much research on multirotors platforms has been carried out focusing on stability and flight control issues rather than on software foundations to ground cognitive reasoning. Albeit deployments are vast in the literature, this work relies on the belief that the next step to achieve the full potential of UAVs is to embed intelligence in these devices. Consequently, this dissertation aims to specify, implement and test a modular, extensive and flexible software architecture for the design of autonomous UAVs.

To better understand our main goal, we introduce the proposed approach that will be deeply described in following chapters. From a software perspective, the current proposal is an instantiation of a three layer architecture divided into different abstraction level operations: reactive, executive, and deliberative. In the deliberative level we have the strategic decision-making processes where is decided what high-level actions will be taken; the executive layer translates those aforementioned actions to low-level information for the controllers; lastly, the reactive layer makes possible to execute the control action and receive new perception from sensors. On the other hand, from the hardware perspective, we worked with two computing platforms, each dedicated to the different layers of the proposed architecture (that is Openpilot CC3D to reactive layer and BeagleBone computer to executive and deliberative layers), which will be explained throughout this document.

In order to achieve this, the project was divided in several goals as follows:

1. review the technical background for this dissertation as well as the relevant works to identify the gap in the current state of the art;

2. implement the interaction between the reactive and executive layers to decouple the upper layers from the low-level control implemented by the firmware of the Openpilot CC3D board;

3. extend the model to support additional sensors. Different set of sensors are necessary to assure proper operation of the UAV platform in different environments. In the current thesis we consider a minimal configuration of "external" to integrate the existing ones. Namely the set of sensors to further consider other than the embedded 3-axis gyroscope and accelerometer, are four infrared and two sonar devices;

4. assess the sensors integration. A correct reception of coherent sensorial data is of imperative importance for the UAV to estimate and update its current state;

5. adjust and evaluate the low-level control. Based on the different payloads and dimensions of the quadcopter, different values of the controller parameters are necessary. To ensure that the quadcopter is stable enough to flight, the PID control loops require tuning;

6. implement an algorithm that based on the available sensor data localizes the drone in an environment. In order to avoid undesireable drift the robot must be able to tell where he is and where he is going;

7. integrate a reasoning engine and link that same engine with the rest of the robot. To easily develop and test agent behavior method a researcher must be able to run high abstract plans from within the robot.

This dissertation intends to contribute to the body of knowledge of unmanned aerial vehicles by prototyping a low-cost quadcopter framework that uses the proposed architecture in order to modularize the research in this field while decouples each layer from the others. For instance, performance of different decision-making processes may be evaluated regardless the motion control, though we are aware of their influence of these parts in the whole multirotor performance.

The proposed architecture is divided in three layers. The top layer encapsulates everything related to high level planning, it answers what the UAV has to do. The middle one contains a description of the environment and links the planning algorithms with the motion controllers. The bottom one controls the UAV, answering the question how is it done.

1.3 Dissertation Structure

The rest of the document is organized in the following way. In chapter 2 a review of the literature on the topic is done, it starts by looking into background questions inevitably necessary to the rest of the project and follows to related works that have the same or similar goals as this dissertation. In chapter 3 a detailed description of the proposed approach is given. It focus in better defining the problem and explaining the proposed method. Chapter 4 presents the results achieved during the final stage of the project. At last, in chapter 5, it is written an overview of the document, the final solution is analyzed, future paths are highlighted and a look back is taken.

Chapter 2

Literature Review

This chapter contains both the background research and the review of related works. The objective here is to bestow the reader with the needed information to analyze the rest of the document.

2.1 Background

This section introduces the basic knowledge needed before starting the dissertation. Here the used concepts throughout this document are explained in detail. This work involves five main areas which we review in this chapter and they are: autonomous agents, mobile robotics and autonomous vehicles, quadcopter dynamics, localization and state inference, and finally deliberation and planning.

2.1.1 Autonomous Agents

Multi-agent systems (MAS) is the area of Distributed Artificial Intelligence (DAI) that "is concerned with coordinated intelligent behavior among a collection of (possibly pre-existing) autonomous intelligent 'agents:' how they can coordinate their knowledge, goals, skills, and plans jointly to take action or solve (possibly multiple, independent) problems" [Gas87]. This concept rises many issues regarding how agents interact and create societies, how the environment affects the sphere of perceptions and influences of the agent.

However, in some scenarios, issues regarding communication between agents, coordination and modeling of other agents are not present. This happens because there is only one agent in the environment, resulting in what is called single-agent systems (SAS). SAS are a special case of multi-agent systems [SV00] where a single entity receives all perceptions and decides its actions. This assumption does not constraint the existence of another entity in the environment rather it means that the agent does not recognize any other and existing ones are modeled as part of the environment.

Even though there is no unique definition of an agent, the agent-based designs were (and still are being) successfully deployment in different domains. As Tolk stated [TU09], an agreed-upon definition is not mandatory for success. Franklin, aiming to answer the question: "Is it an agent,

or just a program?" [FG97], concluded that it is possible to identify the essential features of an agency but there was still room for improvement in the existent definitions; many other authors try to answer this question [TU09, RNC+95, JSW98]. This dissertation considers the definition presented by Jennings and Wooldridge [JSW98] that defines an agent to be a computer system with the three following properties: *situateness*, *autonomy* and *flexibility*.

- **Situateness:** The agent belongs to an environment, affects and is affected by it. An agent has to be able to sense some information from the environment and to manipulate parts of it such as other agents or passive objects. It may also be manipulated by such agents or external conditions.

- **Autonomy:** The agent controls its own internal state and decision making process. This does not mean that an agent is a closed entity, it should consider other agents or human opinions but, nonetheless, it must be able to act without those.

- **Flexibility:** The agent can change between responsive states, it can be more or less reactive/deliberative based on its needs. Also, it is able to adapt its objectives and actions to new environments. When facing a new situation, the agent has to be able to respond to it in the best way possible; if the previous objectives become irrelevant then the agent should find a new goal and set of actions to complete such goal.

In this dissertation agents are considered only software agents; this is not an attempt to complement the definition but a way to separate the terms in the rest of the document. The entire system (software and hardware) will be referred as *robot* and will be introduced later in 2.1.2 Mobile Robotics.

We shall now look into the agents themselves. There are essentially three kind of agents architectures: reactive, deliberative and hybrid; and we describe them below.

Reactive

The research in reactive agents started in the middle 80's. The pioneers rejected symbolic Artificial Intelligence (AI) approaches that explained intelligence as a group of facts and rules and supported that intelligence is a result of the interaction between agent and environment [Woo08]. This means that intelligent behavior does not need to be explicit and rather may raise from the relation between many simpler components. In 1986, Brooks proposed the subsumption architecture [Bro86, JSW98], which nowadays is one of the most known architecture for reactive agents. Brooks divided a robot in layers as one would expect, but instead of dividing by functionality, the normal approach till then, he divided according to behavior. In a subsumption architecture there are multiple reactive controllers that regulate one another in a hierarchical way; each controller is given a task and it is constantly trying to achieve its goal. The final output of the agent is the congregation of all controllers outputs. In order to define which controller has priority, the upper controllers can modify the internal state of the lower ones enabling or disabling behaviors. However, subsumption (as well as other reactive architectures) has a grave fault to be of use in

deliberative systems [RNC+95]. When trying to create system that answer complex situations the relations between the controllers becomes too confusing for people to understand, therefore it is considered to be a reactive agent.

Deliberative

Among the best-known deliberative architectures is the Belief-Desire-Intention (BDI) proposed by Rao [RG+95] because it is so popular it will discussed here. BDI separates the mental states of an agent into three "mental attitudes" [RG+95]: beliefs, desires and intentions. Beliefs are the ideas an agent has about the world. It does not consider them completely correct or static, the agent knows a belief can change depending on the environment or in the sensors data, that is why beliefs differs from knowledge. Desires represent the current objectives of the agent; the agent will actively try to get to desired states. Again, as with beliefs, the desires set is mutable; some desires may get outdated or impossible and new ones may surface during the lifetime of the agent. Lastly, there are intentions, which represent multiple course of actions available for the agent at a given time, it then commits to one of the intention until the desires or beliefs change rendering that intention not optimal between the options. An usual objection to BDI agents is their inability to plan their actions [DSSP09], this topic has been receiving more attention by the community and it now has some possible solutions. In [SdSP06], an approach using Hierarchical Task Network (HTN) is presented. The problem with this approach is the great dependency on the programmer in plans creation; it simply provides a look-ahead technique to choose among existing plans.

Hybrid

The problem with both of the architectures presented above is that they do not regulate the entire agent. Reactive architectures do not support high-level planning and deliberative architectures do not control the agent by themselves. Addressing this, researchers created a new model of architectures that tried to take the best of both worlds. Hybrid architectures are a composite of subsystems where each has different functions in the entire agent. These components, more generally referred to as layers, are dotted with different level of reasoning. Layers closer to the inputs are more reactive and layers away from inputs are more deliberative.

The Three-layer Architecture [RNC+95, G+98] raised as a manner to solve problems with the subsumption-like architectures. The idea is simple, there are three layers each taking care of a part of the system. The reactive layer has the control over the sensors and actuators, it answers whenever there is a demand for real-time actions or there is no need for deliberation. The executive layer works as a middle man between the other two, it receives plans resulting of the deliberation and transcripts them into actions. Also it interprets the data from the reactive layer into meaningful information offering the higher layer a representation of the world. The deliberative layer reasons in a higher level, keeps the agent focused on goals, plans for solution and other time consuming tasks that require higher degrees of intelligence. This architecture, in all of its implementations, has over the last years given good results and will be used for this dissertation.

Nevertheless, nowadays more architectures are being proposed to solve this problem and below two are presented:

- **CLARAty:[NWB+03]** CLARAty intends to allow researchers to easily deploy a working architecture for robot control without the need to remake it. The architecture separates the system in two layers, a Functional Layer and a Decision Layer, the functional layer encapsulates everything related to hardware from the Decision Layer while the latter decides how to proceed. The Decision Layer reason about resource availability, intended goals and world modeling, while the Functional Layer keeps the system going, it can be seen as an operative system that abstracts low level controls from the rest of the system.

- **SC-Agent:[PPS+08]** SC-Agent is an architecture divided into two layers. It aims to make robot control easily scalable and robust at the same time. The most important part of the architecture is the communication framework that takes into account specific problems result of distributed systems (such as latency and synchronization). The whole architecture is made of agents, when the deliberative level want to provide a new goal to the reactive one, it simply sends a new agent to it. This approach allows a high level of dynamism in changing environments.

2.1.2 Mobile Robotics & Autonomous Vehicles

The definition of robot evolved together with the technology to build it. Nowadays, researchers understand robots as machines capable of carrying out a complex series of actions automatically and undependable of human interference [Dia93]. This concept is directly connected to the definition of an agent mentioned in the last section before. The difference between them, as their main characteristics are mutual to both, is that an agent may or may not be associated with a hardware device where robot always include software and hardware components.

With this definition a wide range of robot can be identified, this dissertation only considers robots that can move according to their desire, also known as *mobile robots*. This definition, however, is still abstract and applicable to many examples; in order to better define our focus, a taxonomy was designed and below the different classification aspects are described.

Environment: a robot is designed according to the place where it is going to operate. For example, an underwater vehicle has to be waterproof, on the other hand an aerial one does not have this constraint. This means the environment has a big impact on the structure and functionality of the robot and should therefore be highly considered when trying to classify it.

Locomotion: other important characteristic is how does the robot move in the world. In order to operate in new places a robot must change to different locations, this demands a locomotion method. In nature we find examples like legs or wings, in vehicles we find wheel and propellers. Robots due to their higher degree of control can use all of the above, creating this way a big range of options in this category.

Configuration: describing the mean of locomotion is not enough, the configuration in which these actuators appear also influences the abilities of the robot. Propeller-driven airships use the

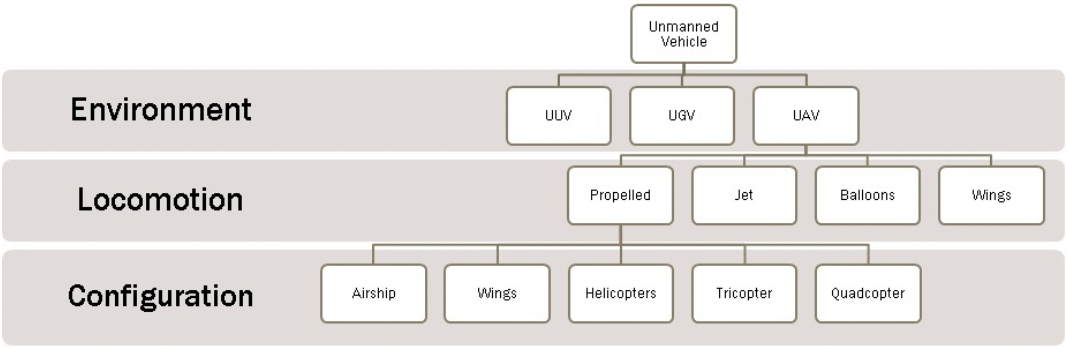


Figure 2.1: Minimum spanning for quadcopter classification

same locomotion mean as a helicopter but with the different configuration different abilities can be achieved and constraints avoided.

Of course one could argue that this taxonomy does not avoid all the ambiguities, a aircraft is a conjugation of balloon and propellers and therefore can fit into two different categories. This problem was understood but discarded as is not important for the research conducted through this dissertation.

Based on the taxonomy, a concise definition of quadcopters can be done, as in Figure 2.1. It is important to note that only the important nodes where explored and for simplicity purposes only some of the nodes are present. A quadcopter is then an UAV which uses four propellers for thrust and has them configured in either a cross or plus format. The quadcopter robot can take off and land vertically which in the UAVs world is a big advantage as it lowers the requirements for a landing platform. Also, it allows the robot to hover in place with considerable stability.

Other topic that was studied about mobile robotics was the proportional-integral-derivative controller (PID controller)[Ast95]. A PID controller is a control algorithm that calculates the inputs based on the feedback from the previous action in addiction to the desired output. The algorithm needs a different quoficient (gain) for each of the three representations of the error. P denotes the proporctional gain which relates to the present error, I relates to the past erros and is called integral gain and lastly D which is the derivative gain and previews future error considering the current evolution of the system.

2.1.3 Quadcopter Model

The general model of a quadcopter has been fully detailed in other works, for example [CDL04, TM06], and will not be discussed in detail here. In Figure 2.2, the coordinate system for the

quadcopter can be seen. The world frame W is a 3-Dimensional (3D) space composed of x_W, y_W and z_W, with the z_W axis pointing away from the center of the planet. The body frame B is composed of x_B, y_B and z_B with z_B pointing in the same direction of z_W in perfect hover state, x_B and y_B going in the direction of motor 1 to 3 and 2 to 4, respectively. This dissertation uses ϕ, θ and ψ for the roll, pitch and yaw angles which give a rotation matrix R:

$$R = \begin{bmatrix} cos\psi cos\theta - sin\phi sin\psi sin\theta & -cos\phi sin\psi & cos\psi sin\theta + cos\theta sin\phi sin\psi \\ cos\theta sin\psi + cos\psi sin\phi sin\theta & cos\phi cos\psi & sin\psi sin\theta - cos\psi cos\theta sin\phi \\ -cos\phi sin\theta & sin\phi & cos\phi cos\theta \end{bmatrix}. \tag{2.1}$$

The position of the quadcopter in the W frame is given by the function r, as can been seen in Figure 2.2, and the forces acting on it are the gravity in the $-z_W$ direction and F, the cumulative force of the rotors. The acceleration of the quadcopter in reference to the world can be expressed as

$$r'' = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} m^{-1}. \tag{2.2}$$

And the angular velocity

$$w = \begin{bmatrix} cos\theta & 0 & -cos\phi sin\theta \\ 0 & 1 & sin\phi \\ sin\theta & 0 & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \phi' \\ \theta' \\ \psi' \end{bmatrix} \tag{2.3}$$

Besides forces we also have moments produced by the blade rotation on the rotors but these can be ignored as they cancel out each other. M_1 and M_3 produce a counterclockwise moment and M_2 and M_4 a clockwise one, at hover state or vertical translations they cancel out leaving the

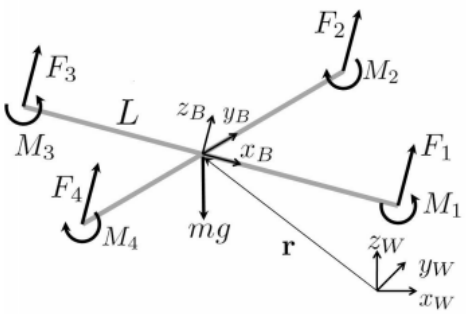


Figure 2.2: Coordinate systems and forces/moments acting on the quadcopter [MMK12]

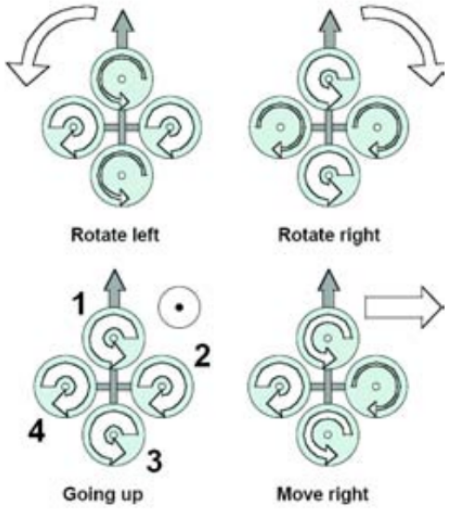


Figure 2.3: Correlation between rotor blades speed and quadcopter motion [Bou07]

quadcopter with no angular velocity change. However, a quadcopter might need to rotate and for this purpose the angular accelerations can be calculated with

$$I \begin{bmatrix} p' \\ q' \\ r' \end{bmatrix} = \begin{bmatrix} l(f_2 - f_4) \\ l(f_1 - f_3) \\ k(f_1 - f_2 + f_3 - f_4) \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.4}$$

where all the rotor are equal and at the same l distance from the quadcopter center of mass, I is the inertia matrix of the vehicle and k is a constant determined from the rotors.

Using this model, controllers may designed to make the quadcopter change its position and attitude. As can be seen on Figure 2.3 by increasing or decreasing the rotation of the blades the quadcopter can yaw and and translate freely in 3D space, bestowing it with six degrees of freedom, a so much appreciated feature in UAVs.

2.1.4 Localization and State

In order to perform autonomous intelligent navigation, a robot needs to know its global localization, i.e. a robot must know its position according to a system wide reference [TFBD01, Thr03]. This information is needed so that the robot can define its state. The robot's state contains all the information about itself and its relation with the environment, for example, its localization, battery level, goals and so forth.

Most of this information, specially localization, is dependent on how the robot models the world. As can be seen see below there are many types of possible descriptions [Jen01]:

- **Topological maps:** a topological map describes the environment based on its utility to the robot, i.e. what in the scope of the robot operations can be performed there. The maps is seen as a graph where: nodes represent places, such as rooms; edges represent links between places like hallways or doors. Moreover, each node contains a description of the place or its abilities. A room may, for example, contain a printer; this would augment the respective node with the ability to give access to a printer. This kind of maps are clearly very useful for high level deliberation. It is easy to plan for a goal on this description. However when computing the trajectory from A to B (where A and B refer to spacial coordinates) these maps are not enough, as they do not contain geometric information about the environment.

- **Feature maps:** a feature map is a list of features extracted from the environment and with known positions. This mapping technique offers a good geometric description of the environment as, by observing a feature and computing its relative position, it is possible to calculate the global position of the robot. The shortcoming here are three: the number of unique features may be too small, i.e. the environment may be too simplistic; the difference from one feature to another may not be enough for the sensors to understand; the feature by themselves do not give any more information about the environment like the topological maps might give.

- **Grid maps:** a grid map divides the map into subspaces, each position is either occupied or free and the robot calculates his position by evaluating the grid around. This approach has the advantage of reducing the path-planning problem to a search and trajectory smoothing algorithm, however the updating process to the entire grid is very computational intense and the grid usually fill a lot of memory.

Moreover, two or more techniques might be coupled together to represent the environment and indeed that is the strategy for the most autonomous vehicles using topological maps for high level reasoning and a different approach for geometric localization [LCK03]. Having a map of the environment there is still the need for localization in it,. In order to solve this issue two algorithms were studied. The methods below are the most common approaches to localization and both can identify the robot localization from an unknown starting position.

Kalman filter [AM12, K$^+$60] is an algorithm that uses a stream of discrete measurements to statistically produce optimal estimates of the real state. Although the algorithm always finds the optimal solution, it has some assumptions that limit its utility. First, the algorithm considers that the sensors have only white noise, i.e. the error can be modeled as a Gaussian distribution. Second, it considers that world is linear which for most application is not true.

The Kalman filter considers that the current state evolves from the previous one according to

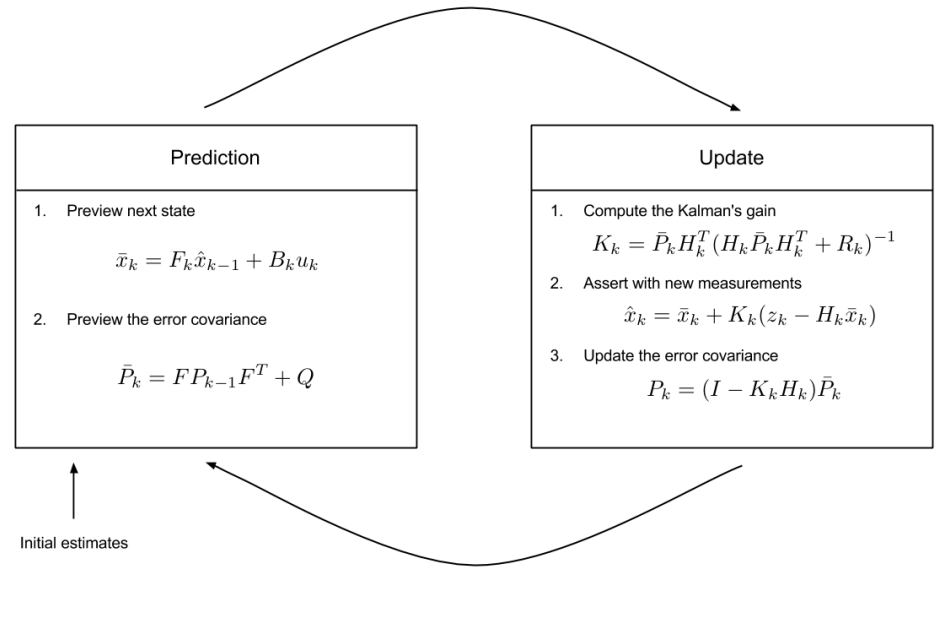


Figure 2.4: Kalman filter overview
[WB95]

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{2.5}$$

$$z_k = H_k x_k + v_k \tag{2.6}$$

$$w_k \sim N(0, Q_k) \tag{2.7}$$

$$v_k \sim N(0, R_k) \tag{2.8}$$

where k means a time instant, x the state at time k, F a matrix that relates the previous state to the current one, B the matrix that maps the inputs to the state of the system, H a matrix that correlates the present state with the current measurements, w and v represent the process and measurement noise respectively and Q and R their covariance matrices.

The algorithm is divided in two phases as can be seen in Figure 2.4. The first, called prediction, previews the *a priori* next state estimate $\bar{x}_k$ based on the current *a posteriori* estimate $\hat{x}_{k-1}$. The second, called update, happens at time k and fuses the new measurement with the estimate $\bar{x}_k$ to obtain an *a posteriori* estimate $\hat{x}_k$.

$$\bar{x}_k = F_k\hat{x}_{k-1} + B_k u_k \tag{2.9}$$

$$\bar{P}_k = FP_{k-1}F^T + Q \tag{2.10}$$

Equations 2.9 and 2.10 describe the prediction process.

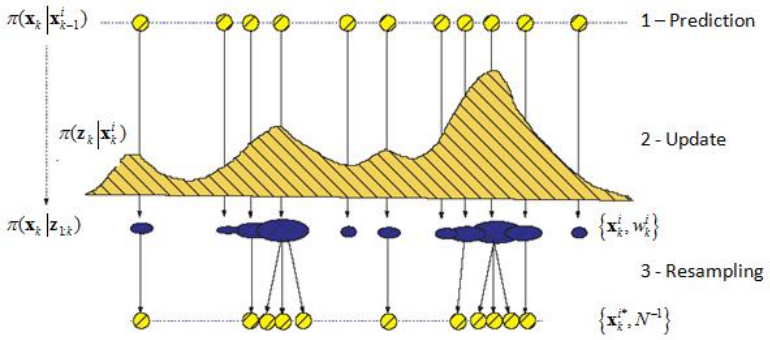


Figure 2.5: Representation of the Sampling Importance Resampling (SIR) algorithm [ODC08]

$$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \tag{2.11}$$

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H_k \bar{x}_k) \tag{2.12}$$

$$P_k = (I - K_k H_k)\bar{P}_k \tag{2.13}$$

Equations 2.11, 2.12 and 2.13 describe the update process. P is the error covariance matrix, $\hat{P}$ its *a priori* estimation and K is a matrix called Kalman's gain.

The Kalman filter is only applicable when the system is linear, but there are many scenarios when this assumption cannot be done. To overcome this limitation other approaches have been proposed [AM12, Thr03]. One of which is the particle filter [Jen01, ODC08].

A particle filter is an estimation technique that uses weighted particles to approach *a posteriori* distribution. A particle is a sample from the *a priori* distribution that can be weight according to its importance in the *a posteriori* distribution. Using this method non-linear update functions can be modeled. Particle filter virtually model any possible distribution, the problem is that it no longer guarantees optimal solutions and may even fail to give one if the number of particles is too small or badly sampled.

There are many algorithms for particle filters, below the Sequential Importance Resampling (SIR) is explained. This algorithm first proposed by [GSS93] uses samples from an importance distribution instead of the actual distribution. The idea is that as the number of particles increases the importance distribution $\pi(x_k|x_{k-1}^i)$ approaches the empirical distribution.

The algorithm can be seen in Figure 2.5 and it is summarized in the following steps:

1. Draw N particles from $\pi(x_k|x_{k-1}^i)$ and calculate the weight for each $w_k^i = \pi(z_k|x_k^i)$
2. Calculate the total weight $T_w = \sum_{i=1}^{N} w_k^i$ and then normalize the weight of each particle $\forall x \in [1,N], w_k^i = \frac{w_k^i}{T_w}$
3. Resample by drawing particles from the current set favoring the particles with higher weights:

3.1. Calculate the cumulative sum of weights $\forall i \in [1,N], c_i = c_{i-1} + w_k^i$, with $c_0 = 0$

3.2. Set $i = 1$ and draw u_1 from a uniform distribution $U[0,N^{-1}]$

3.3. $\forall j \in [1,N]$ make:

3.3.1. $u_j = u_1 + N^{-1}(j-1)$

3.3.2. While $u_j > c_i$ do $i = i+1$, $x_k^j = x_k^i$ and $w_k^j = N^{-1}$

In step one, N particles are drawn from an uniform distribution of all the particles in the space. Each particle represents a possible state or, when talking about localization, a pose. Each particle is then weighted according to the sensors inputs. The weight of a particle is proportional to the probability of getting said sensor input on the state represented by that particle. This generates a list of weighted particles. However the weight of a particle is not as important as the relation between weights; therefore, step number two is to normalize the particles. The list contains now a set of particles and their relative weight. The particle with the biggest relative weight represents the pose where it is most probable to find the robot.

Step three is the preparation of the next population of particles; if the algorithm drew again from the uniform distribution, it would loose the information about the past and each iteration would never return better results. To counter this it is important to draw the particles considering the current weighted set. By calculating the cumulative sum of weights the algorithm creates something like a wheel of fortune, where the size of each cell is as big as the weight of each particle leading to a bigger probability of selecting particles that better represent the current state.

With a representation of the map and its position relative to that map, a robot is able to compute the trajectories between itself and an objective point in space as well as is able to understand its movement state. This allows the robot to freely manoeuvre in its environment without crashing into obstacles.

2.2 Related Works

This section describes important research projects that intend to achieve similar goals then this dissertation's. It does not intent to be an exhaustive survey on the matter, but tries to highlight the main trends and storyline of autonomous vehicles and quadcopters research. The first subsection describes projects linked with quadcopters research. The second describes projects that gather agents and mobile vehicles together in one system.

2.2.1 Quadcopters research and development

This section shows quadcopters projects that can be related to the one discussed in this dissertation, research approaches that have been taken and lastly a small insight into use of quadcopters by individuals. Table 2.1 presents a list of the related quadcopter projects as well as an analysis of their advantages and disadvantages.

Table 2.1: Quadcopter punctual projects advantages and disadvatages

Name/Author	Entity	Advantages	Disadvantages
OS4 [Bou07]	EPFL	• Many controller options; • highly detailed dynamics model.	• No autonomous flight; • expensive.
[Bur10]	DTIC	• Low cost; • testbed for research.	• No operating system on-board; • no autonomous flight.
[GSGA09]	MIT	• Virtual simulation; • stabilization recovery.	• No autonomous flight; • does not support six degrees of freedom.
Microraptor [RYAS09]	Oakland University	• 5th place on SUAS[1]; • video-based localization; • some degree of autonomy.	• Uses redudant sensors; • autonomy dependent on GPS signal.
RAVE [BBP06]	ESTIA LIPSI	• Tracking algorithm.	• Gas-powered; • no real autonomy.
[HHWT09]	Stanford University	• Ability to perform aggressive maneuvers.	• No autonomous flight.

In the following paragraphs there is a description of the history and methodology used in two long running projects. Both of them already had many researchers achievements and have contributed enormously to the development of these UAVs.

Flying Machine Arena (FMA) [LSHD11] is part of Swiss Federal Institute of Technology in Zürich. Although some project that are now the base for the FMA date from the 1990 [Are13], the real start was in 2004 with a PhD Thesis from Eryk Nice [Nic04]. The author states that there were no previous good solutions for a machine that could hover in place autonomously and his success was the first notorious result of the yet to be FMA. Concurrently, there was another project going on, which intended to allow the robot to localize themselves indoor by fusing the information from the robot on-board sensors and a vision system installed on the room; this project was never implemented but was the idea that led to the FMA. Many years passed and the FMA

was still researching but at a slower rate, until 2007 when D'Andrea returned to the academic world and pushed FMA forward. Starting from 2009, we can identify many important research results in quadcopters control and autonomy. In [PD09], authors describe an Iterative Learning Control (ILC) that allow a quadcopter to perform aggressive manoeuvres without the need to precisely model the entire environment as such would be too costly. This algorithm was light enough to run online on the robot and was tested in the FMA quadcopters. At the time, during tests, researchers found that the vision-based localization system used in the FMA gave some misalignments [DD09] due to impacts or hand manipulating of the quadcopters. With this in mind they developed a system that could recalibrate the system automatically even when there were multiple robots. Their research path led to problems like synchronization of quadcopters, in [SALD10] a method to synchronize robots to music is explained. The assumption there was that if it is possible to coordinate a quadcopter to an external signal it is possible to coordinate multiple machines and achieve a harmonious multi quadcopter system.

Two years later FMA published an article [RMHD12] where they report the successful coordination of a group of quadcopter in catching and throwing a ball. Although references that relate the project to a multi-agent system can not be found in the paper, the level of coordination is advanced. In the same year two other papers related to this dissertation were published [MD12, ASD12], the first describes a controller to safeguard mechanical failures in the quadcopters or the vision system. As this technology is getting more public such measures are needed to prevent disastrous events. The second reports a method to generated trajectories for quadcopters fleets. Although some assumption were taken which released the need for negotiation between the robots, the results are still important as such technique is needed in order to allow for fleet manoeuvres.

Another related work that has been getting much attention in the last years is the GRASP Lab at the University of Pennsylvania [MMLK10]. They also have a broad range of subjects but focus their applications to quadcopters. In 2010, they published a paper [MSK10] describing a method to control quadcopters landing on difficult situation, like upside down platforms. During their research they found that this method also allows the quadcopters to pick up objects with the use of a claw. But what is more important is that although the global localization is given by a vision system, the quadcopters have to identify the landing surfaces on their own and thus starting a path to autonomy. Not one year after Lindsey et al. showed in [LMK11] a high degree of autonomy and coordination as quadcopter had to cooperate in order to build a 3D structure. This is an important work because it touches many problems on multi-agent systems, such as the robots had to plan not only their actions but also those of their peers, otherwise one could preclude the others work. By 2012 in [Mel12, MK11], Daniel Mellinger reported methods to both single and multiple quadcopter systems, that allowed the quadcopters to generate and follow aggressive trajectories. Lastly in [KMK12] a fleet of small quadcopters flies in formation with less than a body length of separation. They overcome obstacles without ever crashing into each other or the environment.

Yet not only in research have these robots been used, there are multiple low-cost open-source platforms that can be purchased by individuals [LPLK12]. Such projects include some like: Open-

Table 2.2: Quadcopter frameworks comparison

	OpenPilot	AeroQuad	ArduCopter	Pixhawk
GPS-based waypoint navigation	$\sim$	$\times$	$\surd$	$\times$
Altitude hold	$\sim$	$\surd$	$\surd$	$\surd$
Hardware in the loop simulation	$\surd$	$\times$	$\surd$	$\times$
Support of other multirotor airframes	$\surd$	$\surd$	$\surd$	$\surd$
Support of computer vision	$\times$	$\times$	$\times$	$\surd$
GCS provided	$\surd$	$\surd$	$\surd$	$\surd$
Camera stabilization	$\surd$	$\surd$	$\surd$	$\times$
Used by			[LLK+12, LSCU12]	[MTH+12, MTFP11]

$\surd$ - Supported $\sim$ - Needs addon $\times$ - Not supported

Pilot[2], AeroQuad[3], ArduCopter[4], Pixhawk[5] and some even created their own Raspberry Pi Quadcopter[6]. The table 2.2 provides a comparison between the projects mentioned above. It can be seen that the diversity of choice when picking a quadcopter framework is vast and each of the frameworks has its own focus of action, some are general like the OpenPilot and some directed to a specific goal like the AeroQuad. It should also be noted that some of the projects are already being used in scientific research. Actually the Pixhawk was developed with that objective in mind.

2.2.2 Agent design applied to autonomous vehicles

The concept of agent and robot have fundamental similarities, both have sensors and actuators, both have environments, both have goals, both have some sort of plans, and so forth. This makes the application of agent designs methods in robots an interesting approach. This dissertation follows this perspective and here it reviews some other project that did the same.

In [HDL92] two method to drive a car on a dynamic environment are explored. The first is based on potential fields and is not closely related to this dissertation; the second however is based on a multi-agent approach. In this multi-agent approach, authors try to solve the problem of planning the car motion by designing the car and other controlled vehicles as agents and non-controlled vehicles as environment entities. The car knows the other agents internal state and assumes that they cannot lie with these assumptions it can preview their decisions. As for the non-controlled vehicles, the method previews their motion by extrapolating the current motion to the future with a basic model of motion.

[2] http://www.openpilot.org/
[3] http://aeroquad.com/
[4] https://code.google.com/p/arducopter/
[5] https://pixhawk.ethz.ch/
[6] http://www.botched.co.uk/picopters-maiden-flight/

In 1998, Mizoguchi et Al. [MNOH99] described the application of MAS in an office to create a smart office; their goal was to create a group of agents that could perform delivery of a printing job on demand. They identified six types of agents: delivery task agents, responsible for accepting the print job from the user; sensor monitoring agents, responsible for processing sensor data; mobile robot, responsible for carrying the paper from the printer to the final user; handling robot agent, responsible for picking up papers from the printer and placing them on the mobile robot tray; camera robot agent, responsible for the navigation of the mobile robot; and lastly the printer agent that control the printer. This problem touched some of the more important challenges in MAS like fault tolerance, negotiation and coordination. This paper is related to the present dissertation because it can be seen that the application of agent methodologies increased the overall quality of the system on a measurable way.

The work performed in [DML02] assumes that there are satisfactory solution for the low-level control of robotics therefore they center the research in high-level deliberation. The authors also test the usage of Multi-agent Systems Engineering (MaSE) methodology for the project from which they conclude being a good approach to design a system in a top-down manner. The design is applied to a heterogeneous rescue team of agents where some agents could only perform some tasks and they had to organize themselves.

Agentfly [ŠPV+08] is a MAS simulator for UAVs supporting the free flight concept. Each UAV has an agent and at start each agent has a planned path for the flight. When designing the paths, collision avoidance is not considered and the problem is solved online. The agents are able to communicate effortlessly in this work but in the future works section a hint about possible research on restricted communication collision avoidance is given. The papers clearly shows good results in terms of performance and convergence of the methods used, however by the time it was written there was still no work in implementing those algorithms in real life UAVs.

In [TKH09], the development of a method for search and localization of a team of UAVs is presented. Although a reference to agents cannot be found, one could argue that the proposed architecture goes in the same direction as agent technology. The article highlights problems like cooperation and planning with uncertainties, which are two problems discussed in agents environment. The method was implemented on fixed-wing airplanes with a bottom facing camera that communicate over wifi. Also, each is equipped with GPS, a pilot-static system and cameras; filtering algorithms are used to join the multiple sensors readings.

The IntellWheels [BPRM11] is a wheelchair with a higher level of cognition that offers the end-user, people with disabilities, a more independent life. The project was developed using multi-agents design methods, the functionalities are divided by different agents taking advantage of MAS features like modularity. The control module is similar to the one proposed in this dissertation. It is divided in three layers, the bottom one related to control of the actuators and the higher responsible for the high-level objective planning.

From this set of related works one can identify that some are using the agents perspective in the robot world and some, although not calling it agent, are using the same approach as well. Furthermore projects that use agent system design techniques as an architecture for the internals

of the robot and not only for the robots themselves can be identified. Many works take this way advantages of the feature of multi-agents systems like modularity and task distribution.

2.3 Summary

In this chapter a review of the background concepts needed for the dissertation was done. There was a needed to collect a big number of techniques because the project is by definition big. The research presented here ranged from agents to dynamic models to localization algorithms.

The research answered some of the fundamental questions raised when forecasting the project and created a set of tools for the development stage. All the solutions found have been tested and validated on real world applications which better supports the options taken in the next chapter.

Additionally, this chapter looks into the related works on the area, there is an ever growing number of researchers turning to UAVs in general and quadcopters in specific. Some worry themselves with the low level control of the robots others with applications of these machines, not so many are going for autonomous vehicles and the ones that do, are still in early stages of development. Clearly, there is a desire to place these machines at work in groups of more than one and also a strong desire to make them see and understand the world.

The main conclusion is that the most part of the research is in the last few years and it started by designing the UAVs themselves. Now that this problem has stable solutions, researches are starting to look into the possibilities these machines create in the civil world.

Chapter 3

Methodological Approach

This chapter formalizes the problem statement and describes the method proposed to solve it. After presenting the methodological approach, we draw some conclusions about the developed work.

3.1 Problem Formalization

This work is an answer to the problem of designing a modular and flexible architecture for an autonomous Unmanned Aerial Vehicles. Not only the relations between controllers has to be taken into account, but the entire synchronization of the system from the most deliberative parts to the reactive ones. In order to achieve these features, the following questions have to be made:

1. **How does the robot know its location according to the internal representation of the environment?**
   As an intelligent agent, the robot must keep an internal representation of the environment. This raises the problem of relating that representation to the actual robot position in the real world. An intelligent being keeps information from the past, but this information should not get distorted throughout the time.

2. **How to divide the architecture in a way that is understandable, flexible, and modular?**
   The level of partitioning has to be well thought when designing an architecture. If the system is too much divided, it might have overhead problems, on the other hand, if it has small number of divisions, it might not be flexible enough.

3. **How to successfully give an UAV reasoning capabilities?**
   The problem of designing an intelligent entity has been a problem in the AI field for many years and it had been proposed various method to such purpose. Our concern here is to apply one of these methods that can be used in low-cost UAVs context. These machines do

not support the same computational potential as a normal standalone applications; then, we pursue a different approach to address this issue.

3.1.1 Assumptions

Although the main goal of the research intends to be used in different scenarios, this project was grounded upon some assumptions to better bound the project. This imposes well defined constraints and clarify the chosen paths followed by our methodology. The list of assumptions can be separated in different areas:

- **Environment:** these assumptions refer to the place where the quadcopter must fly without losing balance and control. Although the proposed approach can be used in environments that breach these statements, for now we will only consider a minimal set of examples.

- **Robot:** in this topic, the constraints are stated for the quadcopter regarding both maneuverability and reasoning capabilities.

- **Communication:** the communication performance between the quadcopter and the ground station is not within the research scope of this dissertation and therefore we assume several constraints in this subject;

It must be highlighted that, when choosing our assumptions, we acknowledge the need to guarantee the relevance of the proposed solution. With this in mind, we choose the following assumptions:

Environment

The environment is the place on space that envelopes the robot. Humans live in very dynamic environments where they interact with other humans and with the environment itself; however, in special environments and within limited time frames an environment can be considered static. Static environments are those where nothing changes without the interaction of an agent. This does not mean that these environments are simple; imagine yourself lost in a maze alone, the environment will not change for some time but you will be faced with a hard task nonetheless. This is the kind of environment considered when designing the architecture for this dissertation. It does not consider changes in the environment besides those introduced by the robot itself.

Another important consideration about the environment is that the robot has a detailed description of it; using the same example above, it would be like having a map of the maze. Although this simplifies the problem it is still a hard task to leave a maze in these conditions if you have no idea where you started or where you currently are.

As this dissertation tries to improve the cognitive side of the robot rather than the control algorithms, it is assumed an indoor environment; which means the UAV is safe from most of the atmospheric interferences.

Robot

We assume that the robot is on-line at all times during a mission and that the energy level during the flight does not influence the controllability. Also, we consider that the robot has no faulty components, although there is always noise in the sensors; a possible failure is when some sensor stops working in the middle of a flight. These assumptions are important because we will not prepare the system to support fault-tolerance features.

Some considerations about the work of others must also be made. In the next section we will better introduce the Openpilot CC3D and the TauLabs firmware but they must be mentioned here; the referenced technologies compose the reactive layer of our system and should enable the quadcopter to be controllable and stable.

Communication

As it will be seen in the following section 3.2, the overall project has high dependence on the communication between all layers; however, as it is not the focus of the this dissertation to study communication issues, some assumptions were taken into account. It is considered that the communication is constant and faultless, which means that all parts of system are connected at all time. Also, it is considered that no load of data exceeds the network limit. The biggest threat concerning this matter is the relay of motion trajectories to the reactive layer, but this are punctual transactions and as consistency is guaranteed by the above assumption; it can parted in smaller pieces and sent this way.

3.1.2 Functional Requirements

Now that we defined constraints of our system, it is time to describe what requirements we expect to fulfill. First and foremost, the system should be autonomous, the UAV should function without human interaction besides the initial "start" command. This means that after connecting the battery all the parts of the system should initialize themselves and become responsive. This also means that the system must be able to run reasoning algorithms at real-time. Although the aforementioned requirements specify that the UAV must be stable, we also have to consider the reasoning time so the robot does not stop its functioning due to possible delays. Another important requirement is the modular integration of all parts of the system. When a user wants to change something on the system it should be simple and to ensure this the architecture should be decomposed in atomic and self-contained components.

Regarding the UAV, it must be stable allowing the reasoning layer or an user to lock it in a given position; this raises several requirements. As the quadcopter is by definition unstable even with a good stabilization algorithm and inertial measurement unit (IMU), it is impossible to completely eliminate all drifts. It is also impossible to measure the drift using only the sensor contained in the IMU (gyroscopes and accelerometer). This feature needs sensors that can calculate the position of the UAV in relation to some outside reference. The data from these sensors needs then to be compared to some internal representation of the environment. This task is commonly known as

localization; therefore, there is need to integrate environment sensing sensors, design maps for the runtime environments and perform localization on those maps.

3.2 System Overview

In this section we detail the proposed architecture and its implementation. For a better understanding, it should be noted that conceptual architecture refers to the overall description of the system and the physical architecture the actual implementation.

As stated in the introduction, we need a conceptual architecture that renders our system both modular and flexible. The architecture should allow researchers to change only certain parts of the system and still get a working deployment. This has a dual advantage because not only allows rapid testing of new techniques but also allows us to incorporate other contributions in our work.

We designed the architecture present in Figure 3.1 following the three-layers architecture design paradigm [RNC+95], in order to ensure a modular architecture by default. A layer is defined as repository of software and hardware that serves a very well defined function; here we have:

- Reactive: this layer holds every component that needs fast-computing cycles, most of sensors, and all actuators; for example, the designer should place rotors, IMUs, stabilization algorithms, and control algorithms in this layer.
- Executive: this layer encompass fundamental components to cross-map control commands and agent's actions. Localization algorithms, navigation algorithms, additional sensors not used in the low-level controllers.
- Deliberative: this layer contains the cognitive engine; it receives processed information from the other two layers and output decisions and commands for the next steps.

In our approach each layer is designed as if it was a black box for the other layers. This way if we change the algorithm for localization in the Executive layer, for example, both the Reactive and Deliberative would still function exactly the same. Below we explain each layer and their physical implementation.

3.2.1 Reactive Layer

The bottom layer of the system contains all parts needed to control of the UAV and it has to be adapted based on the UAV configuration. Here we use a quadcopter design with the motors configured as an X.

Instead of implementing the layer from scratch, we opted to use the OpenPilot CC3D platform running TauLabs firmware. The OpenPilot CC3D is a low-cost all-in-one stabilization board, which means that the circuit board already contains all the needed sensors for stable flight; keeping in mind that the sensors needed for stable flight are not enough for autonomous flight. For the second, we need to be able to perceive the outside world, not only the quadcopter position.

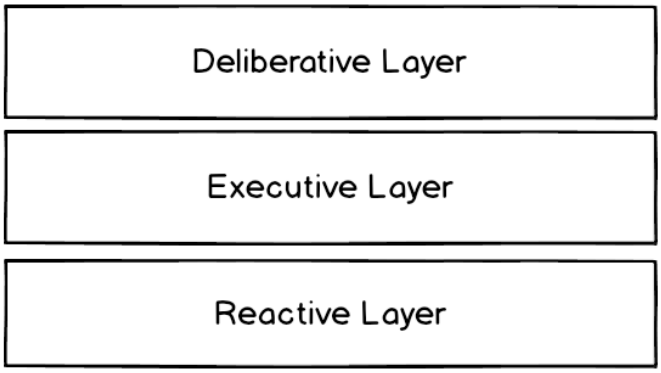


Figure 3.1: Proposed Approach Architecture

Although the OpenPilot Organization develops a matching firmware for their board, we choose to switch to the TauLabs firmware. TauLabs is a open-source software development community that forked from the OpenPilot project in late 2012. We decided to go with their firmware because it targeted teaching and research rather than hobbyist and commercial uses; also it is an active community of developers that gave full support to this project.

Besides the control board and firmware, this layer also contains the sensors and actuators of the UAV as well as other structural components. The table 3.1 documents the used parts and their approximate cost.

The last relevant aspect of this layer is the stabilization algorithm. Although it was not developed by us, there is a need to refine the parameters to match our quadcopter and operation environment. The stabilization algorithm is made of a PID controller inside another PID controller. The output from the first is then the input for the second, as can be seen in figure 3.2. The outer loop assert the attitude of the quadcopter and the inner loop its rate. A quadcopter attitude is its momentary rotation in the α, θ and ω directions, check figure 2.2. Rate is the constant rotation on the same angular directions. The difference between the two is that while attitude is used for holding the quadcopter in position and just moving it from time to time, whereas the rate is used to directly control the position of the quadcopter. Attitude loop is irrelevant when on autonomous flight and if the computational burden is small enough, because the values will be updated very

Table 3.1: List of parts and cost used in building the UAV

Quantity	Part	Name	Cost
1	Frame	Butterfly X250 330mm Shaft Mini QuadCopter Frame	38.00
4	Rotors	Turnigy 1811 Brushless Motor 2000kv	7.51
4	ESCs	Turnigy Plush 6A /.8bec/6g Speed Controller	6.18
4	Propellers	GWS 5x3	1.29
1	Battery	Turnigy 800mAh 3S 20C Lipo Pack	6.09
1	IMU and microcontroller	OpenPilot CC3D	59.90
		Total:	118.97

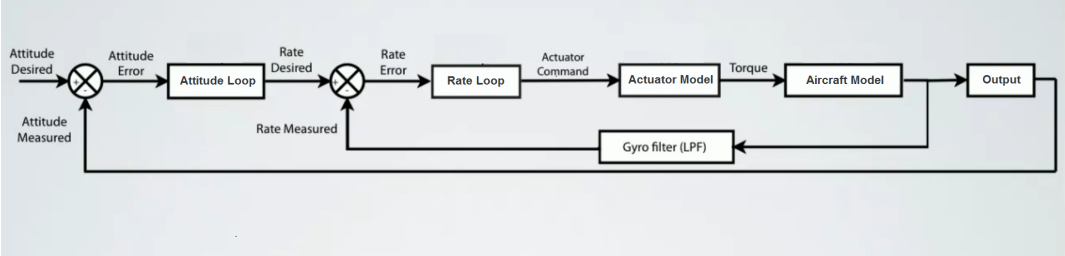


Figure 3.2: UAV stabilization PID

fast an UAV will never be free to return to a levelled position. The reason we decide not to ignore the outer loop is, as was stated above, each layer must be independent of each other, so the UAV might possibly be piloted by humans or computers.

The algorithm has actually three of these chains, one for each direction and each can receive different inputs and gains. However, if the UAV is, like in our implementation, symmetrical it makes sense that the pitch and roll gains are the same. The bounds for the inputs and the neutral value can be changed as fit, but again, if we consider autonomous flight, this make small difference.

3.2.2 Executive Layer

The middle layer holds all software and hardware that works as support for other tasks. In order to get a robot to do what you want, most of the time, it needs to have some information about the environment or maybe it needs to perform computational tasks that are not directly understood as cognitive tasks; everything that qualifies as this goes into the Executive Layer.

The core of the layer is the Robot Operating System also known as ROS; this software system will run on a BeagleBone[1] (BB), which is a credit-card size computer with high performance and a lot of connectivity. ROS is a collection of tools, libraries and conventions that helps researchers and developers develop robotic systems. Although the framework already support most of the tasks relevant to a mobile robot, the algorithms are not particularly indicated to UAVs where the computational resources are so limited, the sensing so minimal, and the response has to be quick.

These extremes are continuously pushing forward the development of new tools and techniques. In our implementation, we go for a minimal sensing approach which aims to reduce the final cost of the UAV, however it also increases the difficulty of success in most tasks. One good example, and a problem we had, is the localization. We started by trying the localization package from [HWB10] but it was not straightforward to apply and the result seemed questionable. Then, we choose to use a sequential importance resampling (SIR) particle filter. The filter is fed the readings from four GP2Y0A02YK infrared (IR) proximity sensor placed in each of the horizontal directions of the quadcopter between the propellers arms; and the control module calculates

[1] http://beagleboard.org/bone

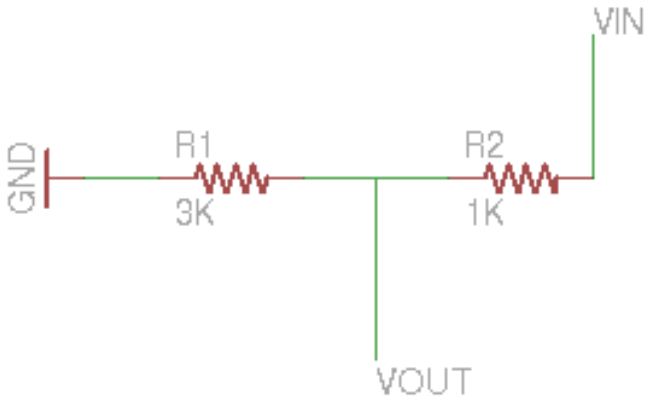


Figure 3.3: Infrared voltage divider

the values of the movements actions. The SIR outputs the most plausible position and that same position is then taken into account when computing the next action for the UAV.

$$D = \frac{A + Bx}{1 + Cx + Ex^2} \tag{3.1}$$

The GP2Y0A02YK IR sensors cost around 10.92 Euros each and can measure distances between 20cm and 140cm. The sensors are very simple and have no embedded ADC so they need to be connected to the analogue pins on the BB. A ROS module developed by us then converts the read voltage in distance with the equation 3.1. The values were computed by performing a regression to a rational function with values taken from the curve in figure 3 in the spreadsheet of the IRs I.1. The output ranges from 0.4V and 2.6V but the BB analogue pins can only take 1.8V at maximum so we need to pass the signal by a voltage divider. With this in mind, the circuit in figure 3.3 was designed. The formula used to calculate the output voltage of a voltage divider is as follows:

$$V_{out} = \frac{R_1 + R_2 + R_3}{R_1 + R_2 + R_3 + R_4} . V_{in} \tag{3.2}$$

This means that to maximize the output range of 0V to 1.8V we should bring the maximum of 2.6V to about 1.8V, therefore:

$$1.8 = \frac{R_1 + R_2 + R_3}{R_1 + R_2 + R_3 + R_4} . 2.6 \tag{3.3}$$

We now arbitrate that $R_1 + R_2 + R_3 = 3k\Omega$ based on the available resources and came to the conclusion that $R_4 \approx 1.2k\Omega$. This only makes the max voltage ≤ 1.85714V but that is almost no lost in performance and still protects the BB.

The values taken from the IR sensors then go into our localization module. This module also receives as parameter a map of the environment. The map as well as the localization are in 2-dimensions. As we stated in the assumptions section 3.1.1, the UAV only operates in known environment for which a map is designed. The algorithm starts by selecting random particles from

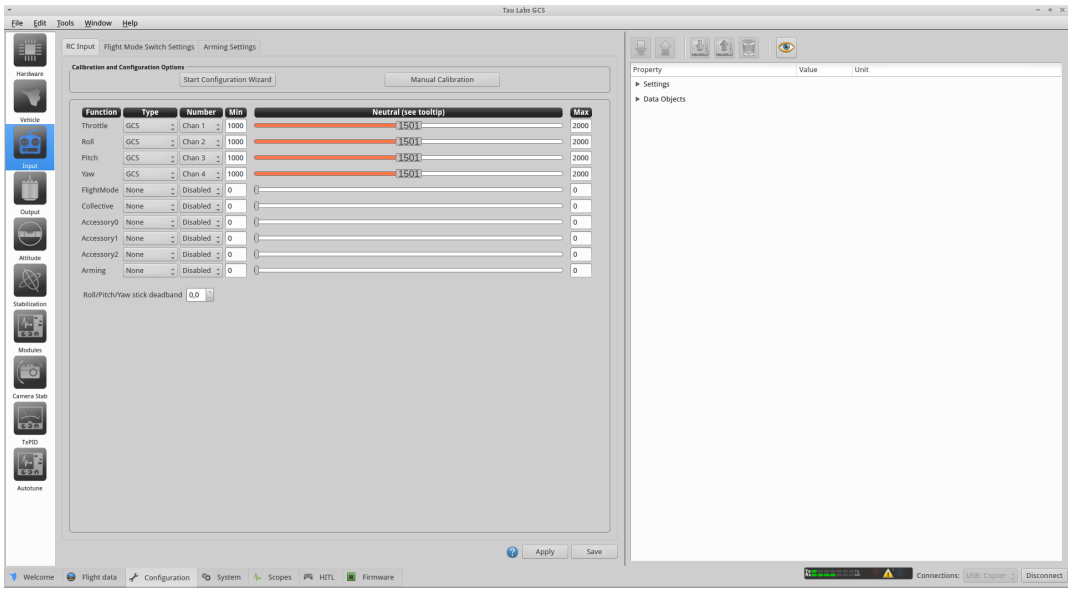


Figure 3.4: Taulabs GCS input configuration

a uniform distribution on the map space and then starts a cycle in which it continuously reads data on the topics "/IR_Readings" and "/MovementCommands". When data enter the "/IR_Readings" topic, the algorithm knows that there is fresh information on the sensors. It weighs all the particles, making the ones where received sensors data is most probable to happen heavier; the particles are then resampled. In this step heavier particles will more likely be selected to continue into the next generation. The resampling process consists in computing the cumulative weight for each particle on the set and repeatedly select numbers from a uniform distribution $U(0,1)$. A particle is resampled if the cumulative weight is bigger than threashold value and only one particle is resampled for each number. When data enters the "/MovementCommands" topic the algorithm understands that a movement command was issued and applies it to the particle set. The movement data is an array of four numbers, in this order: throttle, roll, pitch and yaw. The bounds of each command are configurable in the TauLabs ground control station 3.4. When new data arrives, the localization algorithm passes it through the UAV dynamics model in order to obtain the expected movement, applies some disturbances and updates all the particles. Particles close to an obstacle are truncated to that obstacle border, we chose to do this because in spite of being impossible to stand too close to an obstacle, the number of particles has some influence in the next generation so this decision aims to conserve diversity without accepting the impossible.

Although the method above explains how the robot localizes himself on a 2D map, an UAV operates in a 3D space. We opted to separate the altitude control as this focus do not focus in the implementation of a 6-dimensional localization algorithm and, in further works, the method can be updated to localize the UAV in 3 or 6 dimensions. The altitude is measured by a LV-MaxSonar-EZ0 sonar sensor pointing downwards, connected to the UART2 Rx pin in the BB. The LV-MaxSonar-EZ0 has a serial connection that outputs data on a RS232 format but with inverted

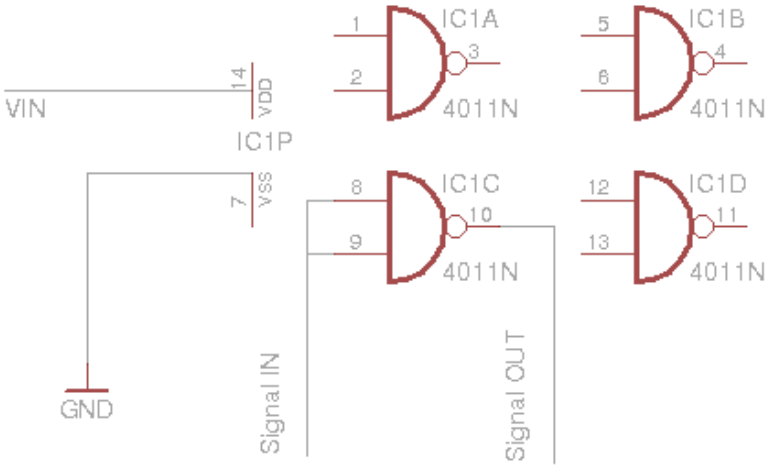


Figure 3.5: Sonar signal inverter

signals; this means that the data had to be inverted before feed the BB. In order to achieve this we designed the following circuit 3.5.

We implemented a PI controller to command the altitude of the UAV; the PI is a PID with null contribution from the derivative component. The controller increases the throttle based on the difference between the set altitude and the measured by the sonar.

Another challenge faced in the development of this layer was the communication protocols. The firmware uses a protocol called UAVTalk, however there was no module in ROS that could interpret UAVTalk and publish the information on other formats for the remaining ROS modules. "UAVTalk is a highly efficient, extremely flexible and completely open binary protocol designed specifically for communication with UAVs."[2] The protocol works by sending binary object between two ends; in each side of the connection link, there is an object manager that contains a set of objects. These objects are used to store data inside the applications. When a layer needs to communicate with the other, an object is compressed and sent to the other side of the link where it rewrites the other end object. This guarantees that the protocol can be used for many different kind of objects and is data independent as the packing and unpacking of the data is done by the manager and not by the protocol itself. UAVTalk messages are composed as seen in table 3.2.

The only restriction imposed is that the Object ID has to be previously agreed upon by both side of the link and currently it does not support multiple senders and receivers, only connection with two entities. As state above, ROS did not support UAVTalk or UAVObjects out-of-the-box, so we had to create a library that could not only use the protocol, but also keep a live set of UAVObjects. We wanted to make it generic enough that it could be used in other projects, therefore not ROS dependent, and, this way, contribute to the OpenPilot/TauLabs community. This library is based on the Taulabs ground control station; it is divided into two main modules, the UAVObject manager and the UAVTalk link. The manager is a system that keeps the objects alive and hold a reference

[2]http://wiki.openpilot.org/display/Doc/UAVTalk

Table 3.2: UAVTalk message composition

Field	Length (bytes)	Value
Sync Val	1	0x3C
Message type	1	Object (OBJ): 0x20, Object request (OBJ_REQ): 0x21, Object with acknowledge request (OBJ_ACK): 0x22, Acknowledge (ACK): 0x23, Negative-Acknowledge (NACK) : 0x24. Note: The most significant 4 bits indicate the protocol version (v2 currently)
Length	2	Not needed since object ID predicts this but useful for blind parsing. Length of header and data, not checksum.
Object ID	4	Unique object ID (generated by parser, also used for framing and version control)
Instance ID (optional)	2	Unique object instance ID. Only present in UAVObjects that are NOT of type single instance
Data	0-255	Serialized (packed) object. The length of data is inherent knowledge about that particular object's size as identified by ObjectID.
Checksum	1	CRC-8 checksum

to their positions in memory; the UAVTalk link parses the messages and send new data packets. A class diagram description of the library can be found in A.1.

3.2.3 Deliberative Layer

The top layer of the system is where all the cognitive tasks are performed. We choose to use the Jason framework because it allows a fast prototyping using a logic language. Next, we describe the difficulties in connecting mentioned system with the ROS running in the executive layer.

Jason is an interpreter of AgentSpeak [RHB07], an agent-oriented programming language. Agentspeak is based on logic programming and implements the BDI architecture. This makes Jason a very interesting engine for implementing cognition on the UAV, but the Jason environment offers few connectivity. First, it is written in Java and therefore needs a Java Virtual Machine (JVM) to run which makes it not compatible with ROS. Second, the state inference cycle runs in a discrete time fashion and the perception/actions can only be given at certain times.

The reasoning cycle for a Jason agent can be found in either [RHB07] or in a simplified version in Figure 3.6. In the Figure 3.6 we tried to highlight the parts of the cycle that are most important in this dissertation. *Perceive* and *act* are the two methods that needed to be overwritten in order to allow communication between the Deliberative and Executive layers.

In order to allow the Java environment (where the Jason is running) and the ROS environment (where the perceptions are being collected and the actions executed), we used Inter-Process Communication (IPC) methods. IPC allows processes to share information with the help of the

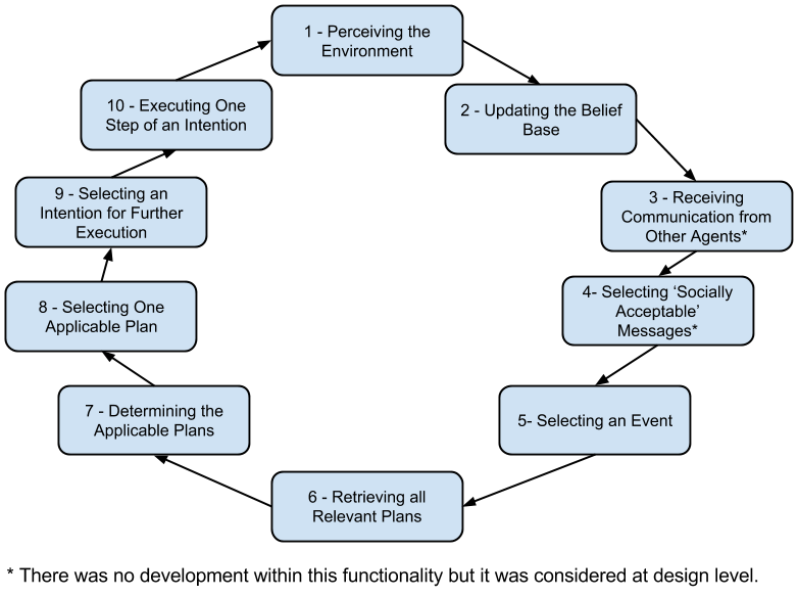


Figure 3.6: Jason agent reasoning cycle

operation system. After studding several options between the IPC family we choose to use named pipes.

The logic solution is to encompass the IPC under the Executive layer. There are two modules that communicate with the Deliberative Layer, “allmaPerceive” and “allmaAct”. “allmaPerceive” collects information about the environment and stores the most recent data. When the perceive method is called in the Jason agent, it relays the collected information making sure to pass only the most recent values. The “allmaAct”, on the other hand, publishes the decisions taken by the reasoning engine to the rest of the ROS when the algorithm decides on a certain action. It keeps sharing the same decision until a new one arrives, this way if the above layer crashes the system still retains some robustness. This module could be extended to also function as a fail-safe system, that would drive the UAV to a safe position where it could wait for rescue.

3.3 Summary

The proposed architecture is adapted from both the robotics and the MAS world concerning with the two main requirements of our project that are the modularity and flexibility of the architecture. The result was a system divided in layers where each has its own function and responsibility. These layers separate the system by degrees of deliberation and computational power resulting in a top to bottom chain of command. By combining the layers, it is possible to make an UAV autonomous allowing researches and developers to create robots capable of helping people, surveillance areas, or answer disasters. We ended up with a autonomous quadcopter that runs both ROS and Jason on-board, can localize itself in an environment and is a great tools for research.

Chapter 4

Test & Results

The current chapter presents our tests and the consequent results. As we had a hardware failure in the beginning of the test phase we had to divide the scenarios into components and test each separately. Therefore this chapter is divided by components.

4.1 IR sensors integration

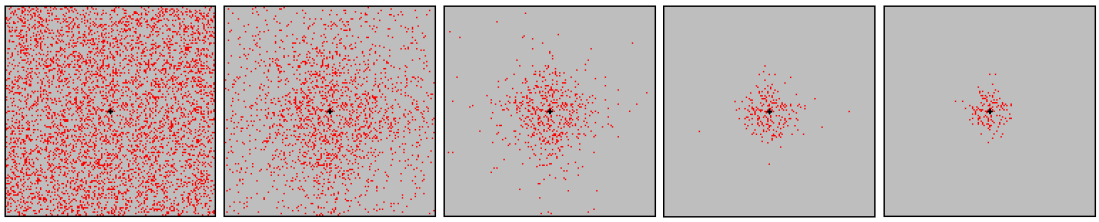
To test the accuracy of the ROS module that reads the IR sensors as well as confirming that those values were approximate enough for the localization algorithm we mounted a rig that allowed us to configure the distances with approximation of less than a centimeter of error and record the values for latter analysis.

The rig has 4 IR sensors mounted simultaneously but we present the results separately for each axis. The table below was created from a sample set of 10000 readings.

From this we can determine that if the distances are small, i. e. below 48cm, the error is irrelevant as it is smaller than the accuracy we have from the rig. The problem found is that at bigger distances the error increases which might reduce the accuracy of the localization algorithm. Fortunately the farther away the robot is from an obstacle the less important accuracy is for localization, as it is simpler to accommodate higher levels of uncertainty.

Table 4.1: IR sensors test results

	Measurement 1	Measurement 2	Measurement 3	Measurement 4
Actual Distance	43	48	71	96
Measured distance	43.44	48.22	73.78	106.51
Variance	0.25	0.22	1.12	3.60
Error	0.44	0.22	2.78	10.51



(a) (b) (c) (d) (e)

Figure 4.1: Localization algorithm run with robot stopped in the center of the environment

4.2 Localization

In order to test the localization algorithm we devised a test environment from which we made some sensor readings and fed those readings to the localization algorithm. To verify the state of the algorithm at each time step we also developed a visualization program that shows the multiple particles inside the environment.

The chosen environment was a square box with 149 centimeters on the side. This environment is simple, easily replaceable and follows all the assumptions stated in the previous chapter. Also in the previous chapter we described the localization algorithm in use, it required the user to input a number of particles and a ratio to valorize some particles over others. The chosen values for those were 5000 and 2 for number of particles and differentiation power respectively.

The output of the visualizer can be seen in figure 4.1. It took only 5 iterations to get a area of about 20cm, the quadcopter width, and an error of 3cm, however each iteration took about 1.16 seconds which is not viable for real world applications. Although no rigorous test was performed we can clearly state that a quadcopter needs to update the command inputs with a frequency of less than 1 second.

To determine if the algorithm was real time capable we tested different number of particles, retaining the 5 iterations rule constant. The results can be seen in figure 4.2. The graph shows the time taken per iteration and the difference between the calculated and actual position after the 5 iterations. The time taken is, as expected, the line that increases with the number of particles. The difference is the line than decreases and in our test got to around 3cm. We decide to not test more than 5000 particles because as was already said more than a second per iteration is too much.

The optimal number of particles in this environment and within 5 iteration time is around 3250 which takes us to a 5cm error. However that would take half a second per iteration which in turn would take longer to achieve good results. As this seemed a bit to high for airborne robots we concluded that a lower number of particles is more advantageous. In spite of losing 10% of the quadcopter width in accuracy it should not hurt the overall reasoning of the Deliberative layer.

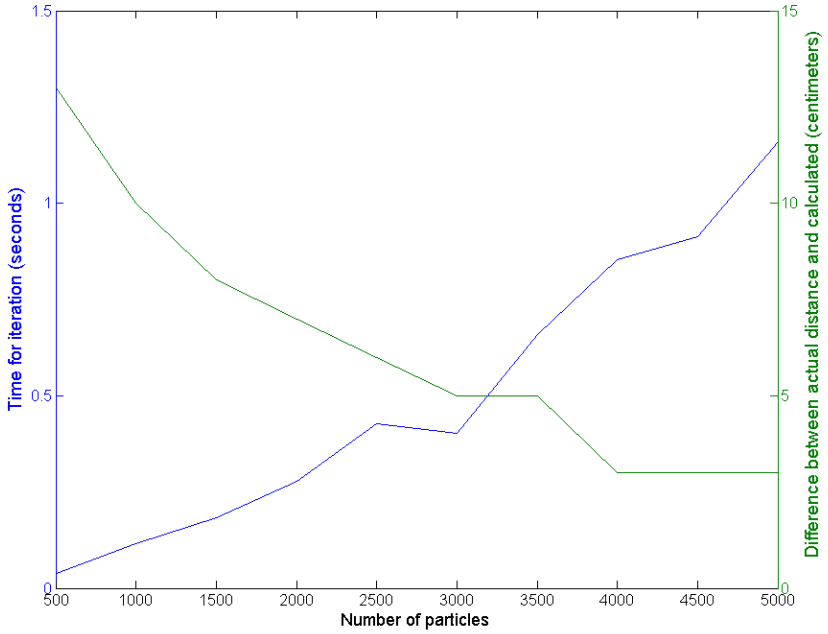


Figure 4.2: Localization algorithm performance comparison

4.3 Jason

In this section we will illustrate some performance pointers of Jason interpreter running on the Beaglebone-based platform. As none of the agents developed by us represented the complexity of a working one we decide to stress test our platform using a gold miners[1] sample. We can consider that the agent controlling the quadcopter is one of the miners and the rest are virtual agents spawned to support certain parallel tasks. Also some of the challenges in the sample can be correlated to real life situations. We are interested in guaranteeing that our system would run in real time.

The computational weight of the entire process was then probed with jvisualvm[2] to retrieve the figures 4.3 and 4.4.

As can be seen the processor usage never overcomes the 70% which leaves some freedom for more agents or more demanding tasks. Also the processing power is constant between 40% and 50% reinforcing the previous statement. The only situation when the processing power overcomes these values is when there was a lot of communication between the agents. The second figure shows the heap usage and size for the duration of the test. The values is very uncertain between four and six megabytes but the important aspect is that it never overtakes the 8MB limit. This is important because other parts of the system, e.g. localization's map, required large pools of memory to run.

[1] http://jason.sourceforge.net/Jason/Examples/Entries/2007/6/21_Gold_Miners_(Jomi_Hubner_and_Rafael_Bordini).html
[2] http://docs.oracle.com/javase/6/docs/technotes/tools/share/jvisualvm.html

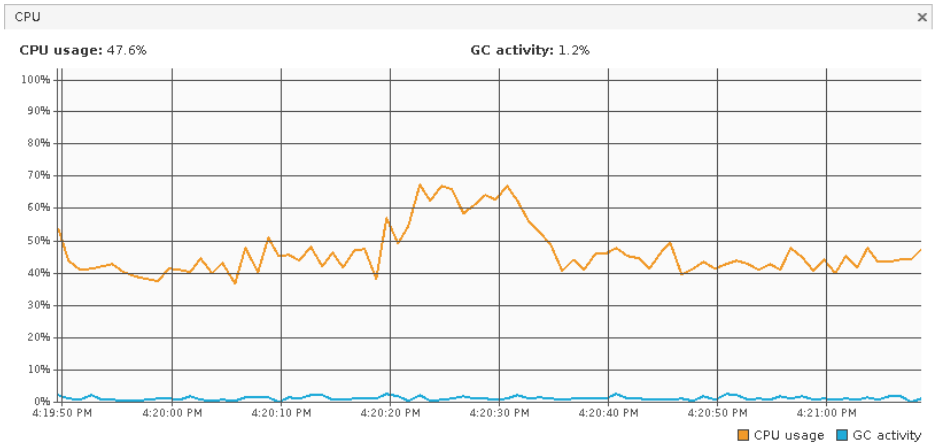


Figure 4.3: CPU usage running Jason

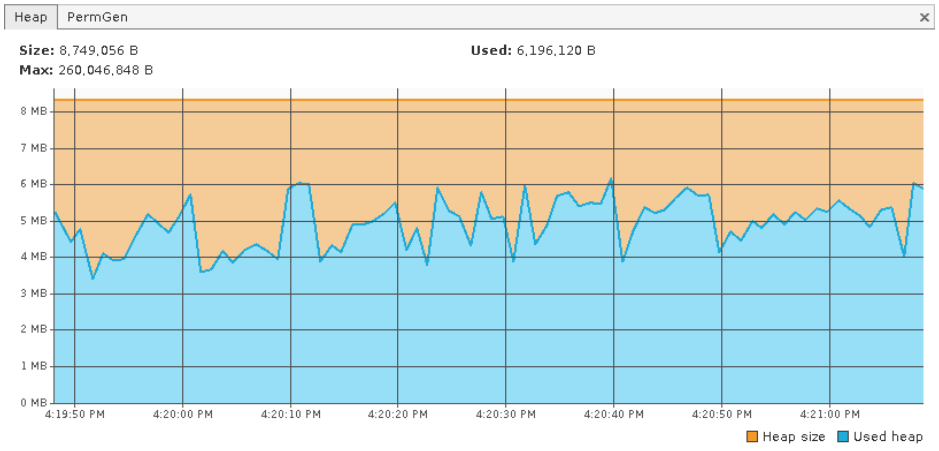


Figure 4.4: Heap memory usage running Jason

These results show that the Jason interpreter even when running aboard the quadcopter is able to maintain enough responsiveness to work as a deliberative engine.

4.4 Summary

The tests showed that each individual part works and is able to perform its assigned tasks. In spite of not being able to experiment the system as a whole we can assume that it would properly work with some tinkering.

The overall adjust of the system might require some effort as there are many values to change. But it is certain that the chosen path was correct.

The robot is able to measure distances to nearest obstacles, use those measurements to localize himself and pass the conclusions into the reasoning engine. This allows the quadcopter to identify possible drifts and correct them, the biggest challenge when flying autonomously with these machines.

The next test we intended to make was to assemble the system and let the quadcopter perform tasks like take-off and hold a position or follow a predetermined path.

Chapter 5

Conclusion

In this chapter we will take some conclusions about the developed project, the chosen path, and the problems faced throughout this dissertation.

5.1 Final Remarks

After studying both the robotics and the agents fields, it was concluded that both offer similar problems and solutions; this allows for a combination of methods that result in a simpler, yet stronger, system. In the state of the art, we identified clear ways this topic is trying to evolve. Some are going into the field of assistance robots that do some tasks on place of the humans. Others are trying to improve upon the current sensoring techniques to allow quadcopters to better understand their environment.

Here, the chosen path was one that lead to a modular architecture where research in new fields causes minor implementation overhead. By simplifying the deployment process, the bridge between development and test can be shortened and the research accelerated. Although this project still possesses many assumptions that cannot be considered true in the real world, this is a first step towards a complete autonomous and intelligent quadcopter.

We implemented an architecture that tried to conjugate the robots and MAS worlds together. The final product is an autonomous quadcopter with localization abilities that would be able to fly on its own. Unfortunately we run into some hardware failures and resource limitations which were not solved in time to finish our testing phase. Nevertheless, all tested parts already show relevant results. In table 5.1 the reader can consult a SWOT analysis of the project, which clearly states the crucial points of the research performed.

Some of the outcomes of this dissertation are already being used in other research groups. We received notice that an aerospace research group at Georgia Institute of Technology is using our uavtalk library to interface with an Openpilot CC3D.

Also, a paper introducing the work done here was accepted at SIMUTools2014.

Table 5.1: Summary SWOT analysis

	Helpful (to achieve the objective)	Harmful (to achieve the objective)
Internal origin (project attributes)	• Modular architecture; • easy to replicate; • low-cost platform; • well defined and relevant contributions.	• Not thoroughly tested; • hardware choices do not fill their purpose completely.
External origin (environment attributes)	• Different path from major researchers; • high interest from society.	• Law policies might stop further advancements (on US already are).

5.2 Further improvements

The current state of the project is above everything untested. The main improvement to do right way would be to test all the parts together and in stressed environments. This should make some bugs surface, which would in place raise some more improvements.

Regarding improvement that we are already aware off, we have the full implementation of the UAVTalk protocol. Currently, the Taulabs firmware does not use the full potential of UAVTalk, multiple instance objects, for example, are not yet needed in the UAV. As we had only a short time frame for our implementation, we also do not implement this feature in the UAVTalk library; one possible improvement would be to extend the library so that it could be used in the future.

Another possible improvement would be a set of tool to easily configure the system. There are many variables that need tinkering which can only be done by trial and error. As examples of this, we have the PID coefficients, the number of particles, and so forth. One last thing that is worthy mentioning is the quadcopter parts, the frame and rotors used are not very indicated to stable flight. A bigger and more powerful quadcopter would be more stable and allow us to run on other environments.

5.3 Future works

The main purpose of this project was to launch new research project within the faculty so this section is very broad in ideas. Currently the UAV uses 4 IR sensors to localize itself and a possible work that can be seen from here is to change the main perception technique to something more applicable to different UAVs; a laser range finder, for instance, would be applicable to multiple UAV configuration and still function similarly. Other possibility would be to move into vision-based sensing and use stereoscopic vision to retrieve the depth data.

Other area where multiple research project could blossom is the control theory one. The quadcopter has serious stability challenges that could be counter with better control algorithms and techniques. Also, there are other kind of UAV where this dissertation architecture can be applied for which the controllers are difficult to design.

Still the most important area where future research could be performed is MAS. As was stated in the introduction, UAVs are starting to attract many research teams around the world and the application of MAS techniques to these systems is something which will be done in a near future. The methods of cooperation and coordination present in the MAS research could bring the idea of UAVs swarms driving the world forward from paper to real life.

5.4 Lessons Learned

Every project as some chaos involved and it is impossible to plan for everything ahead; this dissertation was no exception. From hardware faults to crashes, we had everything. The most important lesson we learned was always buy two; redundancy is key to project planning, one chance is never

enough and a project that involves hardware will have failures from that same hardware. Other important thing that we can take home is that the order in which the tasks are scheduled is important to diminish risks; if we had performed the most uncertain tasks first, we would have found some problems sooner and had more time to deal with them.

References

[AM12] Brian DO Anderson and John Barratt Moore. *Optimal filtering*. DoverPublications. com, 2012.

[Are13] Flying Machine Arena. History - flying machine arena. "http://www.flyingmachinearena.org/history/", 2013. 2013-06-25.

[ASD12] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1917–1922. IEEE, 2012.

[Ast95] Karl J Astrom. Pid controllers: theory, design and tuning. *Instrument Society of America*, 1995.

[BBP06] B Bluteau, R Briand, and O Patrouix. Design and control of an outdoor autonomous quadrotor powered by a four strokes rc engine. In *IEEE Industrial Electronics, IECON 2006-32nd Annual Conference on*, pages 4136–4240. IEEE, 2006.

[Bou07] Samir Bouabdallah. *Design and control of quadrotors with application to autonomous flying*. PhD thesis, 2007.

[BPRM11] RA Braga, Marcelo Petry, Luís Paulo Reis, and Antonio Paulo Moreira. Intellwheels: modular development platform for intelligent wheelchairs. *Journal of rehabilitation research and development*, 48(9):1061, 2011.

[Bro86] Rodney Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, 1986.

[Bur10] Leon K Burkamshaw. Towards a low-cost quadrotor research platform. Master's thesis, 2010.

[CDL04] Pedro Castillo, Alejandro Dzul, and Rogelio Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *Control Systems Technology, IEEE Transactions on*, 12(4):510–516, 2004.

[DD09] Guillaume Ducard and Raffaello D'Andrea. Autonomous quadrotor flight using a vision system and accommodating frames misalignment. In *Industrial embedded systems, 2009. SIES'09. IEEE international symposium on*, pages 261–264. IEEE, 2009.

[Dia93] Esmeralda M. L. Dias. *Robótica: conceitos gerais e sistemas Rhino e Rob 3i*. Fundação para a Divulgação das Tecnologias de Informação, 1993.

[DML02] Scott A DeLoach, Eric T Matson, and Yonghua Li. Applying agent oriented software engineering to cooperative robotics. In *FLAIRS Conference*, pages 391–396, 2002.

[DSSP09] Lavindra De Silva, Sebastian Sardina, and Lin Padgham. First principles planning in bdi systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1105–1112. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent agents III agent theories, architectures, and languages*, pages 21–35. Springer, 1997.

[G+98] Erann Gat et al. On three-layer architectures. *Artificial intelligence and mobile robots*, pages 195–210, 1998.

[Gas87] Les Gasser. Distribution and coordination of tasks among intelligent agents. In *First Scandinavian Conference on Artificial Intelligence*, Tromsø, Normay, March 1987.

[GSGA09] Rahul Goel, Sapan M Shah, Nitin K Gupta, and N Ananthkrishnan. Modeling, simulation and flight testing of an autonomous quadrotor. In *IISc Centenary International Conference and Exhibition on Aerospace Engineering, ICEAE, Bangalore, India*, pages 18–22, 2009.

[GSS93] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

[HDL92] M Hassoun, Y Demazeau, and C Laugier. Motion control for a car-like robot: potential eld and multi-agent approaches. In *Proc. of the Int. Workshop on Intelligent Robots and Systems. IEEE. Raleigh, NC (USA)*. Citeseer, 1992.

[HHWT09] Haomiao Huang, Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3277–3282. IEEE, 2009.

[HWB10] Armin Hornung, Kai M. Wurm, and Maren Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[Jen01] Patric Jensfelt. *Approaches to mobile robot localization in indoor environments*. PhD thesis, 2001.

[JSW98] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[K+60] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[KMK12] Alex Kushleyev, Daniel Mellinger, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Robotics: Science and Systems*, July 2012.

REFERENCES

[LCK03] Dongheui Lee, Woojin Chung, and Munsang Kim. A reliable position estimation method of the service robot by map matching. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2830–2835. IEEE, 2003.

[LLK+12] Daewon Lee, Hyon Lim, H Jin Kim, Youdan Kim, and Kie Jeong Seong. Adaptive image-based visual servoing for an underactuated quadrotor system. *Journal of Guidance, Control, and Dynamics*, 35(4):1335–1353, 2012.

[LMK11] Quentin J. Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. *Robotics: Science and Systems*, June 2011.

[LPLK12] Hyon Lim, Jaemann Park, Daewon Lee, and HJ Kim. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. *Robotics & Automation Magazine, IEEE*, 19(3):33–45, 2012.

[LSCU12] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1043–1050. IEEE, 2012.

[LSHD11] Sergei Lupashin, Angela Schollig, Markus Hehn, and Raffaello D'Andrea. The flying machine arena as of 2010. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2970–2971. IEEE, 2011.

[MD12] M. Muller and R. D'Andrea. Critical subsystem failure mitigation in an indoor uav testbed. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 780–785. IEEE, 2012.

[Mel12] Daniel Mellinger. *Trajectory generation and control for quadrotors*. PhD thesis, University of Pennsylvania, 2012.

[MK11] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[MMK12] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[MMLK10] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *Robotics Automation Magazine, IEEE*, 17(3):56 –65, 2010.

[MNOH99] Fumio Mizoguchi, Hiroyuki Nishiyama, Hayato Ohwada, and Hironori Hiraishi. Smart office robot collaboration based on multi-agent programming. *Artificial Intelligence*, 114(1):57–94, 1999.

[MSK10] D. Mellinger, M. Shomin, and V. Kumar. Control of quadrotors for robust perching and landing. In *Proceedings of the International Powered Lift Conference*, Oct 2010.

[MTFP11] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2992–2997. IEEE, 2011.

[MTH+12] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

[Nic04] Eryk Brian Nice. *Design of a four rotor hovering vehicle*. PhD thesis, Cornell University, 2004.

[NWB+03] Issa A Nesnas, Anne Wright, Max Bajracharya, Reid Simmons, Tara Estlin, and Won Soo Kim. Claraty: An architecture for reusable robotic software. In *AeroSense 2003*, pages 253–264. International Society for Optics and Photonics, 2003.

[ODC08] Helcio RB Orlande, George S Dulikravich, and Marcelo J Colaço. Application of bayesian filters to heat conduction problem. *EngOpt*, pages 1–5, 2008.

[PD09] Oliver Purwin and Raffaello D'Andrea. Performing aggressive maneuvers using iterative learning control. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1731–1736. IEEE, 2009.

[PPS+08] JL Posadas, JL Poza, JE Simó, G Benet, and F Blanes. Agent-based distributed architecture for mobile robot control. *Engineering Applications of Artificial Intelligence*, 21(6):805–823, 2008.

[RG+95] Anand S Rao, Michael P Georgeff, et al. Bdi agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319. San Francisco, 1995.

[RHB07] Michael Wooldridge Rafael H. Bordini, Jomi Fred Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, Ltd, 2007.

[RMHD12] Robin Ritz, Mark W Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative quadrocopter ball throwing and catching. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4972–4978. IEEE, 2012.

[RNC+95] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[RYAS09] Osamah A Rawashdeh, Hong Chul Yang, Rami D AbouSleiman, and Belal H Sababha. Microraptor: A low-cost autonomous quadrotor system. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, California, USA*, 2009.

[SALD10] A Schollig, Federico Augugliaro, Sergei Lupashin, and Raffaello D'Andrea. Synchronizing the motion of a quadrocopter to music. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3355–3360. IEEE, 2010.

[SdSP06] Sebastian Sardina, Lavindra de Silva, and Lin Padgham. Hierarchical planning in bdi agent programming languages: A formal approach. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1001–1008. ACM, 2006.

[ŠPV+08] David Šišlák, Michal Pěchouček, Přemysl Volf, Dušan Pavlíček, Jiří Samek, Vladimír Mařík, and Paul Losiewicz. Agentfly: Towards multi-agent technology in free flight

air traffic control. In *Defence Industry Applications of Autonomous Agents and Multi-Agent Systems*, pages 73–96. Springer, 2008.

[SV00] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[TFBD01] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[Thr03] Sebastian Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35, 2003.

[TKH09] John Tisdale, Zuwhan Kim, and J Hedrick. Autonomous uav path planning and estimation. *Robotics & Automation Magazine, IEEE*, 16(2):35–42, 2009.

[TM06] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *Control Systems Technology, IEEE Transactions on*, 14(3):562–571, 2006.

[TU09] Andreas Tolk and Adelinde M Uhrmacher. Agents: Agenthood, agent architectures, and agent taxonomies. *Agent-Directed Simulation and Systems Engineering*, 78, 2009.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[Woo08] Michael Wooldridge. *An introduction to multiagent systems*. Wiley. com, 2008.

REFERENCES

Appendix A

Class Diagrams

A.1 UAVTalk Library

This appendix shows a class diagram for the developed UAVTalk library. This library was a major contribute for the community and is already being used by other research groups.

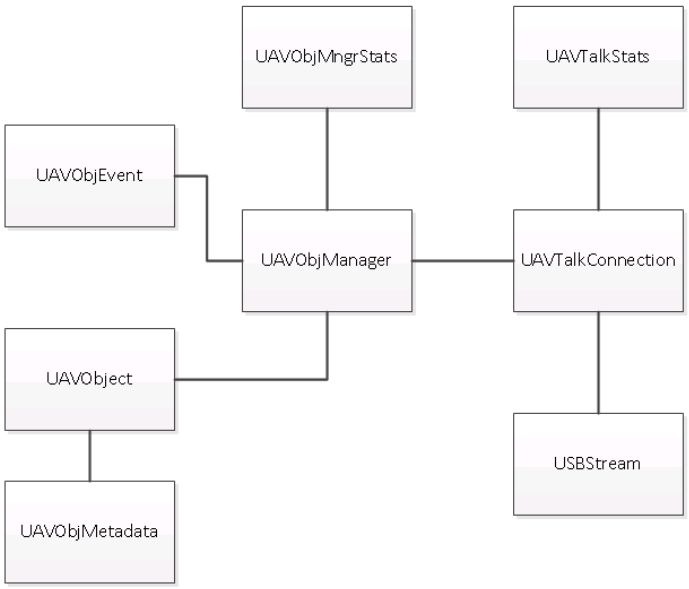


Figure A.1: UAVTalk Library class diagram

Annex I

Spreadsheets

I.1 GP2Y0A02YK Infrared Sensor

This section contains the datasheet for our infrared sensors.

GP2Y0A02YK

Long Distance Measuring Sensor

■ Features

1. Less influence on the colors of reflected objects and their reflectivity, due to optical triangle measuring method
2. Distance output type
   (Detection range:20 to 150cm)
3. An external control circuit is not necessary
   Output can be connected directly to a microcomputer

■ Applications

1. For detection of human body and various types of objects in home appliances, OA equipment, etc

■ Absolute Maximum Ratings

(T_a=25°C)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	−0.3 to +7	V
*1 Output terminal voltage	V_O	−0.3 to V_{CC} +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

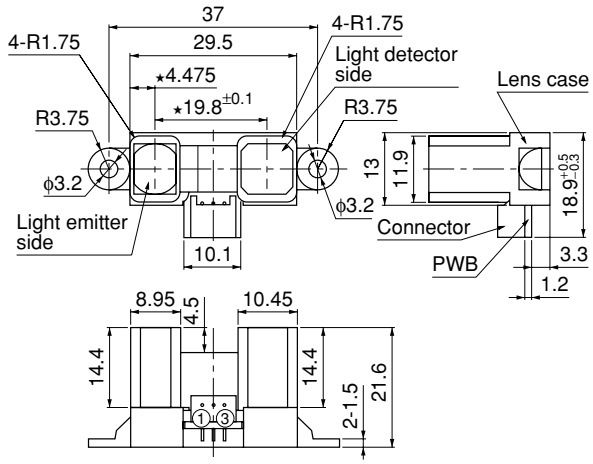
*1 Open collector output

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating Supply voltage	V_{CC}	4.5 to 5.5	V

■ Outline Dimensions

(Unit : mm)



Terminal connection
① V_O
② GND
③ V_{CC}

❊ The dimensions marked ⋆ are described the dimensions of lens center position.
❊ Unspecified tolerance : ±0.3mm

■ Electro-optical Characteristics

(T_a=25°C, V_{CC}=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range	ΔL	*2 *3	20	–	150	cm
Output terminal voltage	V_O	*2 L=150cm	0.25	0.4	0.55	V
Difference of output voltage	ΔV_O	*2 Output change at L=150cm to 20cm	1.8	2.05	2.3	V
Average dissipation current	I_{CC}	–	–	33	50	mA

Note) L:Distance to reflective object
*2 Using reflective object:White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio;90%)
*3 Distance measuring range of the optical sensor system

Fig.1 Internal Block Diagram

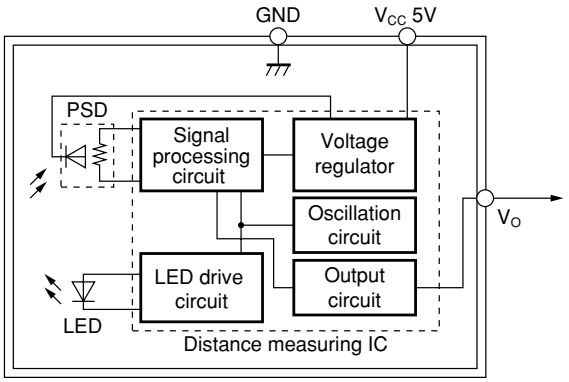


Fig.2 Timing Chart

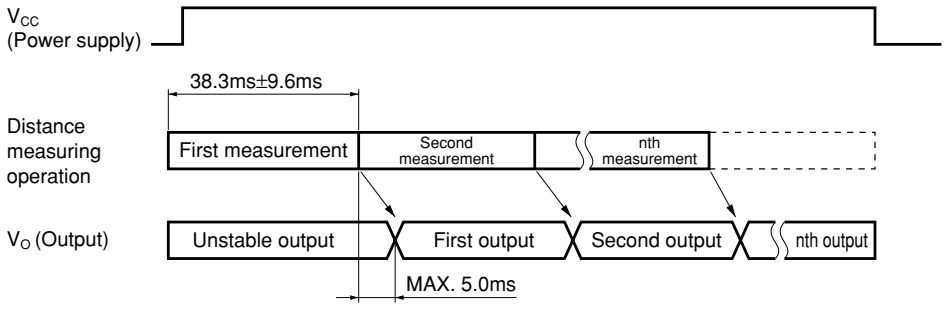
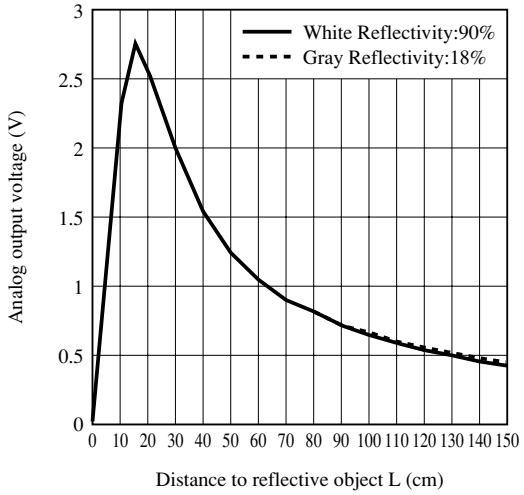


Fig.3 Analog Output Voltage vs. Distance to Reflective Object

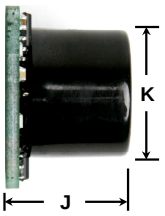


I.2 LV-MaxSonar-EZ0 Sonar Sensor

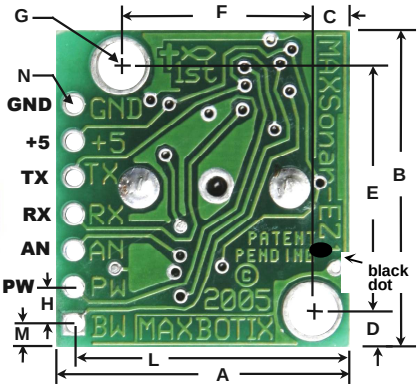
This section contains the datasheet for our sonar sensor.

LV-MaxSonar®-EZ0™ High Performance Sonar Range Finder

With 2.5V - 5.5V power the LV-MaxSonar®-EZ0™ provides very short to long-range detection and ranging, in an incredibly small package. The LV-MaxSonar®-EZ0™ detects objects from 0-inches to 254-inches (6.45-meters) and provides sonar range information from 6-inches out to 254-inches with 1-inch resolution. Objects from 0-inches to 6-inches typically range as 6-inches. The interface output formats included are pulse width output, analog voltage output, and serial digital output.



approximately actual size



A	0.785"	19.9 mm
B	0.870"	22.1 mm
C	0.100"	2.54 mm
D	0.100"	2.54 mm
E	0.670"	17.0 mm
F	0.510"	12.6 mm
G	0.124" dia.	3.1 mm dia.

H	0.100"	2.54 mm
J	0.610"	15.5 mm
K	0.645"	16.4 mm
L	0.735"	18.7 mm
M	0.065"	1.7 mm
N	0.038" dia.	1.0 mm dia.
weight, 4.3 grams		

values are nominal

Features

- Continuously variable gain for beam control and side lobe suppression
- Object detection includes zero range objects
- 2.5V to 5.5V supply with 2mA typical current draw
- Readings can occur up to every 50mS, (20-Hz rate)
- Free run operation can continually measure and output range information
- Triggered operation provides the range reading as desired
- All interfaces are active simultaneously
- Serial, 0 to Vcc, 9600Baud, 81N
- Analog, (Vcc/512) / inch
- Pulse width, (147uS/inch)
- Learns ringdown pattern when commanded to start ranging
- Designed for protected indoor environments
- Sensor operates at 42KHz
- High output square wave sensor drive (double Vcc)

Benefits

- Very low cost sonar ranger
- Reliable and stable range data
- Sensor dead zone virtually gone
- Lowest power ranger
- Quality beam characteristics
- Mounting holes provided on the circuit board
- Very low power ranger, excellent for multiple sensor or battery based systems
- Can be triggered externally or internally
- Sensor reports the range reading directly, frees up user processor
- Fast measurement cycle
- User can choose any of the three sensor outputs

Beam Characteristics

The LV-MaxSonar®-EZ0™ has the most sensitivity of the LV-MaxSonar®-EZ™ product line, yielding a controlled wide beam with high sensitivity. Sample results for measured beam patterns are shown below on a 12-inch grid. The detection pattern is shown for;

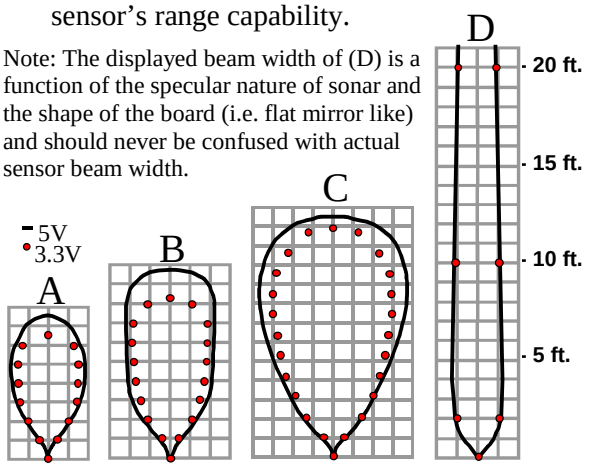
(A) 0.25-inch diameter dowel, note the narrow beam for close small objects,

(B) 1-inch diameter dowel, note the long narrow detection pattern,

(C) 3.25-inch diameter rod, note the long controlled detection pattern,

(D) 11-inch wide board moved left to right with the board parallel to the front sensor face and the sensor stationary. This shows the sensor's range capability.

Note: The displayed beam width of (D) is a function of the specular nature of sonar and the shape of the board (i.e. flat mirror like) and should never be confused with actual sensor beam width.



beam characteristics are approximate

Email: info@maxbotix.com
Web: www.maxbotix.com

LV-MaxSonar®-EZ0™ Pin Out

GND – Return for the DC power supply. GND (& Vcc) must be ripple and noise free for best operation.

+5V – Vcc – Operates on 2.5V - 5.5V. Recommended current capability of 3mA for 5V, and 2mA for 3V.

TX – When the *BW is open or held low, the TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital "R", followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232. When BW pin is held high the TX output sends a single pulse, suitable for low noise chaining. (no serial data).

RX – This pin is internally pulled high. The EZ0™ will continually measure range and output if RX data is left unconnected or held high. If held low, the EZ0™ will stop ranging. Bring high for 20uS or more to command a range reading.

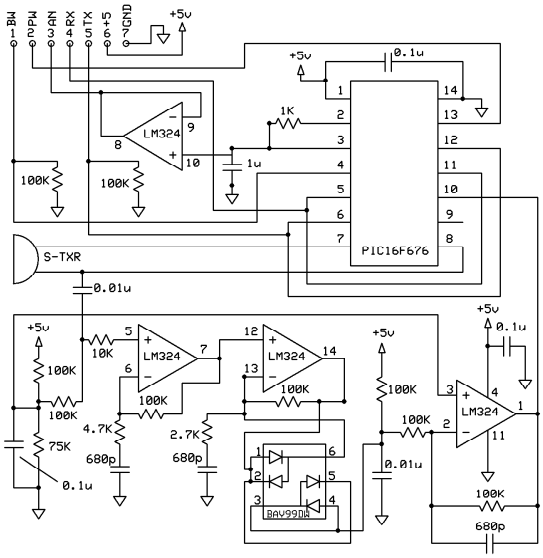
AN – Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.

PW – This pin outputs a pulse width representation of range. The distance can be calculated using the scale factor of 147uS per inch.

BW – *Leave open or hold low for serial output on the TX output. When BW pin is held high, the TX output sends a pulse (instead of serial data), suitable for low noise chaining.

LV-MaxSonar®-EZ0™ Circuit

The LV-MaxSonar®-EZ0™ sensor functions using active components consisting of an LM324, a diode array, a PIC16F676, together with a variety of passive components.



LV-MaxSonar®-EZ0™ Timing Description

250mS after power-up, the LV-MaxSonar®-EZ0™ is ready to accept the RX command. If the RX pin is left open or held high, the sensor will first run a calibration cycle (49mS), and then it will take a range reading (49mS). After the power up delay, the first reading will take an additional ~100mS. Subsequent readings will take 49mS. The LV-MaxSonar®-EZ0™ checks the RX pin at the end of every cycle. Range data can be acquired once every 49mS.

Each 49mS period starts by the RX being high or open, after which the LV-MaxSonar®-EZ0™ sends thirteen 42KHz waves, after which the pulse width pin (PW) is set high. When a target is detected the PW pin is pulled low. The PW pin is high for up to 37.5mS if no target is detected. The remainder of the 49mS time (less 4.7mS) is spent adjusting the analog voltage to the correct level. When a long distance is measured immediately after a short distance reading, the analog voltage may not reach the exact level within one read cycle. During the last 4.7mS, the serial data is sent. The LV-MaxSonar®-EZ0™ timing is factory calibrated to one percent at five volts, and in use is better than two percent. In addition, operation at 3.3V typically causes the objects range, to be reported, one to two percent further than actual.

LV-MaxSonar®-EZ0™ General Power-Up Instruction

Each time after the LV-MaxSonar®-EZ0™ is powered up, it will calibrate during its first read cycle. The sensor uses this stored information to range a close object. It is important that objects not be close to the sensor during this calibration cycle. The best sensitivity is obtained when it is clear for fourteen inches, but good results are common when clear for at least seven inches. If an object is too close during the calibration cycle, the sensor may then ignore objects at that distance.

The LV-MaxSonar®-EZ0™ does not use the calibration data to temperature compensate for range, but instead to compensate for the sensor ringdown pattern. If the temperature, humidity, or applied voltage changes during operation, the sensor may require recalibration to reacquire the ringdown pattern. Unless recalibrated, if the temperature increases, the sensor is more likely to have false close readings. If the temperature decreases, the sensor is more likely to have reduced up close sensitivity. To recalibrate the LV-MaxSonar®-EZ0™, cycle power, then command a read cycle.

Email: info@maxbotix.com
Web: www.maxbotix.com