UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Development of a guidance and automatic positioning system for builder drones with a theodolite

Dissertation presented by
Nicolas SORENSEN , Rémy VERMEIREN

for obtaining the Master's degree in
Computer science

Supervisor(s)
Pierre LATTEUR, Ramin SADRE

Reader(s)
Sébastien GOESSENS, François WIELANT

Academic year 2017-2018

Abstract

This master's thesis was developed as part of a research led by Pr. Pierre Latteur and his team at the Université catholique de Louvain (UCL), in the Civil and environmental engineering department. The team studies the possibility of masonry work using UAVs. As construction elements adapted to the use of UAVs have already been developed, the next step is the automation of the process. In this context, three complementary automatic positioning systems were developed this year. In this thesis is explained the development of a positioning system using a robotic total station (RTS). In spite of the fact that a real-scale guidance test could not be performed, the positioning system as itself has been proven centimeter accurate. Further work should first test a concrete autonomous flight of a UAV with the three systems implemented, then perform a masonry test with the adapted construction elements.

Acknowledgements

We would like to thank:

Pierre Latteur, who supports the global project, supported us throughout the year and was easily available if we had any trouble concerning the project;

Sébastien Goessens, who also supports the project, followed us closely during this year and was always available whenever we had any question or request. Finally, he facilitated the contact and organization with people outside the project, which saved us quite some time;

Ramin Sadre, who followed the project and was available for any IT specific trouble we had;

Geoffrey Mormal, who in spite of his tight schedule, still spent a few days helping us installing and configuring ALX Systems' framework. We would also like to thank him for providing us his expertise and sharing his knowledge in the field of UAVs automation;

Joseph Lachina, for helping us integrating our solutions to ALX Systems' framework and providing his expertise on embedded computers;

David Güth, for being the contact person of Leica for any question about their products and for allowing us to borrow a Leica MS60;

Éric Mourmaux, for making himself available to let us try a Leica MS50;

Artyom Maxim, for his availability for any problem related to his script and for bringing knowledge on his master's thesis;

Frédéric, **Thibault** and **Zakariya**, for the nice atmosphere within the team and the fruitful collaboration throughout the project.

Nicolas Sorensen I would like to personally thank my parents and my brother for their support throughout this year and throughout my studies in general. Thanks also to all my friends for the good times spent in Louvain-la-Neuve during those five years.

Rémy Vermeiren I would like to particularly thank my mom for lending her car when necessary, Fanny, my friends and all my family for their support during my studies.

Contents

1 Introduction **11**

1.1 Context . . . 11
1.2 Problem . . . 12
1.2.1 Positioning systems on this project . . . 13
1.2.2 Guidance with ALX Systems' system . . . 16
1.3 Objectives . . . 16
1.4 Resources . . . 17
1.4.1 Workspaces . . . 17
1.4.2 Software . . . 17
1.4.3 Hardware . . . 17
1.5 Related work . . . 18
1.5.1 ETH Zurich (2011) . . . 18
1.5.2 Latteur et al. (2015, 2016) [1, 2, 4] . . . 19
1.5.3 Nehri N. and Paques A. (2016) [8] . . . 20
1.5.4 Maxim et al. (2017) [9] . . . 21

2 Positioning system **23**

2.1 Automatic prism location with the total station . . . 23
2.1.1 Total station . . . 23
2.1.2 Prism . . . 25
2.1.3 RTS robotization capabilities . . . 26
2.2 Automation of the positioning system . . . 26
2.2.1 Interaction between a computer and the total station . . . 26
2.2.2 GeoCOM protocol . . . 27
2.2.3 Position and track the prism . . . 30
2.3 Coordinate system . . . 34
2.4 TCRP1203 performance evaluation . . . 34
2.4.1 Measurements . . . 34
2.4.2 Tracking speed . . . 35

3 Guidance system **41**

3.1 Objective . . . 41
3.1.1 General architecture . . . 41
3.2 Autonomous flight Operating Systems . . . 42

3.2.1 ROS – Robot Operating System . . . 42
3.2.2 ALX Systems framework . . . 42
3.3 Drones . . . 44
3.3.1 Compass necessity and related issues . . . 45
3.4 System architecture . . . 46
3.5 Integration of the total station positioning . . . 46

4 Strategy and scenarios **49**
4.1 Strategy . . . 49
4.1.1 Global strategy . . . 49
4.1.2 Positioning systems management . . . 50
4.1.3 Failure scenarios . . . 50
4.2 Testing scenarios . . . 51
4.2.1 Simple scenario . . . 51
4.2.2 Target scenario . . . 51

5 Conclusion **55**
5.1 Solution developed . . . 55
5.2 Difficulties encountered . . . 55
5.2.1 Interferences in the DroneZone . . . 55
5.2.2 PixHawk's Compass . . . 55
5.2.3 ALX Systems . . . 56
5.3 Limitations and future work . . . 56
5.3.1 UAV Positioning with an RTS . . . 56
5.3.2 UAV configuration and guidance . . . 56
5.3.3 Realizing the planned scenarios . . . 57

A Delivered content **63**
A.1 `scripts` folder . . . 63
A.1.1 src . . . 63
A.1.2 tests . . . 63
A.1.3 documentation . . . 63
A.2 `util` folder . . . 63

B Connecting a PC (Windows 10) to the TCRP1203 and MS50-60 **65**
B.1 USB drivers for GEV267,GEV268,GEV269 cable . . . 65
B.1.1 Install the driver on Windows 10 . . . 65

C README.md **67**

D Material **71**

E 3D designs **77**

F Documentation **79**

List of Figures

1.1 Construction elements . . . 12
1.2 Latest drick designed (2017) . . . 12
1.3 Steps in the automated masonry . . . 13
1.4 Image processing part : blocks configuration example . . . 14
1.5 UWB positioning system . . . 14
1.6 Leica TCRP1203 RTS and Leica GRZ4 360° prism . . . 15
1.7 Overview of the complete project . . . 15
1.8 Foam blocks tower . . . 19
1.9 Positioning system using lasers . . . 21
1.10 Fiber composite structure build with UAV . . . 22

2.1 Architecture diagram of the positioning system . . . 23
2.2 Leica TCRP1203 . . . 24
2.3 Leica MS60 RTS . . . 25
2.4 Sokkia ATP1 360° mini-prism . . . 25
2.5 Leica GZR4 360° "big" prism . . . 26
2.6 GeoCOM client/server . . . 27
2.7 Overview of the Client/Server Application (GeoCOM) . . . 29
2.8 Flowchart of the tracking script . . . 31
2.9 "Classic search" search window . . . 32
2.10 Classic search versus PowerSearch . . . 33
2.11 Cartesian coordinates computation . . . 34
2.12 Example of circular path followed by the student . . . 36
2.13 Performance results . . . 37
2.14 Max and average speed for each test . . . 38
2.15 Speed variation during test . . . 39
2.16 Potentiometer utilization . . . 40

3.1 Overview of UAV guidance . . . 41
3.2 Architecture of the electronics on the drone . . . 45
3.3 Roll, pitch and yaw on a UAV . . . 46
3.4 Global architecture of the guidance system . . . 46

4.1 Finite state machine of the global strategy . . . 50
4.2 Target test scenario . . . 52

5.1 Estimation of the orientation of the UAV with UWB beacons 57

E.1 Battery holder 77
E.2 Prism holder 77
E.3 Prism holder utilization. 78
E.4 UWB tag 3D printed case 78

Glossary

ASCII American Standard Code for Information Interchange. Character encoding standard, where characters are encoded on 7 bits, making a total of 128 characters available. 27

ATR Automatic Target Recognition. Automation mode in some Leica robotic total stations (available for TCRP1203 and MS60 models, which were used for this thesis). 26

BIM Building Information Modeling. 12

CAD Computer-aided design. 12

ECAM École Centrale des Arts et Métiers. Higher education institution located in Brussels. 20

EDM Electronic Distance Measurement. Measurement types available in particular with Leica total stations TPS1200 series and Leica Nova series (MS50 and MS60). These types are : IR (with reflector), RL (reflectorless) or LO (Long Range, reflectorless). 34

ETH Eidgenössische Technische Hochschule Zürich, or in English : Swiss Federal Institute of Technology in Zurich. 18

FMA Flying Machine Arena. Research and demonstration platform for fleets of small quadrocopters, in development at ETH Zurich since 2008. 18

IASS International Association for Shell and Spatial Structures. 19

MIT Massachusetts Institute of Technology. 19

OS Operating System. 42

PS PowerSearch. Functionality available on TPS1200 series total stations. It uses a dedicated sensor to perform a faster search to find the prism. 32

RTS Robotic Total Station. Electronic instrument used for surveying. A robotic total station is an electronic theodolite which can be remotely controlled. 11

SLAM Simultaneous localization and mapping. Technique used in robotics to update a representation of an unknown environment and position the robot within it. 56

TCP Transmission Control Protocol. One of the most used protocols for communications across the Internet. 42

TS Total Station. Electronic instrument used for surveying. 23

TST Total Station theodolite. Electronic instrument used for surveying. 23

UAV Unmanned Aerial Vehicle, commonly called "drone". 11

UCL Université catholique de Louvain. 19

UWB Ultra-wideband. Radio technology using very weak, very wide-spectrum signals. 11

WLAN Wireless Local Area Network. 26

Chapter 1

Introduction

1.1 Context

The construction sector benefited from many innovations which arose throughout history, such as the Industrial Revolution and the development of information technology. More recently, additive manufacturing via 3D printers revolutionized the domain : experimental 3D printers aimed at fabricating full-scale buildings are currently in development. However, they are limited in the size of objects they can produce to the extents of their print bed. The research of Pr. Pierre Latteur, T.A. Sébastien Goessens, in collaboration with other researchers, took inspiration from nature to think of a new construction process. Indeed, many species of insects or birds build nests by flying back and forth, carrying various materials such as twigs or mud. In a similar manner, the project aims at using UAVs (Unmanned Aerial Vehicles), commonly called drones, to build structures by using them to carry building blocks.

As ordinary rigid rectangular bricks need a high precision to be placed that drones cannot reach, they are not usable. The team of Pr. Latteur thus developed many designs [1, 2] of blocks aligning themselves naturally, even with an inaccurate drop from a drone (Figure 1.1). They are called "dricks", a portmanteau of "drone" and "brick". The most recent design of a drick (figure 1.2) was developed and successfully tested in 2017 by J-F. Leboutte and V. Parisel, for their master's thesis under the supervision of Pr. Latteur [3].

After the first testable models of dricks were developed, a custom built drone able to carry such blocks has been designed. Whereas tests performed with the drone placing blocks allowing 6.5 cm of inaccuracy were successful [2], they required a skilled pilot for the building process. In order to overcome this problem, Pr. P. Latteur and TA S. Goessens proposed to students in computer science to work on the automation of the drone. The complete automation requires a positioning system, working with a guidance system to control the UAV. In the context of this project, three different positioning systems were developed, by three different teams : our team (Rémy Vermeiren and Nicolas Sorensen) developed an autonomous positioning system using a RTS (Robotic Total Station) to measure the position of the drone ; Zakariya Bouhaddi developed a positioning system using UWB (Ultra-wideband) beacons, to achieve a similar goal as ours ; Thibault Jacques and Frédéric Kaczynski developed a local positioning system using image processing to handle the placement of the blocks more precisely. As a guidance system,

the company ALX Systems[1] helped a lot for the project by providing their proprietary UAV guidance system.

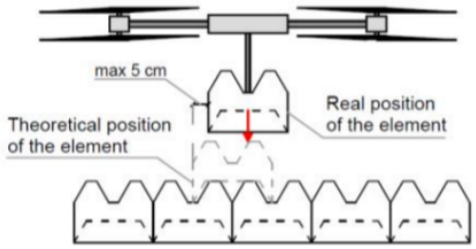


Figure 1.1: Illustration of the inaccuracy allowed for the placement of "dricks" (drone-compatible bricks).

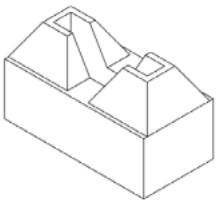


Figure 1.2: Design of a drick by Leboutte J-F. and Parisel V.[3], in particular used by the team working on image processing for their research.

1.2 Problem

The whole process of automated masonry work with drones is divided into four parts (figure 1.3):

1. The project is first designed by the architects and the engineers;
2. It is then modelled into a CAD/BIM[2] model;
3. The previous model is translated into compatible instructions for the drone;
4. The instructions are sent to the drone and the latter begins the construction process.

The three groups described above focused on the fourth part.

[1] `http://www.alxsys.com/`
[2] It basically means that the building is modelled on a computer via a dedicated software.

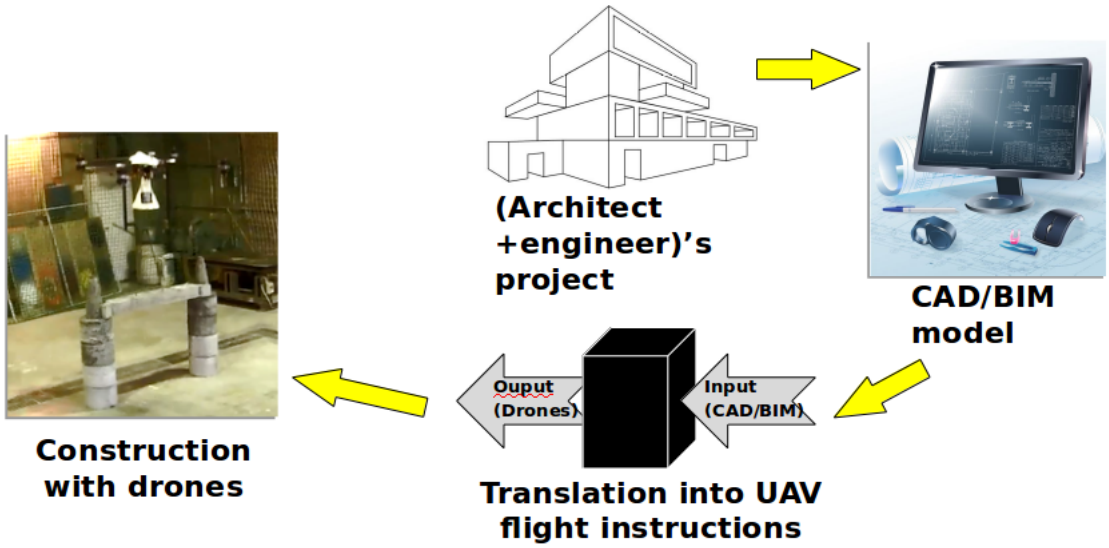


Figure 1.3: All steps involved in the automated masonry work with drones.

1.2.1 Positioning systems on this project

Before entering into the details of our project, we will first talk about the ones of the two other teams.

Computer vision system

Thibault Jacques and Fréderic Kaczynski developed a local positioning system [5], which had to take over the control of the UAV, when the latter had to pick up or to place a block. Their system requires a camera to be placed on the drone. To detect a drick, two methods were developed. The first one uses color points placed on the blocks (figure 1.4) that the computer vision program can detect. The second one uses the ArUco algorithm, which recognizes squared black-and-white markers.

Concerning the performance of their system, the requirements were that it has to have a precision of 5cm or better, which was successfully accomplished: both the color detection technique and the ArUco algorithm have a precision of 3cm along the X and Y axes. Furthermore, both approaches had an error below 3° in the definition of the rotation along the Z axis.

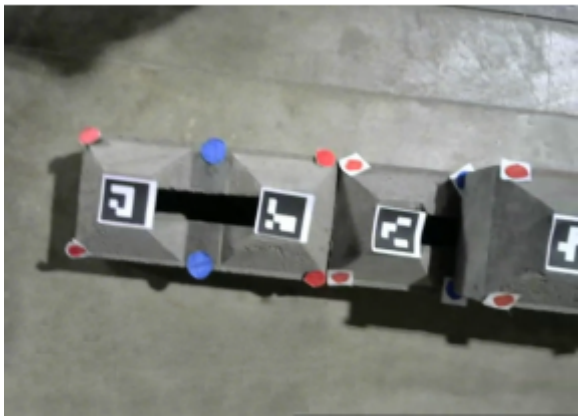


Figure 1.4: Blocks configuration for the image processing part, with both the color points (for color detection) and the black-and-white markers (for the ArUco algorithm).

UWB system

Zakariya Bouhaddi developed a positioning system using UWB beacons [6]. This system was supposed to be used when the UAV has to do broader and less accurate moves. Its requirement was that it had to have a precision of 50cm for each axis, or better.

In practice, four different beacons are used: three of them are "Anchors", which are fixed and for which the position in space is known. The fourth one, called "Tag", is placed on the drone. With the trilateration algorithm, determining the position of a target with a minimum of three reference points in 3D, the position of the "Tag" can be determined.

In the end, the system had a precision of 18.7 cm on the X and Y axes and 39.6 cm on the Z axis, which completely satisfies the requirements.

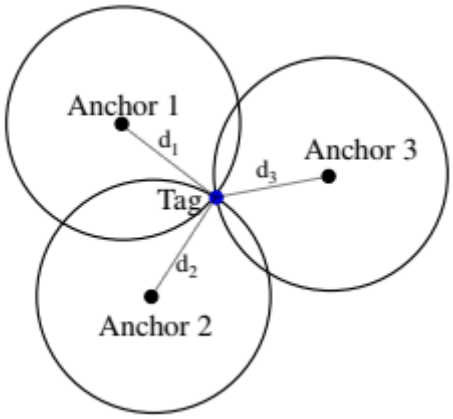


Figure 1.5: UWB positioning system, using 3 fixed "Anchors" and one "Tag" placed on the drone. Using trilateration, the position of the Tag can be calculated.

Total station system

Our master's thesis is in a way between the two above. The positioning system implemented makes use of a robotic total station (Figure 1.6). In practice, the station can detect a 360° prism (which acts as a reflector), lock itself on it and follow it. We make use of this functionality by placing the prism on the drone ; the station can thus track it and give its position in a local

coordinate system, set up beforehand.

Performance-wise, as will be explained in chapter 2, it was not relevant to evaluate the precision of this system, as modern total stations are millimeter accurate. The tracking speed of the RTS used is however evaluated and analyzed.

In conclusion, the three systems are complementary. The UWB system allows faster movements for the UAV, the positioning with the RTS can take over flight control when approaching more precise areas or blocks and the computer vision can finally take over the control to accurately pick up a block or place one. Furthermore, in case of failure of one system while the drone is flying, another one can still take over the control. An overview of the project (where only the camera for the computer vision is missing) is depicted in figure 1.7.

Figure 1.6: Leica Total Station TCRP1203[25] and a Leica GRZ4 360° Prism[26].

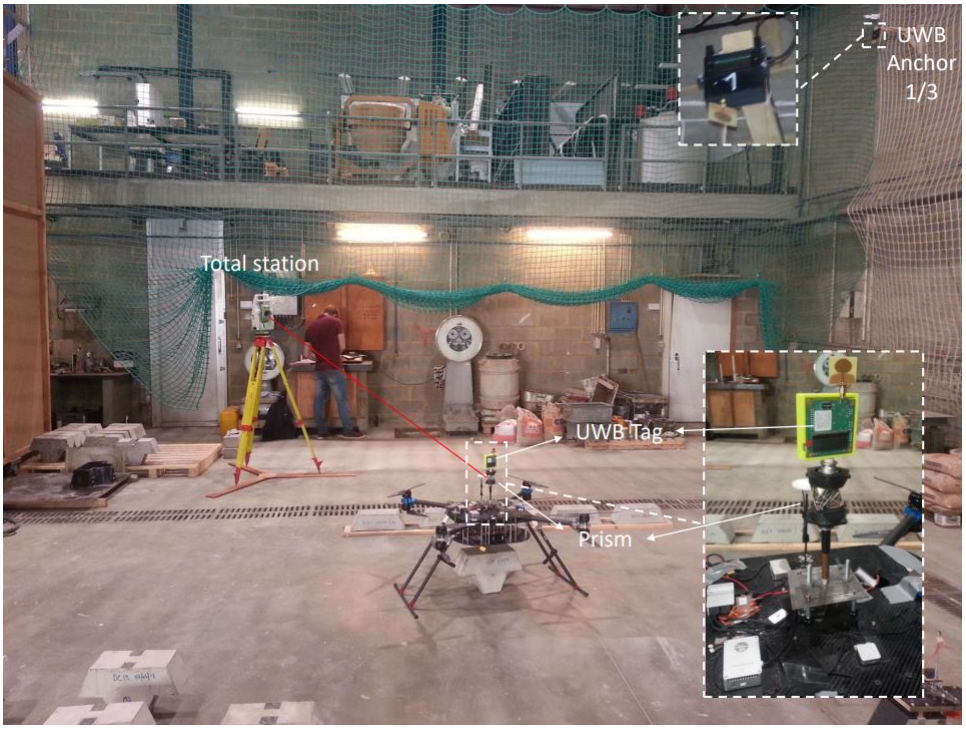


Figure 1.7: Overview of the complete project (except the camera) with : the total station on the left (pointing at the prism), a zoom on one of the three UWB anchors at the top right corner, a zoom on the prism + the tag mounted on the drone at the bottom left corner and finally the "middle-sized" testing drone at the bottom center of the picture.

1.2.2 Guidance with ALX Systems' system

At the beginning of the project, it was agreed with Pr. Latteur and T.A. S. Goessens that we had to focus on finishing the development of our positioning systems before actually working on a guidance system. In order to still be able to try a real-scale test with blocks, they decided to collaborate with ALX Systems, a small Belgian startup located in Liège[7]. They provide UAV solutions for many practical applications such as surveillance, mapping or power line monitoring. Their first main product is an operating system, ALX Operating System, mostly managing the control of the UAV. The second one is ALX Vision, an advanced computer vision program for UAVs.

Concretely, they helped us and provided a guidance system, using their proprietary Operating System. Apart from their program, we benefited from their expertise in UAVs in general.

1.3 Objectives

One global objective sums up all the work done for this thesis. Because of the construction elements we worked with, allowing an inaccuracy of 5cm when they are placed, our positioning system had to be almost centimeter accurate. More specifically, the difference between the real position of the drone and the position given by our system had to be below 5cm.

In order to achieve this global goal, the thesis was divided in a set of smaller objectives. Checkmarked ones were successfully accomplished, the others are still left to be done :

- ✓ Connect the RTS (TCRP1203) to a PC, with a USB cable;
- ✓ Understand the basics of the GeoCOM[3] protocol, used to interact programmatically with the RTS;
- ✓ Understand the Python implementation of GeoCOM by A. Maxim on GitHub;
- ✓ Use the Python script of A. Maxim to get a coordinate measurement;
- ✓ Clean the script: improve error handling and comment the code clearly;
- ✓ Implement an automated prism search in the script;
- ✓ Rebuild the test drone to fit our needs;
- ✓ Integrate the script in the guidance system of ALX;
- ✓ Perform a successful test of the RTS positioning system in ALX Systems' simulator;
- (To do) Perform an automated "hover flight" with the system of ALX controlling the test drone.

For the global project of building structures with drones, involving the three positioning systems, another set of objectives were defined:

- ✓ Configure the test drone hardware, to allow the installation of the three systems;

[3]The GeoCOM protocol is explained in section 2.2.2.

- ✓ Integrate the three systems (RTS, UWB and Computer Vision) on the test drone;
- (To Do) Perform an automated "hover flight" with the system of ALX and the three positioning systems working together ;
- (To Do) Perform a predefined circuit with the drone ;
- (To Do) Automatically pick up a drick, lift it then drop it ;
- (To Do) Perform at real-scale the target scenario described in chapter 4 .

1.4 Resources

1.4.1 Workspaces

DroneZone

A dedicated zone for UAV flights has been installed in the building of the Laboratoire Essais mécaniques, Structures et génie civil (LEMSC). It is a flight space of 700m^3, secured and delimited by safety nets. In addition to the convenience of this area for flight tests, it also was a large zone to develop the positioning with the RTS, as developing such a system in an office would have been complicated.

LEMSC electronics laboratory

This laboratory was available to perform any hardware-related work such as welding and adapting cables for the electronics on the drone.

1.4.2 Software

The main software resource for this thesis was the script of A. Maxim on GitHub, a very useful development base. The GeoCOM reference manual [19], though not concrete software as itself, has also been essential for the project. Finally, ALX Operating System, the UAV control system provided by the company, was essential for autonomous flights of the test drone.

Many other software resources were used but, because they are less important and it would be cumbersome to mention them all here, they are only described in the README file of the project, in appendix C.

1.4.3 Hardware

See appendix D for the full list of the material used for this master's thesis. In addition to the hardware material for the drone, 3D designs for 3D printed objects, used to improve the UAV, are provided. All designs produced are present in the directory of the project and can be seen in appendix E.

1.5 Related work

In this section, for the papers studying a whole guidance system, the focus here is on the positioning system. This master's thesis indeed focuses on a positioning system, coupled with the guidance system of ALX Systems to fly the drone.

1.5.1 ETH Zurich (2011)

ETH Zurich seems to be a pioneer in projects involving construction processes using UAVs and blocks, with its global project "Aerial Construction"[4]. It was in particular launched with a real-scale experiment in 2011, where four autonomous quadrocopters built a 6m-tall tower with 1500 foam modules (figure 1.8). The corresponding paper was published in 2014 by Augugliaro et al. [11]. Other articles on similar projects were published, in particular investing the building of tensile structures with UAVs [12, 13]. The most recent one, published in 2015, studied knot-tying with UAVs for aerial construction [14].

For the 2011 experiment, a platform called the ETH Flying Machine Arena (FMA) was used. In development at ETH Zurich since 2008, it is a "research and demonstration platform for fleets of small quadrocopters"[11]. The FMA uses an overhead commercial motion capture system, more precisely a 19-camera Vicon T-40 system, which can track and record the positions of marked objects. All the objects used in the experiment (e.g. the four quadrocopters) were marked using retroreflective tape.

In order to guide the UAVs, a ground station (simple PC) first receives the motion capture system's measurements and the desired trajectory for the current UAV. It processes these data then sends the result via a radio link to the UAV. The onboard computer processes this information and controls the motors accordingly.

In terms of performance, the whole guidance system is very accurate : 98.27% of foam modules placements were under 25mm of lateral placement error, 91.2% were both under 2.5cm of lateral displacement and under 2° of orientation error.

In conclusion, for the positioning system :
Advantages:

- Very accurate (< 2.5cm of placement error laterally);
- Many vehicles can work at the same time;
- Reliable system, as long as the cameras have a good overview of the construction site.

Drawbacks:

- Very expensive. The price of the newest Vicon cameras for motion capture starts at $3500;
- Motion capture is not an ideal choice on real-scale construction sites situations : it would be too complicated to set up and to use;

[4] `http://www.idsc.ethz.ch/research-dandrea/research-projects/aerial-construction.html`

- Simpler alternatives for positioning exist.

Figure 1.8: 6m-tall tower composed of 1500 foam blocks, built by four autonomous quadrocopters during a real-scale experiment in France in 2011 [11].

1.5.2 Latteur et al. (2015, 2016) [1, 2, 4]

A first paper, published in 2015 as conference proceedings from the IASS (International Association for Shell and Spatial Structures), introduced the idea of using UAVs to perform "additive manufacturing of architectural structures". It investigates the feasibility of the whole process, (described in four steps in section 1.2), from the design of the building to its construction with UAVs. In a dedicated section, the authors study many positioning systems, mentioning the possibility to work with a theodolite station, which is what is developed in this master's thesis.

The second paper on the subject was published in 2016, still in the context of the IASS, but this time with a smaller team composed only by P. Latteur (UCL), S. Goessens (UCL) and C.T. Mueller (MIT). The document mainly shows the progress made towards the realization of the idea. First, it studied more in depth specific block designs, namely the drick60 and drick60i, the most promising ones in terms of "drone-compatibility". Then, real-life tests were performed

and even software simulations of structures built with those blocks. Also, successful tests with a custom-made quadcopter were performed, with the construction of a column made of cylindrical concrete blocks [4].

1.5.3 Nehri N. and Paques A. (2016) [8]

This paper is about a master's thesis which implements the idea of using, as a positioning system, fixed lasers on the drone which all point to reference walls, one for each axis in a 3D coordinate system (figure 1.9). The authors are two ECAM students who were under the supervision of P. Latteur and S. Goessens. They were the first team to work on an automated positioning system for the whole project of masonry work with drones.

In terms of results, the positioning and guidance systems were tested separately. The positioning system was tested by taking 9000 position measurements while the testing UAV was just placed on the floor, not moving. A standard deviation of 0.82cm was computed for the X axis [5].

For the guidance system, 12 000 position measurements were taken, while the drone was only hovering and the controller was sending a fixed position to maintain. They computed a standard deviation of 2.8cm between all those measures, which is satisfactory for posing blocks which allow the aforementioned tolerance of 5cm.

In the end, concerning the positioning system:

Advantages:

- It is pretty accurate ($<$ 1cm std. dev.);
- It is not subject to electromagnetic perturbations, or to perturbations of light and pressure;
- It works on most surfaces (except glass, for example) and doesn't depend on the condition of the reference walls;
- Measurements are fast (100 per second).

Drawbacks:

- The maximal range of the lasers is high (until 3km), but the current system with the reference walls is not compatible with a high distance;
- It is actually limited to an indoor use;
- It is very dependant on the yaw angle (rotation angle around the Z axis in a 3D Cartesian coordinate system);
- The cost of the lasers can become high ($>$ 3000 euros) if high-end lasers are used.

[5] The value was the same for the three X, Y and Z axes, as the drone seemed to have a similar positioning precision no matter the axis.

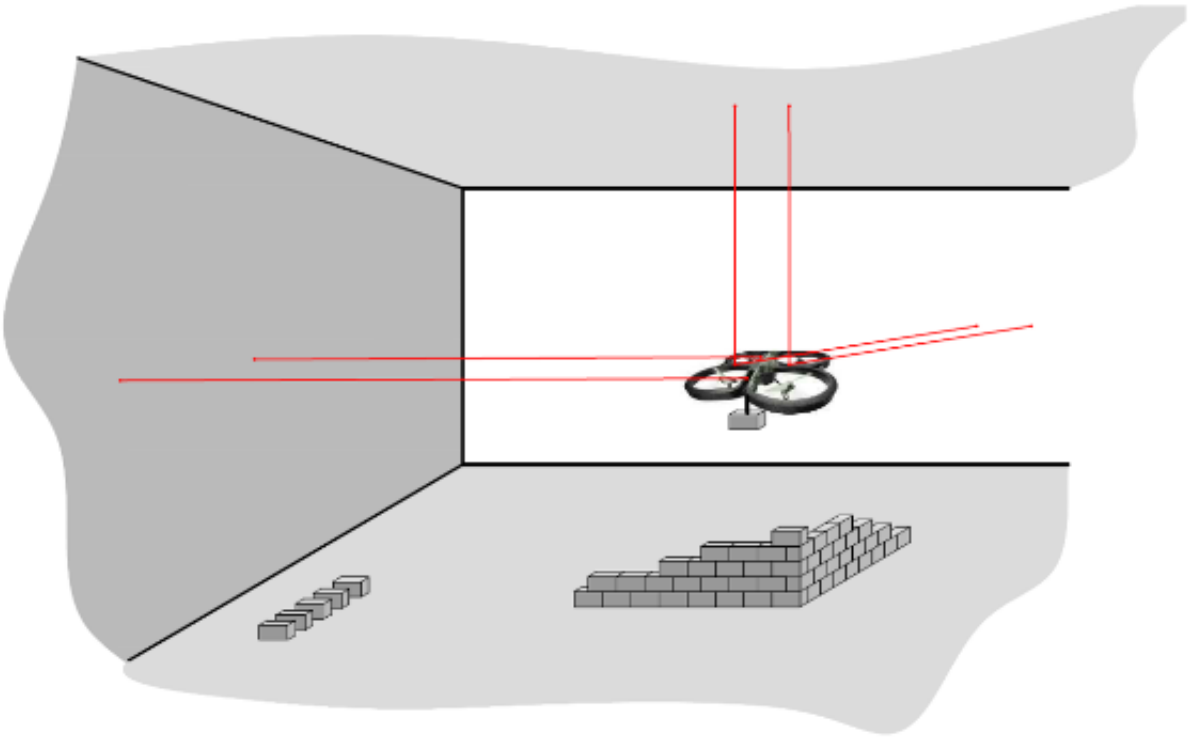


Figure 1.9: Positioning system using lasers pointing to 3 reference walls : one for each axis in a 3D Cartesian coordinate system [8].

1.5.4 Maxim et al. (2017) [9]

This conference paper introduces some work from the complete master's thesis of Artyom Maxim [10], from the University of Stuttgart. The project investigated the use of UAVs for constructing structures with fiber composites, by applying a process called filament winding. As a positioning system for the UAVs, a robotic total station (Leica MS60) has been used, as for this thesis. It was indeed used to track a prism set up on the drone, in order to know its position. As it will be explained in details in chapter 2, we even partly reused some code from this project, made freely available on Github by Artyom Maxim[6], himself reusing some code from Georg Wiedebach [7]. In the end, both our positioning systems are similar, real differences only appear in the technique and in the code used to record the position of the prism. Only the guidance system was completely different.

In order to evaluate the guidance system, tests were performed on a full-scale filament winding experiment. The metric used was the shortest distance (in 3D) between a measured position at a certain timestamp i and the reference path. On a total of 1405 distances measured, a root mean square (RMS) of 32.9cm between all the distances has been computed. However, for the x-y plane only, they achieved an RMS of 10.4cm, while the value for the y-z plane was 31.9cm. This shows that the horizontal precision is better, which is the most interesting type of precision for placing blocks. However, such a precision of is not high enough in the context of construction with blocks.

[6] `https://github.com/art-mx/leica_ros_sph`
[7] `https://github.com/georgwi/leica_ros_sph`

Figure 1.10: Fiber composite structure built by an UAV, guided by a system using a total station (Leica MS60) to position it spatially[9].

The positioning system has the same advantages and disadvantages as the one of this master's thesis, because of the use of a total station. There are of course technical differences between the MS60 and TCRP1203 RTS, but the following comments are relevant for both.

Advantages

- Millimeter measurement precision;
- Long measurement distance : up to 2km with the GRZ4 360° Prism, in optimal conditions [16];
- Very robust (meant to be used outdoors);
- Multiple stations can be used for a same drone, which would increase the reliability of the positioning system.

Drawbacks

- The prism on the drone must stay in direct sight of the station, hence the drone cannot fly behind an object and cannot fly too fast, especially at a short distance;
- A Leica MS60 costs around 48000 euros ; even a TCRP1203 costs 6300.

Chapter 2

Positioning system

2.1 Automatic prism location with the total station

The goal of our positioning system is to get the precise position of the drone in real time. For that purpose, we decided to use a Total Station (TS) with a 360° prism compatible with it. We use a prism as it can be tracked with the TS and it is easy to put it on a drone. We also use a ground station computer to control and get the data provided by the TS, then to send it to the flight controller program.

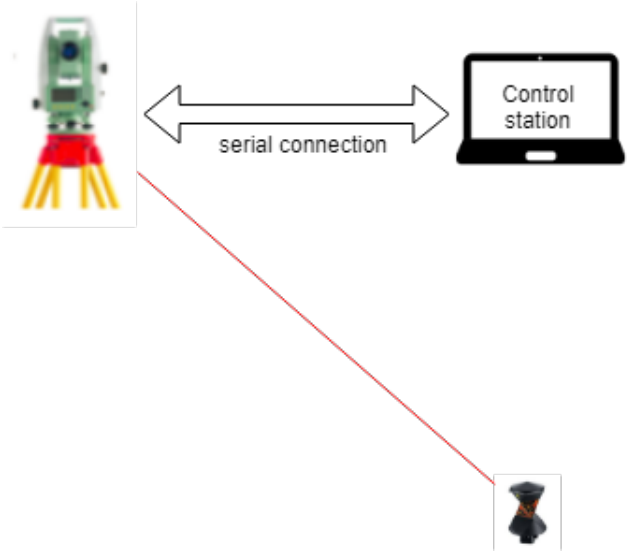


Figure 2.1: Architecture diagram of the positioning system

2.1.1 Total station

A total station (TS) or total station theodolite (TST) is an electronic/optical instrument used for surveying and building construction. The total station is an electronic theodolite integrated with an electronic distance measurement (EDM) unit to read distances from the instrument to a particular point (in our case, a reflector), and an on-board computer to collect data and compute coordinates based on the configuration of the station.

Robotic total stations (RTS) are electronic theodolites that can be remote-controlled from a distance via a remote or by using a computer software. In our case, the goal is to develop a system that automatically configures the RTS to find and lock the reflector and continuously send the data to the UAV's onboard computer. For this purpose, we needed to buy a specific license from Leica for each station used, to be able to use the robotic capabilities programmatically.

TCRP1203 Total Station

The first RTS we used is a Leica TCRP1203. The station is a bit old, lacks usability and is not very user friendly, but since it's the only RTS which was always available for the whole duration of the thesis, we mainly worked with that one.

It is capable to take accurate measurements with a reflector at a maximal range of 1000 meters (with a 360° "big" prism), with an accuracy of 5 mm with a standard prism. It is able to track a prism going at a maximal tangential speed of 5 m/s at 20 meters (up to 25 m/s at 100 meters). Measurements take less than 0.15 seconds to be performed [16].

Figure 2.2: Leica TCRP1203

MS50-MS60 Total Stations

The second RTS we tested is the MS50. This test was just to ensure the script is still compatible with a more modern RTS from Leica, which was fortunately the case. After that, we borrowed for two weeks an MS60 to Leica Belgium with the help of Mr. Güth (Sales engineer for Leica Geosystems Benelux). Apart from again testing the compatibility of the script, we were able to test a bit further the performance (especially the tracking performance) of such a modern RTS.

Both stations, released in 2013 (MS50) and 2015 (MS60), are way more responsive, both at the software and at the hardware level. In comparison to the TCRP1203, the maximal tangential speed at 20 meters is 9 m/s, up to 45 m/s at 100 meters. With a prism, the maximal range is 3 kilometers [18].

Figure 2.3: Leica MS60 RTS

2.1.2 Prism

Our system uses a 360° prism which acts as a reflector, placed on the drone. In practice, the RTS' optics emit a laser beam, that the prism can reflect. It then detects the reflected signal, thus detecting the prism. To actually know the distance between the latter and itself, the RTS simply computes the time that the signal took to go to the prism and come back. From this measured time and the value of the speed of light, the distance is easily computable. This principle is widely used for surveying applications.

Circular prisms are actually more accurate [20], but they always need to be oriented to the RTS to allow measurements, which is not practical at all to perform automated tasks. Furthermore, the differences in accuracy between 360° prisms and circular ones (2 mm max) is not relevant in the context of this master's thesis.

For most of the development of the project, we used a 360° mini-prism (figure 2.4). Apart from its reduced weight, the main difference with a 360° "big" prism (figure 2.5) is the possible range to which it can be detected. A related consequence is that the RTS loses the mini-prism from sight more easily, but this fact is only relevant at very short (< 5m) or high distances ($>$ 300m, the theoretical range limit for the RTS to detect and lock the mini-prism), which are very unlikely (or at least easily avoidable) in real-scale construction tests with a UAV.

Figure 2.4: Sokkia ATP1 360° mini-prism

Figure 2.5: Leica GZR4 360° "big" prism

2.1.3 RTS robotization capabilities

There are 3 different *Automation* modes available on the TCRP1203 [15] and the MS60 [17]:

- None : no target search nor target tracking
- ATR (Automatic Target Recognition) : the RTS can automatically search for a target prism
- LOCK : same as ATR mode, but the RTS can track the target prism once it found it

The total station is thus always configured in LOCK mode in our case.

To continuously know the position of the prism (and thus of the drone), we need to perform three successive tasks with the total station:

1. Launch a search for the prism and find it
2. Lock the station on it
3. Track it and record its position in real-time

Possibly go back to step 1 if the prism is lost from sight (the precise process is described in section 2.2.3).

These functionalities have been automated, in a way which will be detailed in the next section.

2.2 Automation of the positioning system

2.2.1 Interaction between a computer and the total station

The first step is to connect the theodolite to a computer. The total stations can be connected via USB cable and even to a WLAN for the MS50-60. We did not try to connect wirelessly the station to a PC, as the physical cable is the most simple and reliable way. Furthermore, the MS60 was often not available, most of the development of the thesis has been done with the TCRP1203. Leica produces a wide range of proprietary cables, each one made for a specific series of total stations. In our case :

- For the TCRP1203 (TPS1200 series) : GEV267
- For the MS50-60 : GEV269

Once the connection is made, the link created is a simple serial connection.

2.2.2 GeoCOM protocol

GeoCOM, an interface to the sensor functions of most of Leica RTS[1], has been developed by the constructor [19]. It was meant to enhance existing functionalities or even add some custom ones.

For communications, GeoCOM follows the "client-server" model (figure 2.6) : the client sends requests, which the server handles and to which it sends a corresponding response. In practice, the theodolite instrument is considered as the server while the program interacting with it is the client.

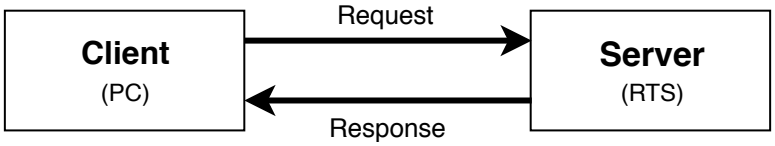


Figure 2.6: GeoCOM is implemented as a simple point-to-point system with a client and a server.

Concretely, GeoCOM is implemented in a simple ASCII protocol, in C/C++ and in VBA. In ASCII, the requests and responses are thus encoded as simple strings of text, which means the protocol can be used on any operating system or platform. To avoid rewriting the whole protocol, we used an existing Python implementation of most predefined ASCII functions of the protocol, made available on GitHub by Artyom Maxim [2], who had already forked the project from Georg Wiedebach [3]. Even if another implementation of GeoCOM by Leica exists in C/C++, Python was chosen. We were indeed way more comfortable with this programming language and it has useful scripting abilities that C/C++ do not have.

The implementation of GeoCOM for this master's thesis is publicly available on GitHub [4] in the folder `script/src/`, while the corresponding documentation can be found both in the folder `scripts/documentation/` on the GitHub repository and in the appendices.

ASCII request

Each ASCII request has the following format [19] :

```
[<LF>]%R1Q,<RPC>[,<TrId>]:[<P0>][,<P1>,...]<Term>
```

Optional items are in brackets []. The angled-brackets <> surround names or descriptions.

[1] Including the ones we used, namely the TCRP1203 and both the MS50-60.
[2] `https://github.com/art-mx/leica_ros_sph`
[3] `https://github.com/georgwi/leica_ros_sph`
[4] `https://github.com/rvermeiren/Leica-GeoCOM-for-drone-tracking`

Table 2.1: ASCII request

Flag	Description
<LF>	An initial line feed ("\n") clears the receiver buffer. This is used ensure that the request is directly treated when it is received.
%R1Q	GeoCOM request type 1.
<RPC>	Remote Procedure Call identification number between 0 and 65535. Identifies the function call.
<TrId>	Optional transaction ID: normally incremented from 1 to 7. Same value in reply.
:	Separator between protocol header and following parameters.
<P0>,<P1>,...	Parameter 0, Parameter 1, ...
<Term>	Terminator string (default CR/LF, use COM_SetTerminator to change the terminator). In our implementation, a terminator string is added implicitly.

ASCII response

Each ASCII response has the following format :

```
%R1P,<RC_COM>[,<TrId>]:<RC>[,<P0>,<P1>, ...]<Term>
```

Optional items are in brackets []. The angled-brackets <> surround names or descriptions.

Table 2.2: ASCII response

Flag	Description
%R1Q	GeoCOM request type 1.
<GRC>	GeoCOM return code. This value denotes the success of the communication. GRC = O means the communication was successful.
<TrId>	Transaction ID - identical to that of the request. If the request had no Transaction Id, then it will be 0.
:	Separator between protocol header and following parameters.
<P0>,<P1>,...	Parameter 0, Parameter 1, ... These parameters will be valid only if GRC is equal to 0.
<Term>	Termintator string (default CR/LF, use COM_SetTerminator to change the terminator). In our implementation, a terminator string is added implicitly.

Organization of the RTS system software

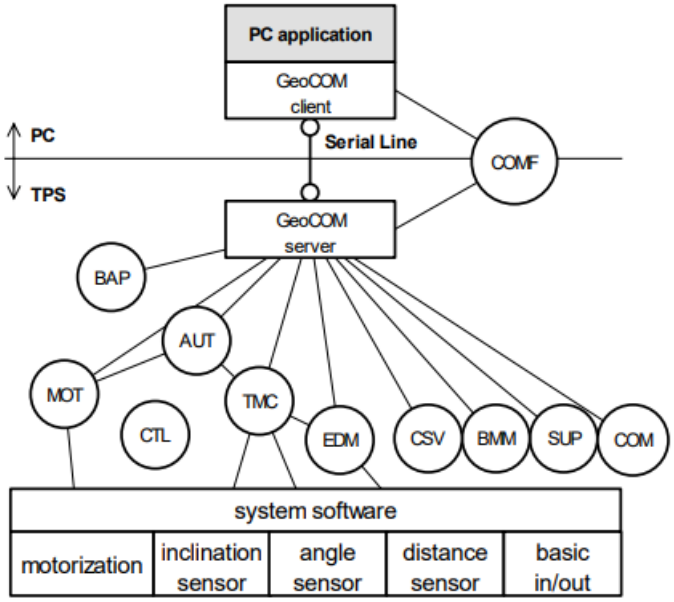


Figure 2.7: Overview of the Client/Server GeoCOM application [19]

Figure 2.7 sums up the overall organization of GeoCOM functions, depending on which sensor or subsystem of the RTS. Each function name indeed starts with a prefix to identify which kind of function is called and on which sensor they act on the total station. The complete description of each of them is explained in the GeoCOM reference manual [19].

AUS The subsystem 'Alt User' mainly contains functions behind the "SHIFT" + "USER" button.
AUT Automatisation; a module which provides functions like the control of the Automatic Target Recognition, Change Face function or Positioning functions.
BAP Basic Applications; some functions, which can easily be used to get measuring data.
BMM Basic Man Machine; functions which controls some basic input/output functionality, e.g. set beep alarm, etc.
COMF Communication; a module, which handles the basic communication parameters. Most of these functions relate to both client and server side.
COM Communication; functions to access some aspects of TPS1200 control, which are close to communication. These functions relate either to the client side or to the server side.
CSV Central Services; this module provides functions to get or set central/basic information about the TPS1200 instrument.
CTL Control task; this module contains functions of the system control task.
EDM Electronic Distance Meter; the module, which measures distances.
MOT Motorization; the part, which can be used to control the movement and the speed of movements of the instrument.
SUP Supervisor; functions to control some of the general values of the TPS1200 instrument.
TMC Theodolite Measurement and Calculation; the core module for getting measurement data

GeoCOM Implementation in Python 2.7

Our implementation of GeoCOM is based on the work of Artyom Maxim[5], who refined a bit the work of Georg Wiedebach[6]. The code, as it was when the repository of A. Maxim was first consulted (October 2017), was functional but lacked good programming practices, which made it weak and thus more likely to crash if an unexpected error occurred. First of all, most possible communication errors were not handled. We also noticed, after a lot of documentation reading, that the return codes of most GeoCOM functions called were not even checked. Finally, the whole script (around 150 lines long) was written all at once and not divided at all into subroutines. As the positioning system developed in this thesis requires a reliable program and that the original one was not safe to use as it was, we made the script less error-prone : it was well divided into functions and all errors are correctly handled. Moreover, many GeoCOM functions not originally implemented were added, as well as a full documentation (following the PEP287 [7] standard).

2.2.3 Position and track the prism

Figure 2.8 sums up all the tasks performed by the script for the positioning system, from the coordinate system set up to the tracking of the prism.

[5] `https://github.com/art-mx/leica_ros_sph`
[6] `https://github.com/georgwi/leica_ros_sph`
[7] `https://www.python.org/dev/peps/pep-0287/`

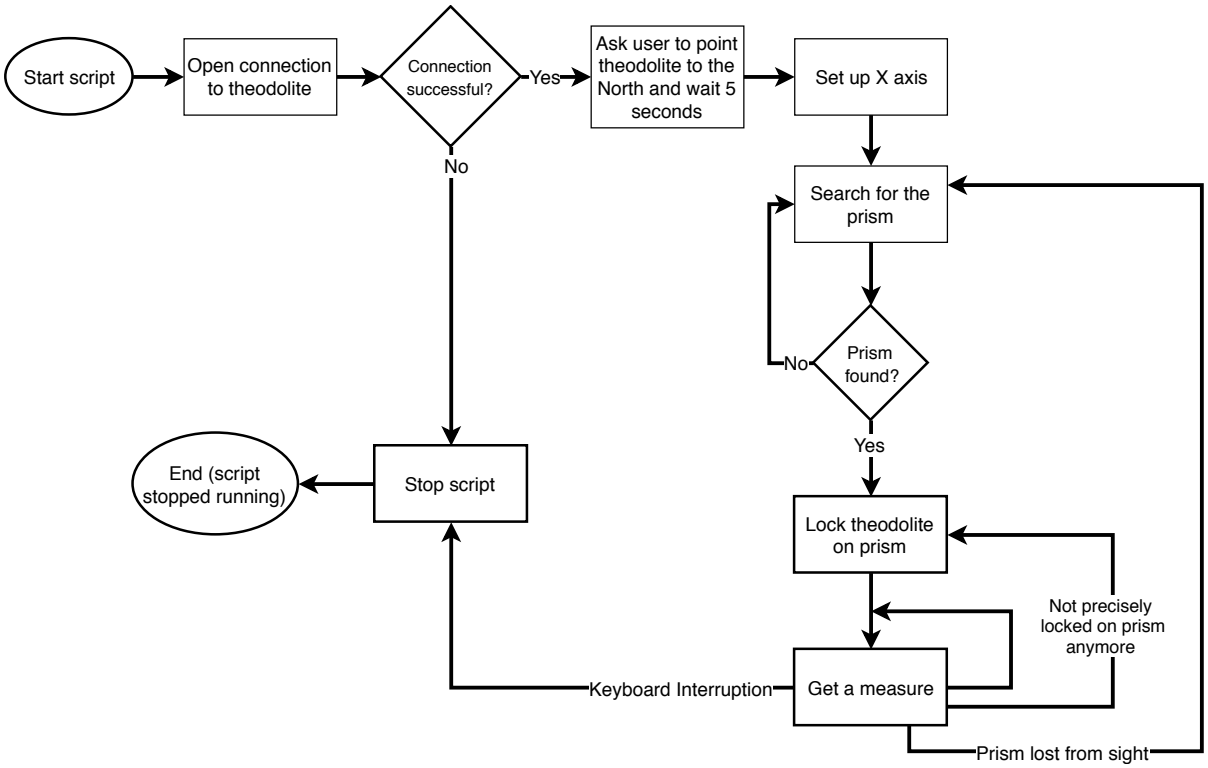


Figure 2.8: Flowchart of the main script which continuously measures the position of the prism.

Setting up the positioning system

First, the script of the system accepts 4 options:

1. `-p` (port) : the port on which the serial connection will be set up. Set to "COM3" by default;
2. `-b` (baudrate) : the baud rate[8] set up for the communication on the serial connection;
3. `-d` (debug) : if `-d` is given, the debug is set to true, meaning some function calls details are printed on the standard output; the default is false;
4. `-B` (Big) : if `-B` is given, the prism type is set to "big prism"; the default is "small 360 prism".

As described in figure 2.8, the script starts by opening a connection with the RTS, on port "COM3" and with a baud rate of 57600, by default. The baud rate should be the same as the one set in the RTS to avoid errors or unexpected behaviour. Once the connection is set up, the user is asked to direct the lens of the station to the north. This allows to set up the direction of the X axis on it (why the north is used is explained in chapter 3). After that, the RTS is considered to be at coordinate (0, 0, 0). The prism type is then set to "big 360 prism" if explicitly given in argument, or "small 360 prism" by default. Finally, a search is launched. This process is explained in the next section.

Once the prism is found and locked, the EDM mode of the station is set to "fast repeated measurement", that allows to take continuous measurements.

[8]From Wikipedia [21]: "Common measure of the speed of communication over a data channel".

Searching for the prism

The search is a crucial part of the script as it's the one that finds the prism in the first place and allows to find it back if it is lost from sight. Two searching approaches were developed, both are available in our code.

The first one is based on the `AUT_Search` function from GeoCOM and is available with all GeoCOM compatible RTS. The function searches for the prism in a given area, in the way depicted in figure 2.10 on the left, the "Classic search". To define the search area, the function takes as argument a vertical and horizontal angle. By default, both are assigned to 20°. Those angles define the maximal angle that the RTS will cover in one direction, starting from its initial position. If, for example, 20° as vertical angle is given, the height of the search area will be the height that the RTS can cover by moving by 20°, either up or down (see figure 2.9).

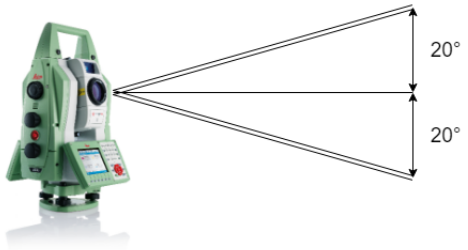


Figure 2.9: "Classic search" vertical search window, with a vertical angle of 20° specified as argument.

For the search to work correctly and if a mini-prism is used, the prism can be at most 350m away from the RTS. The big advantage of this solution is that if the total station is already directed on the prism, the search directly finishes. The drawback of this search is that it is very slow, thus not very efficient if it must find a moving object.

The second approach is to use a function designed to find the prism faster, `AUT_PS_SearchWindow`. Instead of using the main lens of the telescope, the theodolite uses another sensor, dedicated to speed up the search. The name of this functionality is "PowerSearch" (PS). It requires more parameters to be set to work:

1. The "center" of the searching area, which is the orientation in which the RTS starts the search, given as an horizontal and a vertical angle;
2. The searching area, where again both horizontal and vertical angles are required;
3. The distance interval inside of which the RTS can search for the prism.

With all this information set (specific GeoCOM functions exist to set all parameters), the search can begin. The search path is depicted in figure 2.10 on the right, the "Power Search". As can be noticed, the field of view of the dedicated sensor is much larger than the lens used for the classic search. However, on the vertical paths labeled as "move" on figure 2.10, the RTS doesn't actually search for the prism but only moves to the next horizontal "line of sight" to continue the search.

For our application, the range interval is fixed to $[5, 100]$ meters but it can be easily expanded. An important advantage of this approach is the average time to find the prism is really low,

especially compared to the other method. The only drawback is that the search starts at the top left corner of the search area: if the station was already directed on the prism, the RTS does not detect it and begins the search anyway.

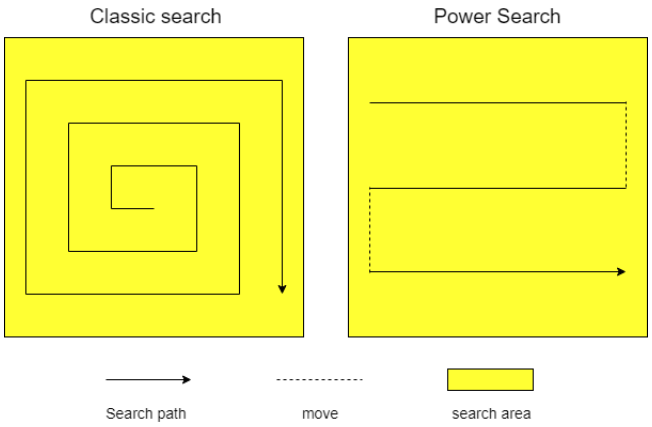


Figure 2.10: Classic search versus PowerSearch

Tracking the prism and measuring its position

The tracking is maintained all the time, because the LOCK automation mode is set on the RTS.

When the script is running, it cannot be guaranteed that the RTS never loses the prism from its field of view. When a measurement request is performed, different codes are returned in the response, depending on the situation:

- If a fully accurate and successful coordinate measurement was performed, 0 is returned;
- If an "inaccurate" (i.e. the measurement was not verified by the internal system of the RTS), 1284 is returned;
- If another code is returned, it means the measurement request failed.

A search is actually launched after 100 failed coordinate measurements in a row (approximately 10 seconds of failed measurements). It can happen for example if the prism is completely lost from sight for a moment and that only the angles of the RTS could be measured.

Concerning the quality of a measurement, even an "inaccurate" measurement can be considered as acceptable by the positioning system. Indeed, it is only considered as such by the RTS when the lens is not aiming at the center of the prism anymore but slightly nearby (the prism is still tracked in this case), hence only guaranteeing centimeter precision and not millimeter precision. This imprecision is not problematic though, since the objective of the thesis is to have a precision lower than five centimeters.

The kind of measurement request we decided to use[9] does not request an (X,Y,Z) coordinate directly on the RTS, but instead only asks for both angles in which the RTS' lens currently is and the distance to the prism. This choice was made to keep the station configuration as simple as possible but also to avoid performance issues while tracking and measuring, because requesting a

[9]Function `TMC_GetSimpleMeas`.

full coordinate from the RTS is slower. The conversion of the angles and distance to Cartesian coordinates is explained in the next section (section 2.3).

2.3 Coordinate system

When the RTS performs a measurement, we get both the horizontal and vertical angles of the lens (θ and ϕ on figure 2.11, respectively) and the distance between the station and the prism (ρ). These are spherical coordinates but we wanted to work with Cartesian coordinates. Here is how to compute them for each axis:

- $\mathrm{x} = \rho \sin\phi \cos\theta$
- $\mathrm{y} = \rho \sin\phi \sin\theta$
- $\mathrm{z} = \rho \sin\phi$

The point we are interested in (P in figure 2.11) has thus the coordinates $(\rho \sin\phi \cos\theta, \rho \sin\phi \sin\theta, \rho \sin\phi)$.

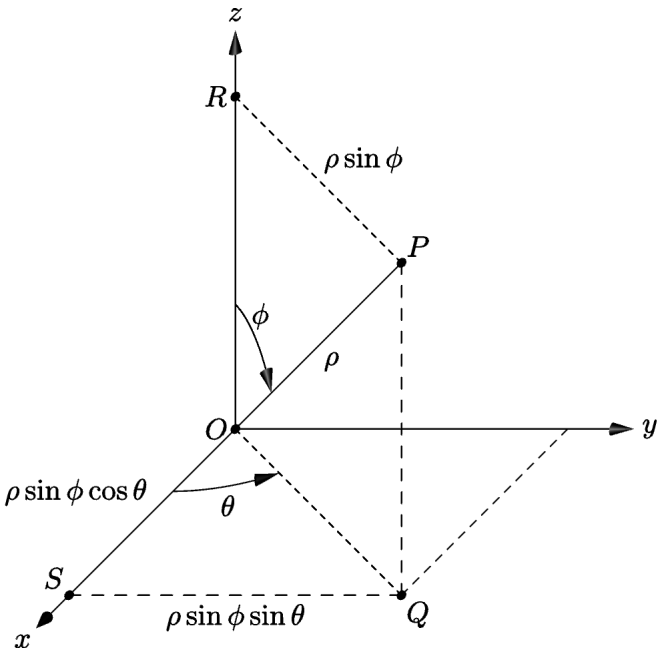


Figure 2.11: Graph highlighting how to compute Cartesian coordinates from spherical coordinates, in a three-dimensional coordinate system [27].

2.4 TCRP1203 performance evaluation

2.4.1 Measurements

According to the technical data on the TCRP1203 [16], the angle measurements standard deviation (horizontally and vertically) is 0.3 mgon (0.00027 degrees).

The standard deviation of distance measurements depends on the measurement type. The instrument can either measure the distance with a visible laser beam (reflectorless) or with a reflector. On Leica total stations, this is called the EDM (Eletronic Distance Measurement)

type. The one used in combination with a prism is the IR type. There are also four EDM modes available : each of theses modes differ in the time the instrument takes to perform one measurement. The accuracy for each mode is depicted in table 2.3.

Since the devices are known to be very accurate and because Leica claims such low measurement deviations, it seemed unnecessary to review their accuracy here. Furthermore, it would be complicated to develop a method with enough precision to serve as a baseline to compare it to the total station.

Table 2.3: Standard deviations of distance measurements for the TCRP1203, using a prism (no matter which type) and depending on the EDM mode currently set on the total station. "Averaging" mode has no delay to measure, because it simply allows to perform repeated measurements in "Standard" mode [15].

EDM mode	**Distance measurement std. dev. with prism**	**Measurement time (s)**
Standard	2mm + 2 ppm	2.4
Fast	5mm + 2 ppm	0.8
Tracking	5mm + 2 ppm	< 0.15
Averaging	2mm + 2 ppm	-

2.4.2 Tracking speed

Testing procedure and conditions

The TCRP1203 is theoretically capable of following, in LOCK automation mode, a prism moving at a maximal tangential speed of 5m/s at 20m (the maximal range in LOCK mode is 300m with a mini-prism). Concrete tests were performed, to evaluate its performance at different speeds.

At a certain distance from the RTS, for one minute, a student was going back and forth on a circular path of around 20m long, holding the prism, while the RTS was recording its position every tenth of a second (figure 2.12 shows an example). For every new coordinate measurement, the testing script was computing the speed in m/s by using the previous coordinate measured. To compute the speed v_t of the prism between a point P_t at the coordinate (x_t, y_t, z_t) recorded at time t, and a point P_{t-1} at the coordinate $(x_{t-1}, y_{t-1}, z_{t-1})$ recorded at time $t-1$:

$$v_t = \frac{dist(P_t, P_{t-1})}{0.1}$$

Where:

$$dist(P_t, P_{t-1}) = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2}$$

Such a test was performed for three distances : 10, 20 and 30 meters. For each distance, the student was going at three different paces:

- Slow walk : $\approx$ 1 m/s (3.5 km/h)
- Fast walk : $\approx$ 1.5 m/s (5.4 km/h)
- Running : $\approx$ 2.5 m/s (9 km/h)

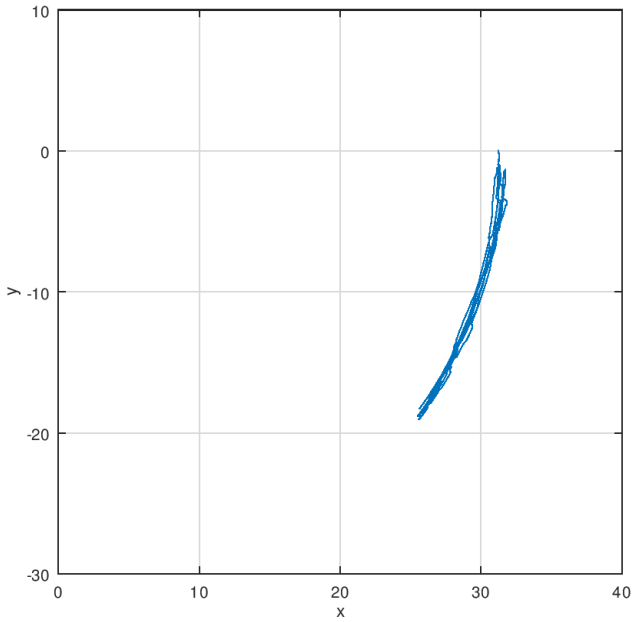


Figure 2.12: Circular path followed by the student and seen from above, for the test at 30m of the RTS and at the fastest pace (≈ 9km/h). The values on the X and Y axes are in meters.

This gives a total of 9 different testing configurations. In the "Running" phase at 10m, no significant result was obtained, as the prism was lost fom sight by the RTS almost immediately after the student began to run.

Concerning the conditions of the tests, they were performed outdoors and in backlight. This wasn't ideal, as it can have an impact on the accuracy of the RTS, but this impact is really low. In a paper investigating the accuracy of total stations [23], doing measurements in the direction of the sun increased, in their tests, the mean square error of the measurements by only a few tenths of millimeters.

As a performance metric, we considered, for a given distance and pace, the number of accurate and "inaccurate" (when the accuracy could not be guaranteed by the RTS' system) measurements. Figure 2.13 sums up the results. The result for "10_fast" is not depicted, as the prism was immediately lost at the beginning of the test. Moreover, when the RTS was recording the measurements during the tests, the time intervals between two were not exactly 0.1s, but slightly higher (between 0.1 and 0.15 seconds, according to the technical data on the TCRP1203). Hence a bit less than 600 measurements were performed in one minute of testing (around 510 in total).

The theoretical threshold for the maximal tracking speed was not tested, as it would have simply been difficult to run at 18 km/h and hold the prism steadily: unwanted vertical movements would have been too important and the results less valid. Moreover, the Leica MS50 and MS60 were not evaluated, because they were not available long enough to work with in the context of the thesis. As described in section 2.1.1, the MS60 was only available for two weeks to allow a bit of informal real-scale testing, while the MS50 was only tested to ensure the script worked on it.

Analysis of the results and discussion

In figure 2.13, one can notice that the performance at 10m is already quite bad at the slowest speed, even worse at around 1 m/s. At 20m, noticeable differences in performance appear between the speeds. When walking slowly, almost a quarter of the measurements are fully accurate. This proportion quickly decreases as the speed increases, but even at the fastest pace ($\approx$ 2.5 m/s), the prism stays in sight of the RTS. At 30m, the tendency is similar, but more clearly noticeable. This behaviour would probably be the same for higher distances, until reaching a distance where all measurements are accurate. Even though the performance metric used shows what seems to be quite bad results, in practice the less accurate coordinates measurements are still more than centimeter accurate and can perfectly be used to position a UAV. Furthermore, starting from 20m, the prism remained locked by the RTS, even at the highest speed tested.

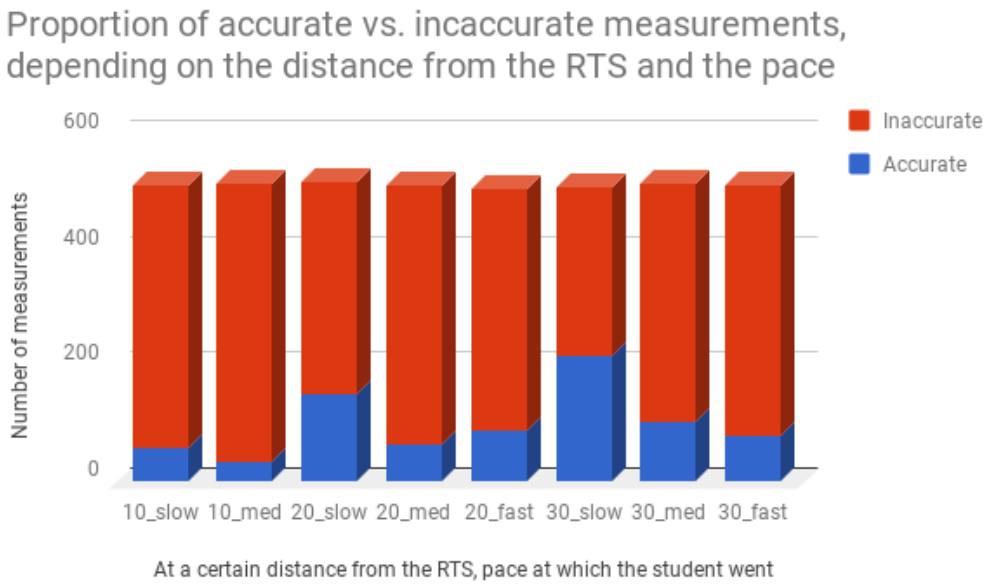


Figure 2.13: For each distance and each pace tested, depicts the proportion of accurate and inaccurate measurements. Labels on the X axis are written as <distance>_<pace> pairs, where <distance> is the distance from the RTS in meters, while <pace> is the pace at which the student was going. ["slow", "med", "fast"] correspond to ["Slow walk", "Fast walk", "Running"], respectively. The bar for "10_fast" doesn't exist, as data for this test was not available, because the prism was immediately lost at this distance and speed.

The maximal speeds reached during the tests, for each possible condition, are depicted in figure 2.14, along with the average speed, comparatively. Such high maximal values were not reached in reality: two factors can lead to them. First, during the tests, the student induced small unwanted vertical movements while holding the prism and moving at the same time on the circular path, because holding it by hand makes it hard to keep a linear trajectory. This can lead to sometimes higher speeds measured, as a greater actual distance is travelled by the prism (a greater distance than the one actually covered by the student holding it). In addition, measuring the speed was only done by recording some specific coordinates at certain times (almost every 0.1s but in practice between 0.1 and 0.15s, as explained above). Therefore, those results are not

exactly representative of the true continuous motion of the prism during the tests.

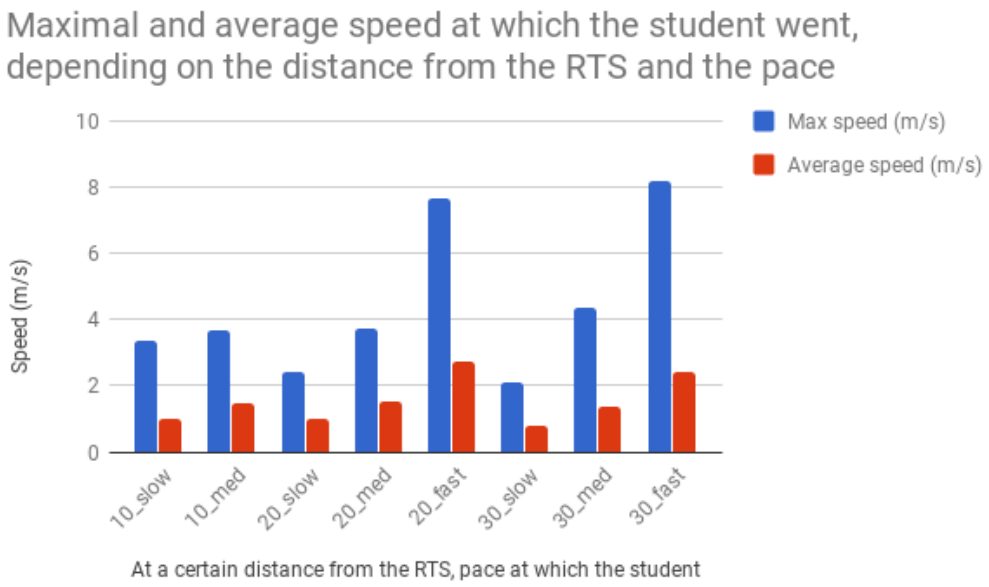


Figure 2.14: For each distance and each pace tested, depicts the maximum and average speed at which the student went during the tests. Labels on the X axis are written as <distance>_<pace> pairs, where <distance> is the distance from the RTS in meters, while <pace> is the pace at which the student was going. ["slow", "med", "fast"] correspond to ["Slow walk", "Fast walk", "Running"], respectively. The bar for "10_fast" doesn't exist, as data for this test was not available, because the prism was immediately lost at this distance and speed.

If both effects occur for a particular measurement, a significantly higher speed than the real one (the real is approximately equal to the average speed computed in each test) is measured. Such "anomalies" can be seen on figure 2.15. For the test where the prism was 30m away from the RTS and going at around 2.5 m/s, the graph shows the speeds measured and the moving average of the speeds, with a period of 2 (each point for the moving average is the average speed of the last two values). First, the points with the highest values probably are speeds measured with both unwanted effects described. Then, one can notice the "back and forth" motion of the movements performed during the tests, causing this particular shape for the graph. Finally, the small vertical unwanted movements of the prism appear as the sharp fluctuations around the average.

In conclusion, the "anomalies", at least those induced by the vertical movements of the prism, highlight the fact that the RTS can still keep the latter locked, even at such high speeds, in particular at higher speeds than the ones the tests were meant to evaluate.

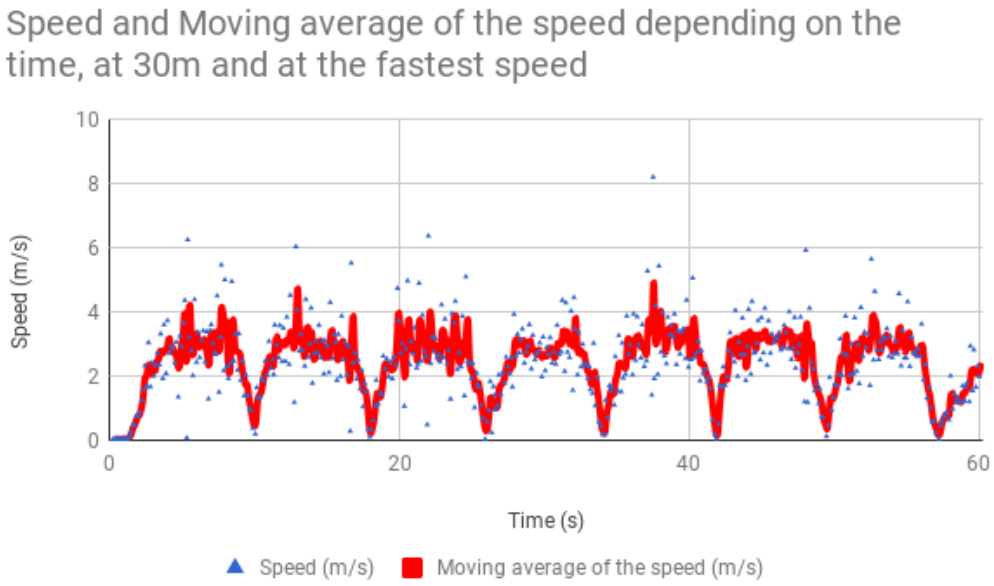


Figure 2.15: Speed of the prism (in blue) and moving average of its speed (in red), with a period of 2 (each point for the moving average is the average speed of the last two values), depending on the time, for the test where the prism was at 30m and the student moving at the fastest speed (around 2.5 m/s).

In the end, the testing procedure allowed to simply and quickly evaluate the tracking performance of the TCRP1203. However, it obviously suffers from two important drawbacks. The first one is the approximate method to move with the prism, including the undesirable vertical movements. The second one is the way to compute the speed, computed using the last two coordinates measured, which doesn't capture the real continuous motion of the prism.

Alternative ways to measure speed

There exists better ways than the method chosen. Using a Pixhawk flight controller to measure speed is possible, thanks to its accelerometer. Unfortunately, the only Pixhawk available for the thesis (a second one which was not mounted on the "small" research UAV) was faulty. The speed values returned by the device were varying from around 1m/s without any movement involved in the test and were becoming completely inconsistent when moving with it.

Roberts and Boorer [22] is a paper investigating the performance of RTS at positioning moving targets. In the experiments described, they used a potentiometer placed at the center of a circular rotating boom (figure 2.16) to measure the position of a prism, placed at the end of the boom. The RTS is then placed at a certain distance from the center of the boom. This allows comparisons between the measurements of the RTS and the potentiometer.

One could imagine a similar setting but slightly adapted to only evaluate the tracking speed. First, the total station would be placed at the center of the boom. Then, the boom would also be moved manually but the potentiometer would be used to measure the rotating speed. Thus, to test the RTS, the operator would simply move the boom at various speeds. While this is still a manual manipulation, this would remove the unwanted vertical moves and allow to move at a particular speed more accurately.

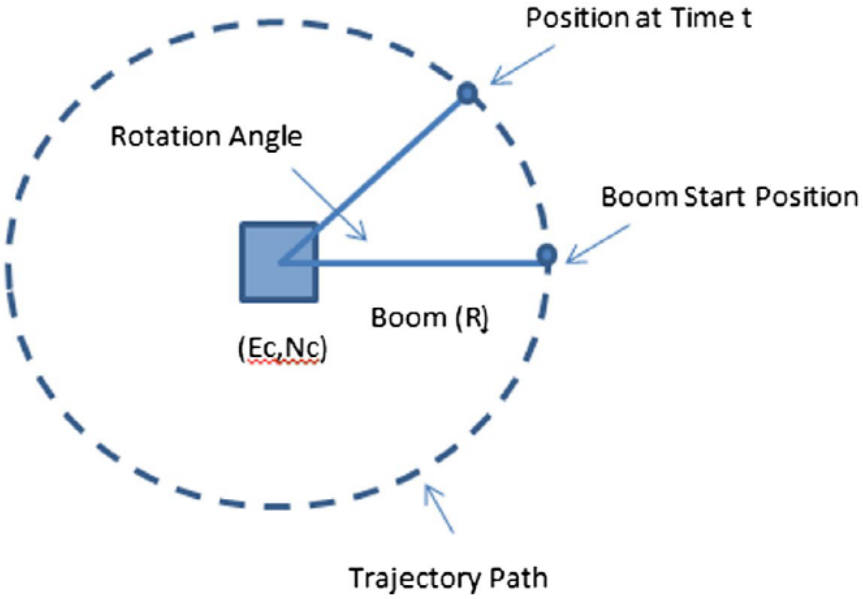


Figure 2.16: Illustration describing a circular boom used in [22], where a potentiometer was placed at its center. This allowed to compare its measurements with the ones of a Leica MS50.

Conclusion

To conclude, tracking the prism on a drone is possible with the TCRP1203. We strongly advise to place the RTS at least as far as 20m from the prism, in order for the tracking to be reliable enough. Informal tests with the prism mounted on a flying drone were also performed : the tracking was indeed almost fully reliable, as long as the drone was not doing much abrupt movements and not moving too fast ($\approx$ 1.5 m/s or less). Using more modern RTS such as the Leica Nova series (MS50, MS60) is ideal, as their performance is of course way better.

Chapter 3

Guidance system

This chapter explains the integration of the positioning system, working with the RTS and the prism, with the guidance system of ALX Systems. We will detail how the framework of ALX works concretely and why theirs was chosen.

3.1 Objective

The goal here is to integrate all the systems developed this year by master's students (RTS, UWB and image processing) with the ALX Operating System used and try to perform an autonomous flight.

3.1.1 General architecture

The idea is to put an on-board computer on the UAV, to fulfill two goals. The first one is to communicate to the flight controller when and how it needs to move. The second one is to inform the UAV about its position, given by the three different systems implemented in the context of this thesis. The GPS or another embedded system in the flight controller are not used here.

An overview of this architecture can be found in the figure 3.1.

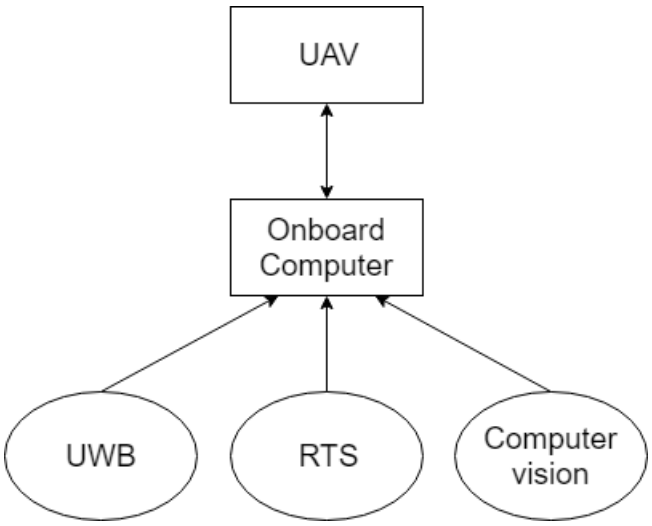


Figure 3.1: Overview of UAV guidance

3.2 Autonomous flight Operating Systems

Choosing the right guidance system for this thesis was an important choice. Two main options were available at the beginning of the project. In the end, the second one was chosen.

3.2.1 ROS – Robot Operating System

The ROS foundation describes its operating system (OS) as follows:

> ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers [28]

ROS is structured as a loosely coupled set of nodes, each of them containing a functionality communicating with others by message passing and working independently (if one crashes, the others are not directly impacted).

Advantages

- **Open source:** a big community supports the project and a lot of libraries are available and free to use;
- **Highly documented:** lots of documentation, tutorials and free online courses exist on the subject;
- **Code reuse:** many implementations have been already done for drones and are free to reuse;
- **Development stays in our hands:** no dependence on a third party team to progress on the project.

Drawbacks

- A lot of implementation needed: this system requires a lot of implementation by ourselves and thus a lot of working time;
- Lack of expertise: the system is not specifically made for UAVs, so it's not perfectly optimized.

3.2.2 ALX Systems framework

ALX Systems' framework is divided in four main components.

- **Ground station:** it is the user interface for controlling the UAV. It can run on many devices (like a tablet or a small computer). The user can create and manage missions from there and give some basic instruction as, for example, a safe return to the base, or a complete stop. This station is connected by TCP (Transmission Control Protocol) to the ALX server.

- **ALX server:** it's the connection between the ground station and the UAV. Usually, this server is set on the cloud, but for this project, and to not use resources from ALX, it's installed on the same computer as the ground station. In our case, ALX server and ALX OS are connected with a Wi-Fi connection.
- **Onboard ALX OS:** it's the core of the embedded control system on the UAV. It receives instructions from the server, information from the Pixhawk flight controller, and also information from the computer vision. After processing all the data, it gives the right instructions to the flight controller to move the UAV. The information given to the flight controller can be relative (move 10 centimeters forward) or global (for this, the data received is converted into GPS coordinates).
- **Onboard ALX Vision:** it's the computer vision system. It provides information about the environment to the UAV by processing images retrieved from on-board cameras. Information is sent to ALX OS. This computer vision can run on the same device as ALX OS but for safety reasons and performance, it is better to install it on another computer or to use a specific graphics card.

Advantages

- **Very efficient solution:** ALX's system is powerful, effective, with a lot of functionalities and security;
- **Expertise from a real company:** expertise from people working in the UAV field. It helped to solve some issues we didn't even know about our UAVs;
- **Outsourcing of the guidance system:** no need for us to develop a complete guidance system, we reuse the one from ALX Systems;
- **Opened office:** we could go sometimes to their head office in Liège, to work on our project and ask for help if necessary;
- **Ready-to-use computer vision:** we could reuse ALX Vision.

Drawbacks

- Needed to use a Pixhawk flight controller: requirement to be able to use ALX OS;
- Two onboard computers: one for ALX OS and one for ALX Vision because the computer vision can slow down ALX OS if they run on the same computer;
- **Availability of the team:** as ALX is a young startup in growing, the team is not available all the time and an appointment can be difficult to set;
- **Black box system:** we needed the company to work on the guidance system, as the latter is not open source or free to read. This constraint comes from the fact that ALX sells its system to other companies and to military organizations.

Constraints from ALX OS

There are two main constraints for the positioning system because of the use of ALX OS.

The first one is that the main script getting data from the RTS has to be written in C#, as it needs to be compiled as a C# library (".dll" file) to be provided to ALX OS. As Python 2.7 was the chosen language to develop the script, a "wrapper" class in C# executing the script in a C# environment had to be written. Fortunately, we could use IronPython, an open source module able to do exactly that. Note that IronPython has to be installed on the computer to be usable.

The second one is concerning the coordinate system used for the positioning system. The coordinates sent by the script communicating with the RTS are translated by ALX OS into GPS coordinates, to communicate with the flight controller. As GPS coordinates follow the format of geographical coordinates, where the direction in latitude and longitude have to be specified [1], our coordinate system had to be compatible. That is why the direction of the X axis is pointing to the north. In order to do this, an external compass is used and the RTS is manually pointed to the north. This practice should be evaluated in the future. Replacing the GPS module in the ALX OS with other techniques could be an alternative, but this is outside the scope of this thesis.

3.3 Drones

In order to test the guidance system, a UAV had to be configured. A small hexacopter was built, not too expensive nor too big (around 60cm in diameter), to limit security issues implied by the use of a bigger one. The following description covers how this small drone was built but the process is the same to integrate the system on bigger UAVs, for real-scale usage.

Figure 3.2 shows the main configuration of the drone and the connections set up on it. As can be noticed, two Odroid XU4 are installed on it, one for ALX OS and the other for ALX Vision. The Odroid for ALX OS is controlling the autonomous flight via a USB connection with the flight controller (PiwHawk). The latter is configured with the ArduPilot firmware, as described in the first setup guide on ArduPilot website [2]. It also receives the different information from the sensors and sends instructions to the motor controllers. The whole system is powered by a LiPo battery, coupled with a power converter (not shown in figure 3.2).

In figure 3.2, we can see two optional devices. The telemetry module is used to monitor the drone status with an external computer. It is helpful when testing the configuration of the UAV but not necessary for the usage of our system. The other one is the radio receptor. It is used to receive commands from a remote controller. When we autonomously command the UAV, it is not usable, but we left it on the UAV to do test flights before trying to fly autonomously. Furthermore, it allows to verify if the drone flies correctly, i.e. that the hardware works as expected and that the flight controller is calibrated well.

[1] Example of a GPS coordinate : 41°**N** 2°**E**, meaning 41° in latitude in the direction of the North, 2° in longitude in the direction of the East.

[2] http://ardupilot.org/copter/docs/initial-setup.html

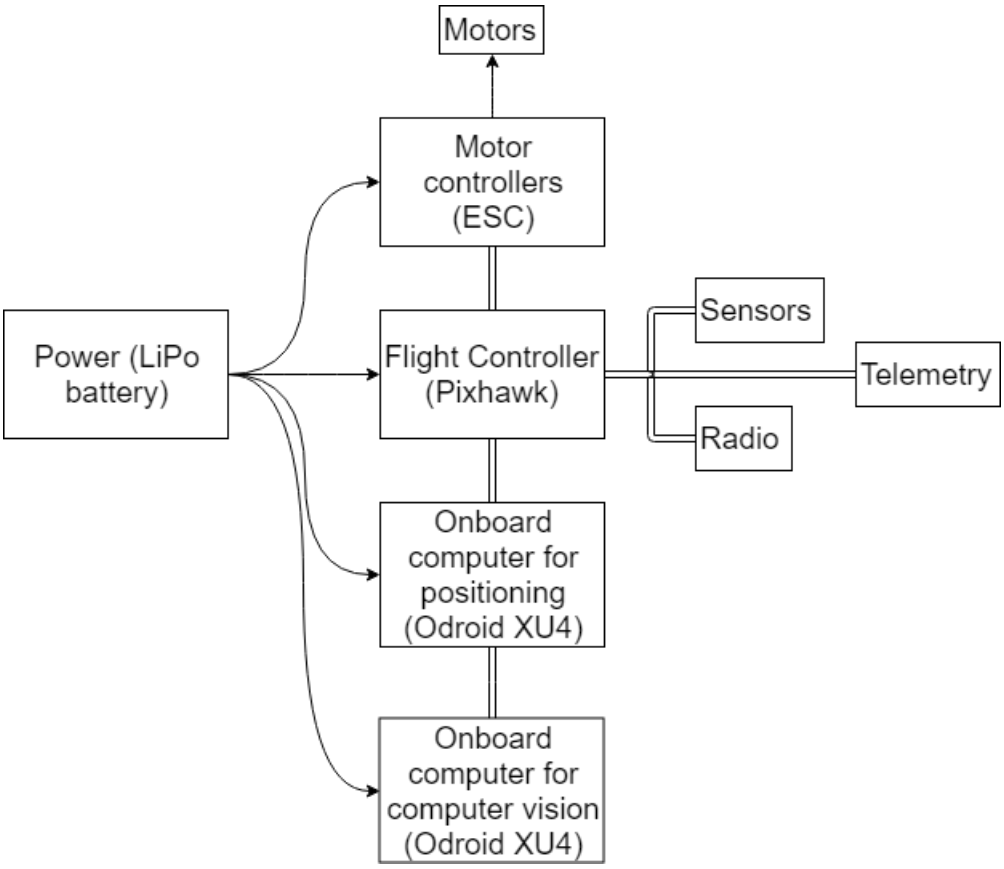


Figure 3.2: Architecture of the electronics on the drone

3.3.1 Compass necessity and related issues

To be controlled automatically, a UAV always needs to be aware of its position in space (hence in an (x,y,z) coordinate system) and of its orientation (roll, pitch and yaw angles, see figure 3.3). In this project, the position is given by the three positioning systems implemented. The orientation, however, depends on other mechanisms. The roll and pitch angles are easily and reliably be determined by the gyroscope integrated to the flight controller, while the yaw angle is measured via an internal compass. Using the latter is actually a big issue in the configuration of the drone. It is indeed mandatory because without it, the flight controller is not able to know the yaw angle of the UAV, i.e. in which direction it is oriented. Unfortunately, the compass is subject to interferences in the context of the project, as we worked in a closed environment with a lot of metallic structures and interfering Wi-Fi signals. In the future evolution of the project, it would be necessary to replace it with another system, such as a computer vision that models the environment and helps the UAV to know its orientation.

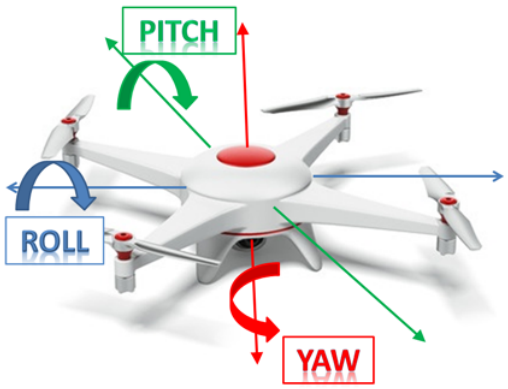


Figure 3.3: Roll, pitch and yaw angles (rotation angles around the X, Y and Z axes, respectively) illustrated on a UAV [29].

3.4 System architecture

Globally, after installing all the necessary devices on the drone and having configured all the software, the architecture of the complete system could be represented as in figure 3.4. It regroups all the positioning systems and the guidance system. The different communication channels used between devices are depicted as well.

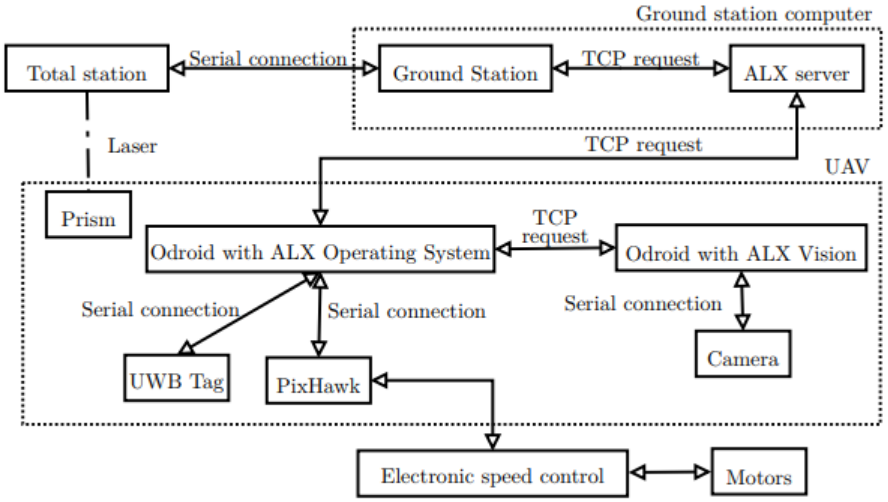


Figure 3.4: Global architecture of the guidance system. For all links named "TCP request", all of these requests are sent with a Wi-Fi connection to the Odroid with ALX OS, except the Odroid with ALX Vision ; the latter is connected to it with an Ethernet cable, as the two computers are placed side by side. The links named "Serial connection" mean that a simple cable connection is set up.

3.5 Integration of the total station positioning

To integrate our positioning system as described in chapter 2, it was necessary to write a C# wrapper to execute the code. This wrapper implements two functions used by the ALX OS. The *wrapper.cs* file can be found in the project folder, in the *scripts/src/dll* folder.

Two functions in *wrapper.cs*:

The first one is `open(port,baud)`. This function starts the Python script and configures the RTS. The port and the baudrate argument are the settings necessary to configure the serial connection to the total station. Once the latter is set up, the script continuously makes measurements and sends the valid received data to ALX OS. The second function is `close`, which is called by ALX OS when all work is done. It safely closes the script and the serial connection, to be sure we can reopen the connection later without errors.

Compilation of *wrapper.cs*

To be usable by the system, the file and all its dependencies need to be compiled in a *.dll* and copied in the ALX OS directory. Instructions on how to compile and integrate these files in the ALX OS can be found in the *README.md* file included in the *scripts/src/dll* folder.

Testing of the integration

Due to some issues (see section 5.2.3) met during the project, an autonomous flying test did not happen at the time this thesis is written. However, we tested our positioning system both with a successful integration on a simulator of ALX Systems and by manually simulating a flight. The system was responsive, functional and very promising. A future test in real conditions has been planned.

Chapter 4

Strategy and scenarios

4.1 Strategy

In parallel to the development of the guidance system, it was needed to develop a strategy for the final goal of building structures with a UAV. In addition, in case of failure of the system and for safety reasons, multiple kinds of process were thought to minimize damage to the UAV or its environment. Unfortunately, they were never implemented or used concretely, due to a few problems detailed in section 5.2.3. Their implementation is for future work on the project, as explained in section 5.3. All students working on the project this year worked on the scenarios, as it concerned the global result of autonomous construction with UAVs.

4.1.1 Global strategy

It was designed as short as possible, to keep the first tests simple. Only the main required features are kept, to let the scenario opened for later extensions.

Four main steps, executed repeatedly, have been defined:

1. Travel to the location of the brick to pick
2. Go down to approach and land on the brick to pick it up, then go up once it is done
3. Travel to the target location
4. Go down to approach and land on the target location, drop the brick and go up once it is done

In the finite state machine in figure 4.1, each node represents a step the UAV follows. The transitions represent the different readings of the sensors (e.g. to know the height) or the stored variables (e.g. to keep track of the counts) during the execution of the scenario.

The strategy execution starts at the black disk. The UAV first needs to go up to the travel height set in the system, which is set to 1m high here. Once it is high enough, the UAV enters in the *Taking* phase. It first goes to the XY coordinates of the target block. Once in position, the UAV will start to go down to pick up the brick. If something unexpected happens, a retry mechanism is triggered to restart the process from the beginning. If retries fails 3 times, the UAV tries to go to the safe zone, land and shut itself down. The *Dropping* phase is structured

the same way as the taking one but the UAV drops the block instead of taking it. If the drop works successfully and there are still blocks left to place, the whole process is restarted with the next one.

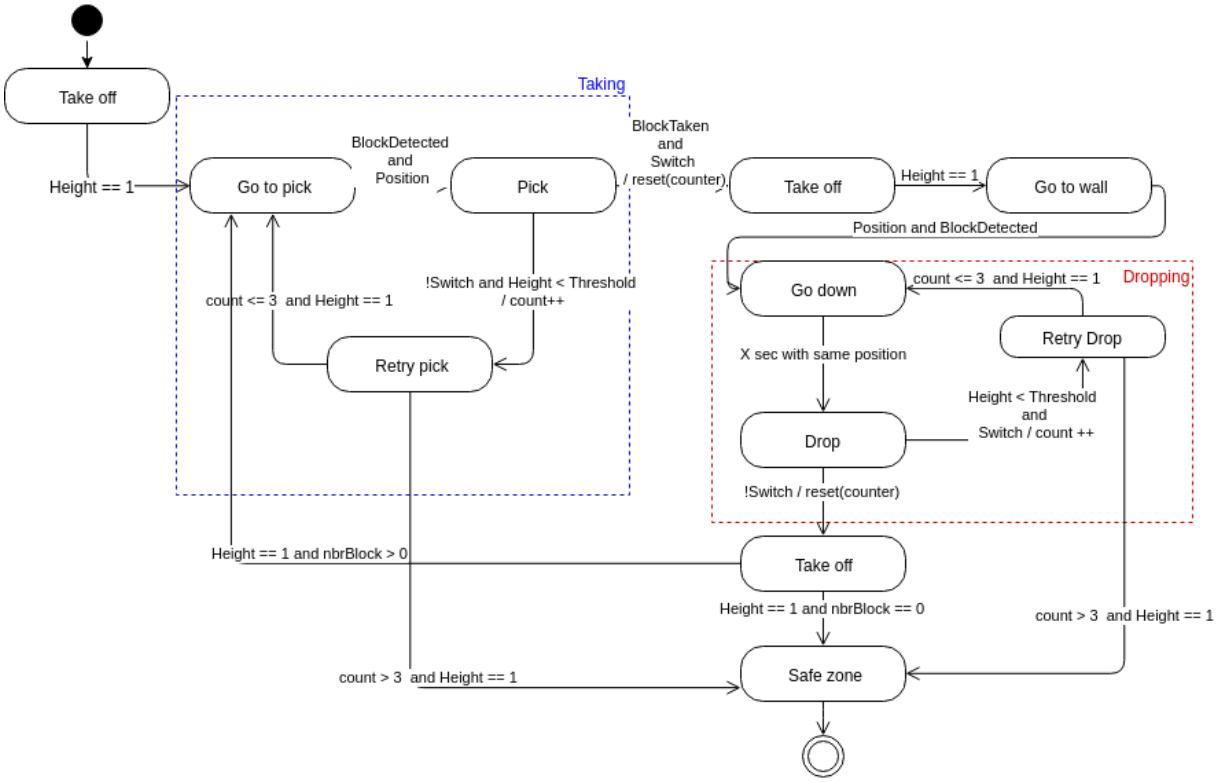


Figure 4.1: Finite state machine of the global strategy

4.1.2 Positioning systems management

The idea of our strategy is to utilize the three positioning systems developed (UWB, RTS and Computer Vision) to guide the drone. UWB is the least accurate one, but can be used to position the UAV at higher speeds than the system with the RTS, so we consider it for large moves. The RTS system is used all the time because of its precision, but still requires a backup system, if the total station lost the UAV of its field of view. Serious guidance issues arise if it happens, as the control of the UAV could be lost by the global system. UWB and the RTS are thus complementary. Finally, the computer vision positioning is only used for the *Taking* and *Dropping* phases, i.e. when taking or dropping construction elements.

4.1.3 Failure scenarios

In case of failure, it is essential to know what to do exactly and how to react. As a team, we thought on how to handle different types of failure to reduce the risk of damage. We also defined a hierarchy in the positioning systems to choose which one is the best to trust depending on the situation. As explained above, the RTS is used all the time, UWB only for broader moves (never when taking/dropping a block) and the vision is used in addition to the RTS when taking or dropping blocks.

The first scenario is the case where the UAV loses a brick during the flight. For now, we think it is better to go back to the safe zone in this kind of situation and let the operator decide if the process must continue or not. Later on, it could be possible to keep on with the construction if the position where the block is lost is not considered problematic.

The second scenario is if one of the positioning systems is lost. Concerning the UWB and RTS, if one of them is lost, we can try to go to the safe zone and land with the one left. Unfortunately, if both fail at the same time, no recover is possible and the UAV must directly land, as the computer vision is not designed to pilot it in this case. In case of instability during landing (sensor indicating an inclination higher than 15°), the motors are directly stopped. This is useful in two cases. First, if the drone is really unstable, it is safer to shutdown the motors to avoid any damage. The second case is if the UAV lands on a wall or if there is something on the floor underneath it. In this case, an inclination appears when the bottom of the UAV touches something so the motors are shut down as well.

Concerning the computer vision system, if the latter crashes, it is still possible to try restarting it while flying, during a phase not involving the taking or the dropping of blocks. If this fails, as the system is irreplaceable, it is necessary to go to the safe zone and land.

All other failures, as for example a motor failure or an embedded computer failure need to be handled by the ALX OS which, most of the time, will perform a total shutdown of the UAV.

4.2 Testing scenarios

For our application, two scenarios were developed to test that the whole system was working correctly.

4.2.1 Simple scenario

The first one is very simple. To test the UWB system and the RTS one, we decide to only make a take off, stabilize for a few minutes and land. It's quite simple to do with a radio controller manually, but this scenario is also useful to test the calibration of the system and its responsiveness.

4.2.2 Target scenario

The second one is described in figure 4.2. In this scenario, a first row of blocks is already present. It is mandatory since our system has a precision under five centimeters : due to this slight imprecision, blocks placed on the floor without any constraints could be offset by a few centimeters. If the first five blocks of this scenario would be placed with the UAV, the first layer of blocks could be totally misaligned and make the rest of the test unfeasible.

Each color on figure 4.2 is associated to a different element:

- Blue – first layer of blocks: base blocks present before;
- Orange – target block position: it is the place where we need to place the block;
- Yellow – pick-up zone: place where the UAV takes blocks. In this scenario, each block is taken at the same position and an operator refills the pick-up zone when it is empty;

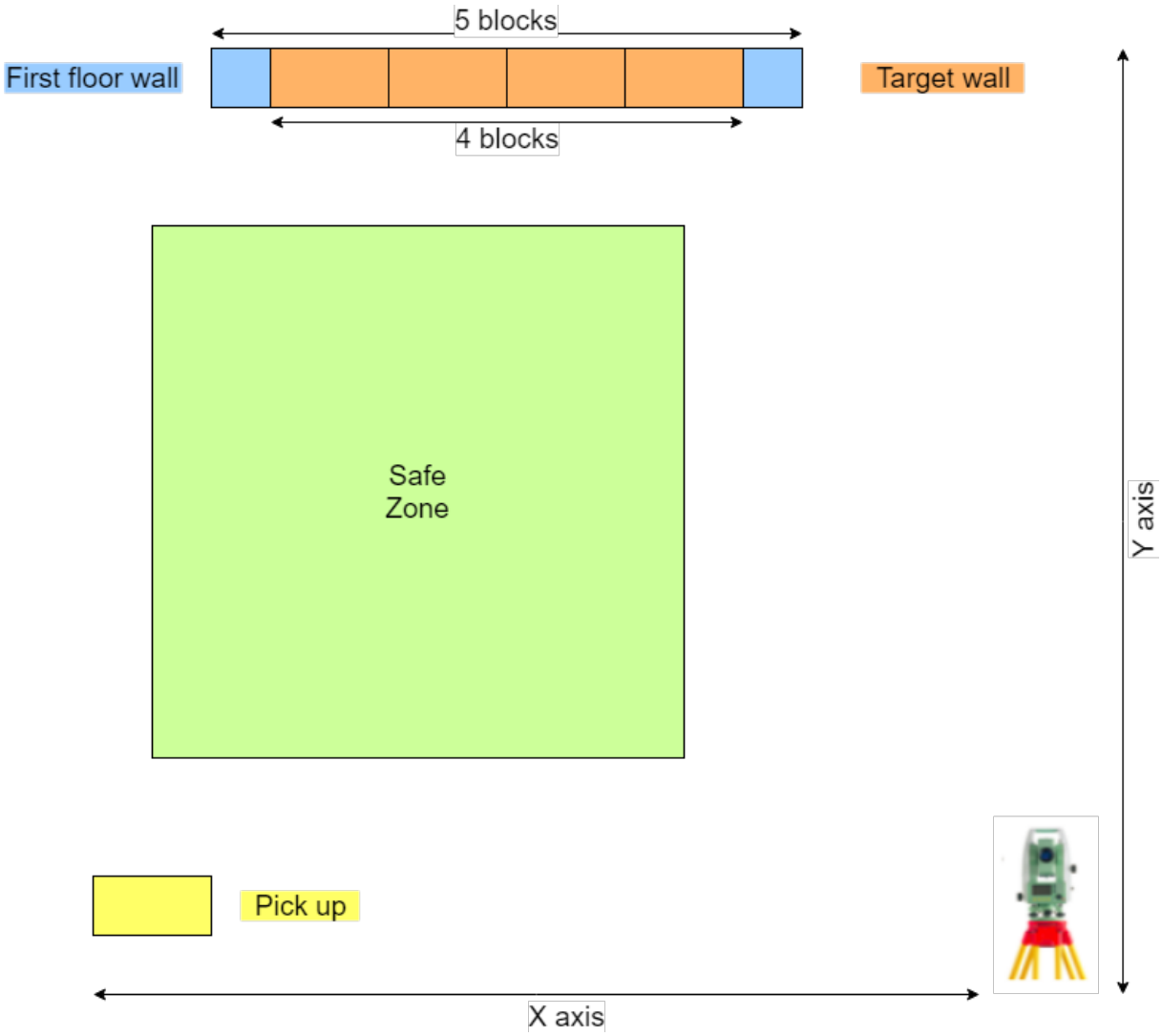


Figure 4.2: Target test scenario, where a small wall of dricks would be built with the autonomous system guiding the UAV.

- Green – Safe zone: zone where the drone can take off and land safely. It tries to land there in case of any issue;
- Total Station: needs to be placed in such a situation to have an overview of the whole testing zone (and of course to be able to use its associated positioning system).

Below is described a potential file that could be used to run the scenario. This file needs to be created with the same total station setup as the one used to guide the drone (same position on the floor and direction of X axis set to the north). Text after "//" is just comments describing the commands theoretically.

```
safe_zone(2.0;4.0;2.0;4.0)
                //safe_zone(x1;x2;y1;y2)
pick_up(5.0;0.0)
                //pick_up(x;y)
base_wall((2.0;8.0);(2.4;8.0);(2.8;8.0);(3.2;8.0);(3.6;8.0))
```

```
            //base_wall((x1;y1);...;(xn;yn))
target_wall((2.2;8.0);(2.6;8.0);(3.0;8.0);(3.4;8.0))
            //target_wall((x1;y1);...;(xm;ym))
```

Chapter 5

Conclusion

5.1 Solution developed

In the more global research project of building structures with UAVs and construction elements, we provided a system to position the UAV, using an robotic total station. The tracking performance of the RTS used, namely the TCRP1203, has been approximately evaluated. It was proven to be usable for the construction with drones, if the distance between the UAV and the RTS is at least 20m. The positioning system however was not tested as itself, only after its integration with the guidance system. It was indeed not relevant to evaluate the quality of the measurements of the TCRP1203, as they are almost millimeter accurate.

For the guidance system, two tests were performed. The first one was a successful simulation of the complete system on a simulator of ALX Systems. The second one was a "manual flight", where the UAV was manually moved and information returned by the system was verified ; this test turned out to be successful too.

5.2 Difficulties encountered

5.2.1 Interferences in the DroneZone

The area is subject to interferences. Routers were installed at many places by the university, to offer a Wi-Fi connection on campus as much as possible. As a consequence, Wi-Fi signals are continuously emitted in the area and they can interfere with the communication between the ground station and the onboard Odroid.

Moreover, it is a closed area where many metallic structures are present in its surroundings, which absorb Wi-Fi signals and can make the overall communications worse.

5.2.2 PixHawk's Compass

The aforementioned interferences can make the compass on the flight controller unreliable. Furthermore, on a real-scale construction site, many metallic structures can be present and can make the communications to a working drone unreliable. Eventually, if the project of construction with UAVs turns out well, the current strategy is not suitable for real-scale construction projects. Alternative ways of knowing the orientation of the drone are discussed in the next section.

5.2.3 ALX Systems

It was Geoffrey Mormal, the CEO of the company, who mainly helped us to install and configure ALX OS and ALX Vision on our devices. As the company is still a startup with only a few employees, G. Mormal has a tight schedule. In addition, he is the main developer of ALX OS and the one that knows it the best. It was thus complicated for him to regularly save a day to help us on the project. As a consequence, a concrete autonomous flight could not be performed on time.

5.3 Limitations and future work

5.3.1 UAV Positioning with an RTS

Multiple limitations appear when using an RTS for the positioning system.

Firstly, the prism (so the drone as well), has to stay in sight all the time, otherwise no positioning at all is possible. This could be an issue on a construction site because any object can potentially obstruct the view between the RTS and the drone. A possible solution would be to use multiple RTS on site: this would make the system more reliable and efficient as completely losing the prism would be less likely to happen. The price however might increase a lot, especially if modern RTS such as a Leica MS60 are used.

Secondly, the range of the system is limited to 300m with the 360° mini-prism, for the TCRP1203. It would not be that much of a problem on a construction site. Nevertheless, the range can still be improved by using a 360° "big" prism, since its weight would not be a problem at all on a drone capable of lifting construction elements. A second solution would be to use a more modern RTS: the maximal range in LOCK mode with the MS50 is 600m and goes up to 1km for the MS60. However, the maximal range with a mini-prism is worst than with the TCRP1203 (200m and 250m, respectively), so both RTS are probably not designed to be used with one.

Thirdly, the tracking performance of the TCRP1203 is still an issue, especially at close range. It gets better with an increasing distance and using it for real-scale construction would be possible, but again, a modern RTS is way better. For the record, the maximal tangential speed to which the MS50 and MS60 can track is 9 m/s, almost twice as fast as the TCRP1203.

5.3.2 UAV configuration and guidance

As mentioned previously, using the compass of the flight controller can be problematic. Fortunately, other alternatives to help the UAV know its orientation exist.

A first good solution would be to use an embedded computer vision program within a known 3D environment, using only an onboard RGB camera. The images provided by the latter can be compared to the 3D model of the environment to provide orientation information to the UAV. The approach has been recently investigated by Li-Chee-Ming & Costas [24]. Concretely, it is feasible to create a 3D CAD model of the construction site used as a reference for the UAV. To ensure the reliability of the system, it would even be possible that the UAV performs a few "reconnaissance flights" to model the environment itself (i.e. SLAM process) at first.

As second alternative, one could use UWB beacons, in particular by placing two "tags" on the drone. Indeed, as explained in [6], by keeping track of the positions of both UWB tags, one can keep track of the orientation. Concretely, the UAV just needs to compare the current positions with the ones previously measured (figure 5.1). Regarding the results of the positioning system using UWB beacons (precision of 18.7cm along the X and Y axes), this is probably not the most accurate way to determine the orientation, but it still needs to be tested in practice.

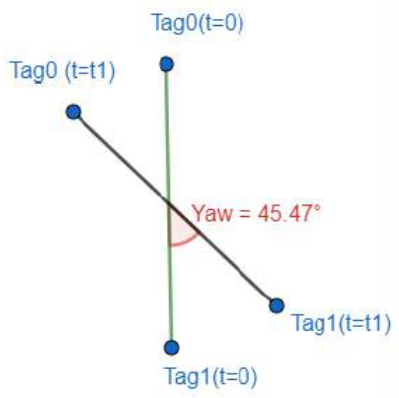


Figure 5.1: Estimation of the orientation of the UAV with two UWB beacons mounted on it. This angle is calculated by comparing the positions of *Tag0* and *Tag1* at a time $t = t_1$ with their positions at the previous time ($t = 0$ in this case).

Regarding the guidance of the UAV, as explained previously, we were not able to perform an autonomous flight with the test drone set up, the three positioning systems, ALX OS and ALX Vision installed. A future first concrete test outdoors would of course be ideal to confirm that the system works as expected.

Concerning the systems of ALX, even if using them has been a very convenient help for the guidance and allowed us to focus on the positioning system, they are proprietary systems and we thus depend on the company for a long-term use. An easier solution for future work on the project would be to switch to a ROS program performing the integration of the three systems and handling the guidance of the UAV.

5.3.3 Realizing the planned scenarios

If this interesting global project is further developed (either by continuing with ALX Systems or by using ROS), being able to pick up and drop dricks autonomously would be a very promising step for the autonomous construction with UAVs. But going even further by successfully accomplishing the target scenario of section 4.2.2, will be the most decisive step and prove, at least at small scale, that it is feasible.

Bibliography

[1] Latteur P., Goessens S., Breton J-S., Leplat J., Ma Z., Mueller C. T. *Drone-Based Additive Manufacturing of Architectural Structures*, presented at the International Association for Shell and Spatial Structures (IASS) Symposium, Amsterdam, Netherlands, 2015.

[2] Latteur P., Goessens S., Mueller C. T. *Masonry construction with drones*, presented at the International Association for Shell and Spatial Structures (IASS) Symposium, Tokyo, Japan, 2016.

[3] Leboutte J-F. and Parisel V., master's thesis under the supervision of Latteur P. *Développement d'un système constructif "drone-compatible" pour la construction de maisons unifamiliales en maçonnerie et en béton.* Université Catholique de Louvain, Belgium, 2017.

[4] Goessens S., Mueller C., Latteur P. *Vers une robonumérisation de la construction*, RUGC, Liège, 25-26-27 May 2016.

[5] Jacques T., Kaczynski F., master's thesis under the supervision of Latteur P. and Sadre R. *Development of a local positioning system for builder drones by image processing.* Université Catholique de Louvain, Belgium, 2018.

[6] Bouhaddi Z., master's thesis under the supervision of Latteur P. and Flémal C. *Etude et développement d'un système de localisation et de guidage d'un drone basé sur la technologie UWB*, ECAM / UCL, Belgium, 2018.

[7] ALX Systems' website. `http://www.alxsys.com/`. Accessed : 30-05-2018.

[8] Nehri N., Paques A., master's thesis under the supervision of Latteur P., Marchand C. and Goessens S. *Etude et conception de l'architecture de contrôle d'un drone destiné à la construction d'une structure en intérieur.*, ECAM / UCL, Belgium, 2016.

[9] Maxim A., Lerke O., Prado M., Dörstelmann M., Menges A., Schwieger V. *UAV Guidance with Robotic Total Station for Architectural Fabrication Processes.* University of Stuttgart, Germany, 2017.

[10] Maxim A., master's thesis under the supervision of Menges A. and Knippers J. *MAV Assisted Fabrication Strategy for Large Scale Fiber-Composite Structures.* Institute for Computational design, University of Stuttgart, Germany, 2016.

[11] Augugliaro F., Lupashin S., Hamer M., Male C., Hehn M., Mueller M.W., Willmann J., Gramazio F., Kohler M., D'Andrea R., *The Flight Assembled Architecture Installation:*

Cooperative construction with flying machines, IEEE Control Systems Magazine, Volume 34, Issue 4, 2014, pp. 46 – 64.

[12] Augugliaro F., Mirjan A., Gramazio F., Kohler M., and D'Andrea R., *Building tensile structures with flying machines*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013.

[13] Mirjan A., Gramazio F., Kohler M., Augugliaro F., and D'Andrea R., *Architectural fabrication of tensile structures with flying machines*, in Green Design, Materials and Manufacturing Processes. CRC Press, 2013.

[14] Augugliaro F, Zarfati E., Mirjan A., and D'Andrea R., *Knot-tying with flying machines for aerial construction*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015.

[15] Leica Geosystems. *Leica TPS1200 : Technical Reference Manual*, V5.5, English, Heerbrugg, Switzerland, 2007.

[16] Leica Geosystems. *Leica TPS1200 : User Manual*, V5.5, English, Heebrugg, Switzerland, 2007.

[17] Leica Geosystems. *Leica MS60/TS60 : User Manual*, V3.0, English, Heerbrugg, Switzerland, 2016.

[18] Leica Geosystems. *Leica Nova Series : Technical Reference Manual*, V2.1, English, Heerbrugg, Switzerland, 2016.

[19] Leica Geosystems. *Leica TPS1200 : GeoCOM Reference Manual*, V1.20, English, Heerbrugg, Switzerland.

[20] Mao J. and Nindl D. (Leica Geosystems). *Surveying Reflectors - White Paper*, Heerbrugg, Switzerland, March 2009.

[21] Wikipedia, *Baud*, `https://en.wikipedia.org/wiki/Baud`. Accessed : 06-06-2018.

[22] Roberts C. and Boorer P. *Kinematic positioning using a robotic total station as applied to small-scale UAVs*, Journal of Spatial Science, Taylor & Francis Group, 2015.

[23] Ashraf A.A. Beshr and Islam M. Abo Elnaga. *Investigating the accuracy of digital levels and reflectorless total stations for purposes of geodetic engineering*, Alexandria Engineering Journal (2011) 50, 399–405.

[24] Li-Chee-Ming J. and Armenakis Costas. *UAV navigation system using line-based sensor pose estimation*, Journal of Geo-spatial Information Science, 21:1, 2-11, Taylor & Francis Group, 2018.

[25] LEICA TCRP1203 R300, Topotienda, `https://topotienda.com/estaciones/964-leica-tcra1203-r100`. Accessed : 30-05-2018.

[26] Leica GRZ4 360° Prism, Opti-cal Survey Equipment, http://surveyequipment.com/leica-grz4-360-prism/. Accessed : 30-05-2018.

[27] Spherical coordinates in Matlab, https://mse.redwoods.edu/darnold/math50c/mathjax/spherical/index.xhtml. Accessed : 30-05-2018.

[28] Dirk Thomas, *ROS/Introduction - ROS*, http://wiki.ros.org/ROS/Introduction. Accessed : 30-05-2018.

[29] DroneKiezer, https://www.dronekiezer.nl/wp-content/uploads/2015/11/drone-pitch-yaw-roll.png. Accessed : 08-06-18.

Appendix A

Delivered content

The `zip` file delivered with this report contains 2 main folders and a `README.md` file at the root. Each folder also contains a more specific `README.md` file for more specific explanations.

A.1 `scripts` folder

Contains all the necessary scripts for the positioning system. It is based on the work of Artyom Maxim [1].

A.1.1 src

The sources folder contains the implementations of GeoCOM and the tracking scripts in *Python 2.7*. One of the scripts uses an improved search method but not compatible with every total station (the *PowerSearch*) as it requires an additional sensor, and the other one uses a "classic search".

A.1.2 tests

The tests folder contains the tracking script modified for testing as explained in section 2. It also contains the output of each test.

A.1.3 documentation

This folder contains the sources and the output of the documentations framework we used. A pdf version of the documentation can be found in appendix F.

A.2 `util` folder

This folder contains all non coded resources like GeoCOM manual, Leica drivers for cables, 3D printed model for the small UAV, etc.

[1] `https://github.com/art-mx`

Appendix B

Connecting a PC (Windows 10) to the TCRP1203 and MS50-60

B.1 USB drivers for GEV267,GEV268,GEV269 cable

These drivers are available in the `util` folder or on the Leica MyWorld Website[1]

B.1.1 Install the driver on Windows 10

- Download the driver and extract it (already done in this folder)
- Plug-in the cable on the Windows computer
- Go to "Device Manager"
- Find the device "FT232R"
- Right click and select "Update Driver"
- Browse your computer into the extracted folder
- Select the folder "Windows XP, Server 2003, Server 2008 R2,Vista, 7, 8"
- Click on next to finish the installation

After this process, it is possible you need to do it again with the new device showed in the list called "Serial USB Converter".

A complete step by step PDF file describing the whole procedure for all systems is available in the zipped folder of the driver.

[1] https://myworld.leica-geosystems.com/irj/portal

Appendix C

README.md

This file is an explanation included in the path of the project, to explain its architecture and how to run the code.

README.md

Development of a guidance and automatic positioning system for builder drones with a total station

Credit

Authors : Nicolas Sorensen & Rémy Vermeiren

Promotors : Pierre Latteur & Ramin Sadre

Assistant : Sebastien Goessens

Repository :

Based on the work of Maxim Artyom and Georg Wiedebach

Trello and Slack

Resources

Markdown Cheatsheet

Python documentation convention

Documentation

The documentation generation is automated using Sphinx. The scripts generating the documentation for `track.py` and `GeoCom.py` are available in the folder `scripts\documentation`.
To generate it on a Windows system : see `scripts\documentation\make.bat`.
To generate it on a Unix system:

- Open a terminal
- Go into the `scripts` folder
- Type `make html` to generate it in HTML
- Type `make latexpdf` to generate a PDF

Further information is available in `scripts\documentation\Makefile`.

To change the directory of the source files of the project where Sphinx looks up for comments (in this case the directory of `track.py` and `GeoCom.py`), go in `scripts\documentation\conf.py` and set `sys.path` to the directory of your choice.

Installation

Requirements

- Windows 10
- Leica cable GEV267 for Total station TCRP1203
- Leica cable GEV269 for Total station MS50, MS60
- Total station with GeoCom support (TCRP1203, MS50, MS60)
- Python2.7
- pip package manager for Python 2.7 (normally already included in Python installation)
- pyserial
  - Install with `C\:Python27\python.exe -m pip install pyserial` in the command line

- USB cable drivers for GEV267,GEV268,GEV269 V3.0 available on Leica MyWorld website or on the "/util/" folder of this repository
  - Install the driver :
    - Download the driver and extract it
    - Plug-in the cable on the Windows computer
    - Go to "Device Manager"
    - Find the device "FT232R"
    - Right click and select "Update Driver"
    - Browse your computer into the extracted folder
    - Select the folder "Windows XP, Server 2003, Server 2008 R2,Vista, 7, 8"
    - Click on next to finish the installation
  - After this process, it's possible you need to redo this with the new device showed in the list called "Serial USB Converter"

Get the source codes

You can download the code on this repository: https://github.com/rvermeiren/Leica-GeoCOM-for-drone-tracking If the link is dead, contact Nicolas Sorensen or Rémy Vermeiren

Run and Usage

Run

```
$ C:\Python27\python.exe track.py
-d (verbose for debug)
-b (big prism -- default = mini-prism)
-p "port" (ex: "COM1" -- This can be found in "Device Manager" on Windows)
```

Usage

Use `-h` to show usage

```
  Options:
  -h,           --help            show this help message and exit
  -p PORT,      --port=PORT       specify used port [default: /dev/ttyUSB0]
  -b BAUDRATE,  --baudrate=BAUDRATE
                                  specify used baudrate [default: "COM3"]
  -d, --debug                     show debug information
  -B, --Big                       use 360 big prism
```

Licenses

The original work was under the following copyright :

```
#Copyright (c) 2013, Marcel Schoch, ASL, ETH Zurich, Switzerland
#You can contact the author at <slynen at ethz dot ch>
#
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions are met:
# * Redistributions of source code must retain the above copyright
#notice, this list of conditions and the following disclaimer.
# * Redistributions in binary form must reproduce the above copyright
#notice, this list of conditions and the following disclaimer in the
#documentation and/or other materials provided with the distribution.
# * Neither the name of ETHZ-ASL nor the
#names of its contributors may be used to endorse or promote products
#derived from this software without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
```

```
#ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#DISCLAIMED. IN NO EVENT SHALL ETHZ-ASL BE LIABLE FOR ANY
#DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
#ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Appendix D

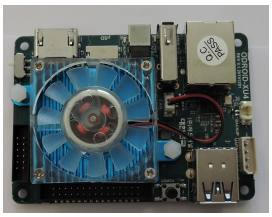
Material

Image	Category and name	Description
	**Total Station:** Leica TCRP1203 (TPS1200 series)	Robotic total station, released in 2004. Can be programmatically controlled with the GeoCOM protocol. Max theoretical tangential speed in LOCK mode : 5m/s at 20m. Borrowed to Pr. S. Lambot (AGRO). Bought in 2004 by E. Mourmaux, member of the CEIU of the university and sold to the AGRO faculty in 2015.
	**Total Station:** Leica MS50 (Leica Nova Series)	Robotic total station, released in 2013. Can be programmatically controlled with the GeoCOM protocol. Max theoretical tangential speed in LOCK mode : 9m/s at 20m. Borrowed twice to E. Mourmaux (CEIU).

	**Total Station:** Leica MS60 (Leica Nova Series)	Robotic total station, released in 2015. Can be programmatically controlled with the GeoCOM protocol. Max theoretical tangential speed in LOCK mode : 9m/s at 20m
	**Total station:** Tripod GST series	Tripod on which a total station can be firmly fixed.
	**Total station and drone:** Leica GRZ4 360° prism	Leica Constant/Offset : +23.1 mm Centering accuracy : 2mm Max LOCK range for TCRP1203 : 500m Max LOCK range for MS50 : 600m Max LOCK range for MS60 : 1km
	**Total station and drone:** Sokkia ATP1 360° mini-prism	Leica Constant/Offset : +30.0 mm Centering accuracy : 1.5mm Max LOCK range for TCRP1203 : 300m Max LOCK range for MS50 : 200m Max LOCK range for MS60 : 250m

	**Total Station:** Leica GEV267 cable	Connects the TCRP1203 to a PC via USB. Sets up a serial connection between the RTS and the PC.
	**Total Station:** Leica GEV269 cable	Connects the MS50 and MS60 to a PC via USB. Sets up a serial connection between the RTS and the PC.
	**Drone:** DJI F550 custom-built	Testing drone used for the development of the project.
	**Drone:** Medium drone	Drone mainly designed to test the positioning and guidance systems on a bigger drone than the previous one.

	**Drone:** Big drone : real prototype	Prototype built in the context of the global research on masonry work with UAVs [2, 4]. It has a maximum payload of 40kg.
	**Drone:** 4S Lithium-Ion battery	Battery used to power the small UAV and all the electronics on it.
	**Drone:** Remote controller	Remote controller to control the small UAV.
	**Drone:** Pixhawk PX4	Flight controller. See documentation on `https://pixhawk.org/`.

	**Drone:** Odroid XU4	Embedded computer managing the whole system of the drone. Two of them are mounted on the UAV (one for vision, the other for the rest). More powerful than a Rasperry Pi 3, it contains an 8-core CPU and 2GB of RAM.
	**Drone:** 3D printer	Personal 3D printer used to print a prism holder, a battery holder and a case for the UWB tag. The 3D models designed are explained in appendix E.
	**Construction:** Drick	Drick designed in 2017 by Leboutte J-F. and Parisel V. [3], for their master's thesis. This is the block design used for tests and developments throughout this year, especially by the Computer Vision team.

Appendix E

3D designs

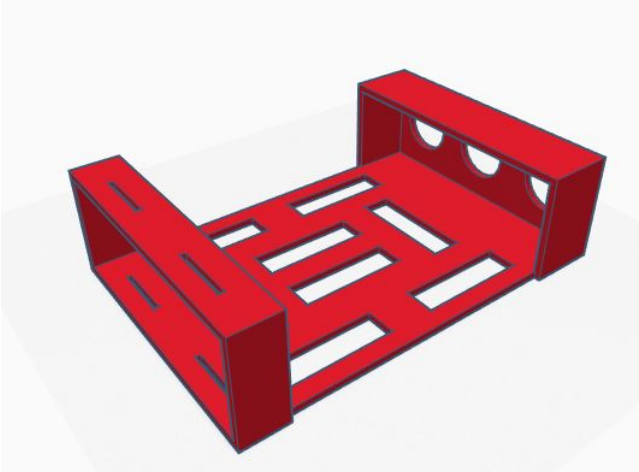


Figure E.1: Battery holder: to be fixed under the F550, can hold one or two LiPo 4S batteries of 3700mAh or one LiPo 4S battery of 8000mAh.

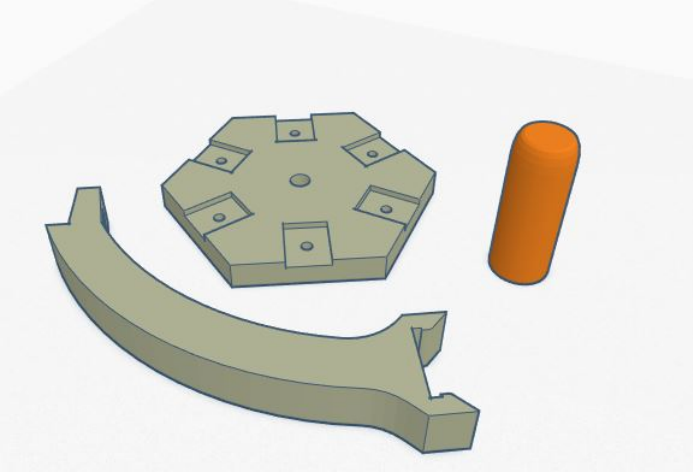


Figure E.2: Prism holder: used to hold the prism at the top of the F550 drone. Picture of utilization in the figure E.3

Figure E.3: Prism holder utilization.

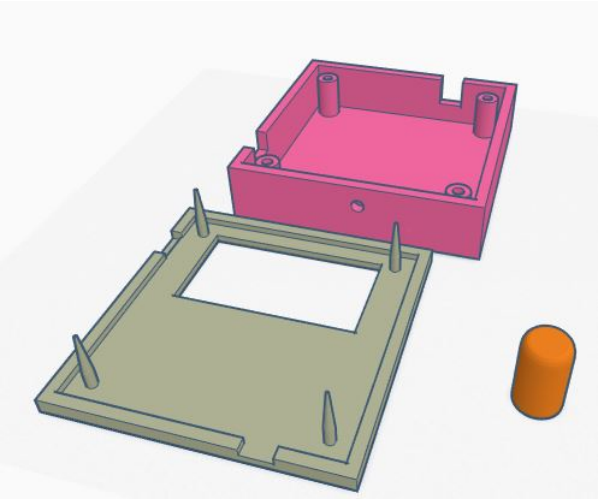


Figure E.4: UWB tag case, to be placed on top of the prism holder or on top of the prism itself.

Appendix F

Documentation

This documentation (originally in HTML, converted to PDF here) gathers information that can be found in the comments of the Python scripts `track.py` and `GeoCom.py`. The Main script is `track.py`. It interacts with the total station to request measurements and displays them on the standard output.

GeoCOM functions here are all functions from Leica's GeoCOM protocol that are implemented in the file `GeoCom.py`.

Main script

`track.`**`compute_carthesian`**(*phi*, *theta*, *radius*) [source]

Compute carthesian coordinates using vertical, horizontal angles and distance measurements.

Parameters:
- **phi** (*float*) – horizontal angle (rad)
- **theta** (*float*) – vertical angle (rad)
- **radius** (*float*) – distance from the station to the prism (m)

Returns: a string with the coordinates, formatted as x;y;z

Return type: str

`track.`**`connection`**(*options*) [source]

Opens a serial connection between the computer and the total station.
Calls **sys.exit** if the connection set up failed.

Parameters: **options** (*Namespace*) – contains the options to configure the connection

`track.`**`get_measure`**() [source]

Request a complete measurement (angles and distance) to the station and handles the possible errors returned by the station.

After 100 failed distance measurements, run a search to try to lock on the prism again.

Returns:

- The coordinates of the prism if :
  - the measurement was successful (RC=0)
  - the accuracy coudln't be guaranteed by the system of the station, but a complete measurement was still possible (RC==1284)
- "2" if only the angles could be measured (RC=1285 or RC=1288)
- "3" if another error occured or if a non-numeric value was received
- "4" if a GeoCom.SerialRequestError occured

Return type: str

`track.`**`powerSearchPrism`**(*cHz=0*, *cV=1.57*, *aHz=1.0*, *aV=1.0*) [source]

Performs a PowerSearch, starting from the angular position (cHz, cV) and searching in the given window (aHz, aV).

Parameters:
- **cHz** (*float*) – starting horizontal angular position
- **cV** (*float*) – starting vertical angular position
- **aHz** (*float*) – horizontal search window
- **aV** (*float*) – vertical search window

Returns: True if the prism is locked, False otherwise

Return type: bool

`track.`**`searchPrism`**(*Hz=20*, *V=20*) [source]

Search for the prism in the given area.

Parameters:
- **Hz** (*int*) – horizontal area in degrees
- **V** (*int*) – vertical area in degrees

Returns: True if the prism is locked, False otherwise

Return type: bool

`track.`**`set_laser`**(*value*) [source]

Turn on/off the laser of the total station.

Parameters: **value** (*int*) – on (value=1) or off (value=0)

`track.`**`set_prism_type`**(*big_prism*) [source]

Set the prism type as “360 big prism” if *big_prism* is True, or to “360 small prism” if False.

Parameters: **big_prism** (*bool*) – Determines if the type of prism if “big” or “small”

`track.`**`set_x_axis`**() [source]

Set the orientation of the carthesian plan by fixing **x** axis.

`track.`**`setup_station_manual`**(*options*) [source]

Set up the station for the purpose of tracking a prism and make fast reapeated measurements.

Parameters: **options** (*Namespace*) – contains the options to configure the station
Returns: True if the setup succeeded, False otherwise
Return type: bool

`track.`**`usage`**(*COM='COM3'*, *baud=57600*) [source]

Define and show usage of the script.

Parameters:
- **COM** (*str*) – number of the COM port to which the USB cable is connected.
- **baud** (*int*) – baud rate of the communication between the PC and the total station.

A higher baud rate will allow more measurements per second but may cause problems (e.g. lack of precision), while a lower one will make less measurements per second but they will be more reliable.

> Warning:
> The baud rate HAS to be the same as the one set on the total station! Otherwise the script won’t work correctly.

Returns: list of values set for the options, or default values
Return type: Namespace object

GeoCOM

All functions specified here are specified as in the TPS1200 GeoCOM reference manual, **V1.20**. Not all of them are specified: the ones used in the main script are described in details, while the others simply redirect to the page of the reference manual where they are described.

Not all functions available in GeoCOM are implemented in this script.

In the specification of some GeoCOM functions, not all RC codes possibly returned by the function are given and described. You can easily check them in the GeoCOM reference manual.

Classes

class GeoCom.**ResponseClass** [source]

Manage the response from the station (transaction ID and parameters returned) and the error codes returned.

Attribute RC_COM:
communication return code
Attribute TrId: transaction id
Attribute RC: request return code
Attribute parameters:
list of returned parameters

setResponse(*response*) [source]

Instantiate a ResponseClass from an ASCII response.

Parameters: **response** (*ResponseClass*) – ASCII response from the station

GeoCom.**getTrId**(*request*) [source]

Get transaction ID from an ASCII request by parsing it.

Parameters: **request** (*str*) – an ASCII resquest
Returns: parsed transaction ID
Return type: int

GeoCom.**HexToDec**(*hex_in*) [source]

Convert an hexadecimal number into a decimal number.

Parameters: **hex_in** (*int*) – hexadecimal number to convert
Returns: decimal representation of hex_in
Return type: int

GeoCom.**SerialRequest**(*request*, *length=0*, *t_timeout=3*) [source]

Send a request to the server (total station).

Parameters:
- **request** – an ASCII request
- **length** – to remove later, not needed
- **t_timeout** – default is 3 seconds, could be higher or lower

Returns: the corresponding response

Return type: ResponseClass

Raises: **SerialRequestError** – thrown if an error occurs during the communication

GeoCom.**CreateRequest**(*cmd*, *args=None*) [source]

Create an ASCII Request based on a command code and, if needed, corresponding arguments.

Parameters:
- **cmd** – function code to send to the Station
- **args** – list of arguments

Returns: an ASCII request with this form [<LF>]%R1Q,cmd,<TrId>:[args]<Term>

Return type: str

exception GeoCom.**SerialRequestError** [source]

COM – Communication

GeoCom.**COM_CloseConnection**() [source]
[GeoCOM manual **p27**]

Close the (current) open port and releases an established connection.

Returns: [error, RC, []]. error=0 and RC=0 if the request is successful.
Return type: list

GeoCom.**COM_OpenConnection**(*ePort*, *eRate*, *nRetries=10*) [source]
[GeoCOM manual **p26**]

Open a PC serial port and attempts to detect a theodolite based on the given baud rate.

Parameters:
- **ePort** (*str*) – serial port
- **eRate** (*int*) – baud rate
- **nRetries** (*int*) – number of retries to initiate a connection

Returns: [error, RC, []]

- error=0 and RC=0 if the connection attempt was successful
- error=1 if not

Return type: list

GeoCom.**COM_SwitchOnTPS**(*eOnMode=2*) [source]
[GeoCOM manual **p96**]

GeoCom.**COM_SwitchOffTPS**(*eOffMode=0*) [source]
[GeoCOM manual **p97**]

GeoCom.**COM_GetSWVersion**() [source]
[GeoCOM manual **p95**]

AUT – Automation

GeoCom.**AUT_MakePositioning**(*Hz*, *V*, *POSMode=0*, *ATRMode=0*, *bDummy=0*) [source]
[GeoCOM manual **p49**]

GeoCom.**AUT_Search**(*Hz_Area*, *V_Area*, *bDummy=0*) [source]
[GeoCOM manual **p56**]

Performs an automatic target search within a given area.

Parem Hz: horizontal search region [rad]

Parameters:
- **V** (*int*) – vertical search region [rad]
- **bDummy** (*int*) – reserved for future use, always set to false (quick reminder: *0* in GeoCOM is considered false)

Returns: [error, RC, []]

- error=0 and RC=0 if the search is successful
- error=1 and RC=8710 if not

Return type: list

GeoCom.**AUT_FineAdjust**(*dSrchHz=0.1*, *dSrchV=0.1*) [source]

[GeoCOM manual **p54**]

Precisely positions the telescope crosshairs onto the target prism.

Parameters:
- **dSrchHz** (*float*) – Search range Hz-axis [rad]
- **dSrchV** (*float*) – Search range V-axis [rad]

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 otherwise

Return type: list

`GeoCom.AUT_LockIn`() [source]

[GeoCOM manual **p60**]

If LOCK mode is activated (AUS_SetUserLockState), then the function starts the target tracking. The command is only possible if a AUT_FineAdjust command has been previously sent and successfully executed.

Returns: [error, RC, []]

- error=0 and RC=0 if the lock is successful
- error=1 if not

Return type: list

`GeoCom.AUT_GetSearchArea`() [source]

[GeoCom **p61**]

Returns the current position and size of the PowerSearch Window.

Returns: [error, RC, parameters]

- error=0 and RC=0 if the request is successful
- error=1 if not
- parameters contains the position and size of the PowerSearch Window

Return type: list

`GeoCom.AUT_SetSearchArea`(*dCenterHz*, *dCenterV*, *dRangeHz*, *dRangeV*, *bEnabled=1*) [source]

[GeoCom **p62**]

Define the starting position of the search and its search area, then activates these PowerSearch parameters if bEnabled=1.

Parameters:
- **dCenterHz** (*float*) – starting horizontal angular position
- **dCenterV** (*float*) – starting vertical angular position
- **dRangeHz** (*float*) – horizontal search window
- **dRangeV** (*float*) – vertical search window
- **bEnabled** (*int*) – activate (=1) or deactivate(=0) the parameters set for PowerSearch.

Returns: [error, RC, parameters]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

`GeoCom.AUT_PS_EnableRange`(*bEnable*) [source]

[GeoCom **p65**]

Enables (bEnable=1) / disables (bEnable=0) the predefined PowerSearch window, including the predefined PowerSearch range limits, set by AUT_PS_SetRange.

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful

- error=1 if not

Return type: list

GeoCom.**AUT_PS_SetRange**(*lMinDist*, *lMaxDist*) [source]
[GeoCom **p66**]

Define the PowerSearch range limits.

Parameters:
- **lMinDist** (*float*) – range lower bound
- **lMaxDist** (*float*) – range upper bound

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

GeoCom.**AUT_PS_SearchWindow**() [source]
[GeoCom **p67**]

Start PowerSearch inside the given PowerSearch window, defined by AUT_SetSearchArea and inside the optional range given by AUT_PS_SetRange.

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not
  - RC = 7 if bad arguments were given (e.g. wrong format or wrong number of args)
  - RC = 26 if function not successfully completed
  - RC = 8720 if the working area is not defined
  - RC = 8710 if no target was found

Return type: list

CSV – Central services

GeoCom.**CSV_GetDateTime**() [source]
[GeoCOM manual **p107**]

EDM – Electronic Distance Measurement

GeoCom.**EDM_Laserpointer**(*eOn=0*) [source]
[GeoCOM manual **p114**]

Turns on/off the laser pointer of the total station.

Parameters: **eOn** (*int*) – On (*1*) or Off (*0*)
Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

TMC – Theodolite Measurement and Calculation

GeoCom.**TMC_SetOrientation**() [source]
[GeoCOM manual **p148**]

Orientate the instrument in Hz direction. It is a combination of an angle measurement to get the Hz offset and afterwards setting the angle Hz offset in order to orientate onto a target.

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

`GeoCom.`**`TMC_DoMeasure`**(*cmd=1*, *mode=1*) [source]
[GeoCOM manual **p141**]

Carries out a distance measurement. Please note that this command does not output any values (distances). In order to get the values you have to use other measurement functions such as TMC_GetCoordinate , TMC_GetSimpleMea or TMC_GetAngle .

Parameters:
- **cmd** (*int*) – TMC measurement mode (see **p127** of GeoCOM ref manual)
- **mode** (*int*) – Inclination sensor measurement mode

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

`GeoCom.`**`TMC_SetEdmMode`**(*mode=6*) [source]
[GeoCOM manual **p167**]

Set the current measurement mode.

Parameters: **mode** (*int*) – measurement mode

Measurement modes available (check **p127-128** for further information) :

Format : *<int value>*. *<enum constant name>* : *<description>*

0. EDM_MODE_NOT_USED : Init value
1. EDM_SINGLE_TAPE : IR Standard Reflector Tape
2. EDM_SINGLE_STANDARD : IR Standard
3. EDM_SINGLE_FAST : IR Fast
4. EDM_SINGLE_LRANGE : LO Standard
5. EDM_SINGLE_SRANGE : RL Standard
6. EDM_CONT_STANDARD : Standard repeated measurement
7. EDM_CONT_DYNAMIC : IR Tacking
8. EDM_CONT_REFLESS : RL Tracking
9. EDM_CONT_FAST : Fast repeated measurement
10. EDM_AVERAGE_IR : IR Average
11. EDM_AVERAGE_SR : RL Average
12. EDM_AVERAGE_LR : LO Average
13. EDM_PRECISE_IR : IR Precise (TS30, TM30)
14. EDM_PRECISE_TAPE : IR Precise Reflector Tape (TS30, TM30)

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not

Return type: list

`GeoCom.`**`TMC_GetCoordinate`**(*WaitTime=100*, *mode=1*) [source]
[GeoCOM manual **p130**]

`GeoCom.`**`TMC_GetStation`**(*WaitTime=100*) [source]
[GeoCOM manual **p155**]

GeoCom.**TMC_GetSimpleMea**(*WaitTime=100*, *mode=1*) [source]
[GeoCOM manual **p132**]

Returns the angles and distance measurement data. This command does not issue a new distance measurement. A distance measurement has to be started in advance (call TMC_DoMeasure before this function). If no valid distance measurement is available and the distance measurement unit is not activated (by TMC_DoMeasure before the TMC_GetSimpleMea call) the angle measurement result is returned after the **WaitTime**.

Parameters:
- **WaitTime** (*int*) – Delay to wait for the distance measurement to finish [ms]
- **mode** (*int*) – Inclination sensor measurement mode

Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful
- error=1 if not. In this case RC can be equal to:
  - 1284 : Accuracy of the measurement could not be verified by the system of the total station
  - 1285 : Only the angles of the station could be obtained (no distance measurement available)

Return type: list

GeoCom.**TMC_QuickDist**() [source]
[GeoCOM manual **p138**]

GeoCom.**TMC_GetAngle**(*mode=1*) [source]
Refer to *TMC_GetAngle5* in GeoCOM manual **p136**

GeoCom.**TMC_GetEdmMode**() [source]

MOT – Motorization

GeoCom.**MOT_StartController**(*ControlMode=1*) [source]
[GeoCOM manual **p119**]

GeoCom.**MOT_StopController**(*Mode=0*) [source]
[GeoCOM manual **p120**]

GeoCom.**MOT_SetVelocity**(*Hz_speed*, *v_speed*) [source]
[GeoCOM manual **p121**]

BAP – Basic Applications

GeoCom.**BAP_GetTargetType**() [source]
[GeoCOM manual **p71**]

GeoCom.**BAP_SetTargetType**(*eTargetType=0*) [source]
[GeoCOM manual **p72**]

GeoCom.**BAP_SetPrismType**(*ePrismType*) [source]
[GeoCOM manual **p74**]

Sets the prism type for measurements with a reflector. Check **p69** of GeoCOM manual for all prism types. Prism types used with the main script:

- BAP_PRISM_360 (value=3) : Leica 360 Prism
- BAP_PRISM_360_MINI (value=7) : Leica Mini 360 Prism

Parameters: **PrismType** (*int*) – Constant associated to a prism type
Returns: [error, RC, []]

- error=0 and RC=0 if the request is successful

- error=1 if not

Return type: list

GeoCom.**BAP_GetPrismType**() [source]
[GeoCOM manual **p73**]

GeoCom.**BAP_SetMeasPrg**(*eMeasPrg*) [source]
[GeoCOM manual **p81**]

GeoCom.**BAP_MeasDistanceAngle**(*mode=6*) [source]
[GeoCOM manual **p82**]

GeoCom.**BAP_GetMeasPrg**() [source]
[GeoCOM manual **p80**]

GeoCom.**BAP_SearchTarget**(*bDummy=0*) [source]
[GeoCOM manual **p84**]

AUS – ALT User

GeoCom.**AUS_SetUserLockState**(*on=0*) [source]
[GeoCOM manual **p42**]

GeoCom.**AUS_SetUserAtrState**(*on=0*) [source]
[GeoCOM manual **p40**]

GeoCom.**AUS_GetUserLockState**() [source]
[GeoCOM manual **p41**]

GeoCom.**AUS_GetUserAtrState**() [source]
[GeoCOM manual **p39**]

GeoCom.**AUS_GetUserLockState**() [source]
[GeoCOM manual **p41**]

GeoCom.**AUS_GetUserAtrState**() [source]
[GeoCOM manual **p39**]

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**