KTH Elektro-
och systemteknik

Quadrocopter manual

for the Smart Mobility Lab

MATTEO VANIN

Stockholm January 29, 2014

Contents

Chapter 1

Introduction

This guide is intended to give an introduction to the hardware and software used for the quadrocopters[1] in the Smart Mobility Lab[2] of KTH - The Royal Institute of Technology, Stockholm. The purpose of this guide is allowing the user to get started with the quads in the SML, understand what their parts are and how to get them to work.

The chapters about hardware and software are partly based on the information available in [1]. Chapters 2 and 5 are a revised version of chapter 2 in [2]; if there are contradictions between this guide and [2], please refer to the former. A chapter about troubleshooting and the appendices follow.

If you want to build a new quad from scratch using the kits shipped by jDrones (`http://store.jdrones.com/`), follow the step by step guide provided in the Wiki section of [1]. If you already have a built quad and you want to get started, you should check the building guide as well and use the present document as a quick reference. In case you need more insightful information on a particular piece, we suggest you exploit datasheets and manuals for the specific piece, that can be easily found online.

Please note that throughout this guide we will be using [1] as a reference, since the hardware currently at disposal of the lab is described there. Nevertheless, that web page has been no longer updated or maintained since June 2013 and the project moved to `http://copter.ardupilot.com/`: if you wish to get new hardware or want insights on the Ardupilot state of the art you should check out this last page.

The steps you should follow to begin using a quad in the SML are the following; you can find guidelines and references for each of them in this guide.

1. Build the quad

[1] For the sake of brevity, in the sequel we shall also refer to the quadrocopter simply as *quad*.
[2] Also referred in the following as *SML*.

2. Install the firmware and set it up
3. Define the quad in the motion capture system
4. Set up the communication link
5. Set up safety measures to avoid damages to people and objects (hang up the nets, remove fragile objects from the workspace, ...)
6. Fly

1.1 Introduction to the resources

The main components of the SML test-bed are the quadrotor (agent), the Qualysis camera system, a PC running the motion capture software and a PC running a program that computes the control inputs and sends them to the quad.

As a preliminary overview, we summarize in the next lines and in figure 1.1 the representation of the testbed and how its various parts are put in communication and form a closed loop.[3]

In order to get the current position and orientation of the quadrotor, we exploit the Qualisys motion capture system. It consists of a set of twelve Oqus infrared cameras: after a calibration process that fixes the reference frame, they are used to capture the position of small retro-reflective markers that are placed on the quadrotor. The information coming from the cameras is sent via ethernet to a desktop computer running a proprietary software that computes the 3D position of the markers and provides the user 6 degrees of freedom data.

The control of the system is performed using LabView and Matlab. The reference signals computed by the control program (i.e. one byte for each of desired throttle, pitch, roll, yaw and flight mode) are sent via wifi to an actuator TMote Sky module onboard the quadrotor. Then these signals are transferred via serial to a serial adapter board, that in turn forwards the logic-level serial signals to the Pololu servo controller[4], that converts them to PWM signals. They, in turn, feed the Arduino CPU board containing two cascade PID loops that stabilize the quadrotor with the desired references. The outputs of the controller are finally delivered to the power distribution board that forwards those signals to the four speed controllers that set the input voltages of each brushless DC motor individually.

In the past, because of the unreliability of an old indoor localization system, IR sensors were mounted on the quad and another TMote Sky module was added to

[3]Please note that the current configuration of the motion capture cameras inside the SML is different from the one depicted in figure 1.1 (see also chapter 3).

[4]Detailed information about the serial board and the Pololu board can be found in the Smart Mobility Lab manual [3].

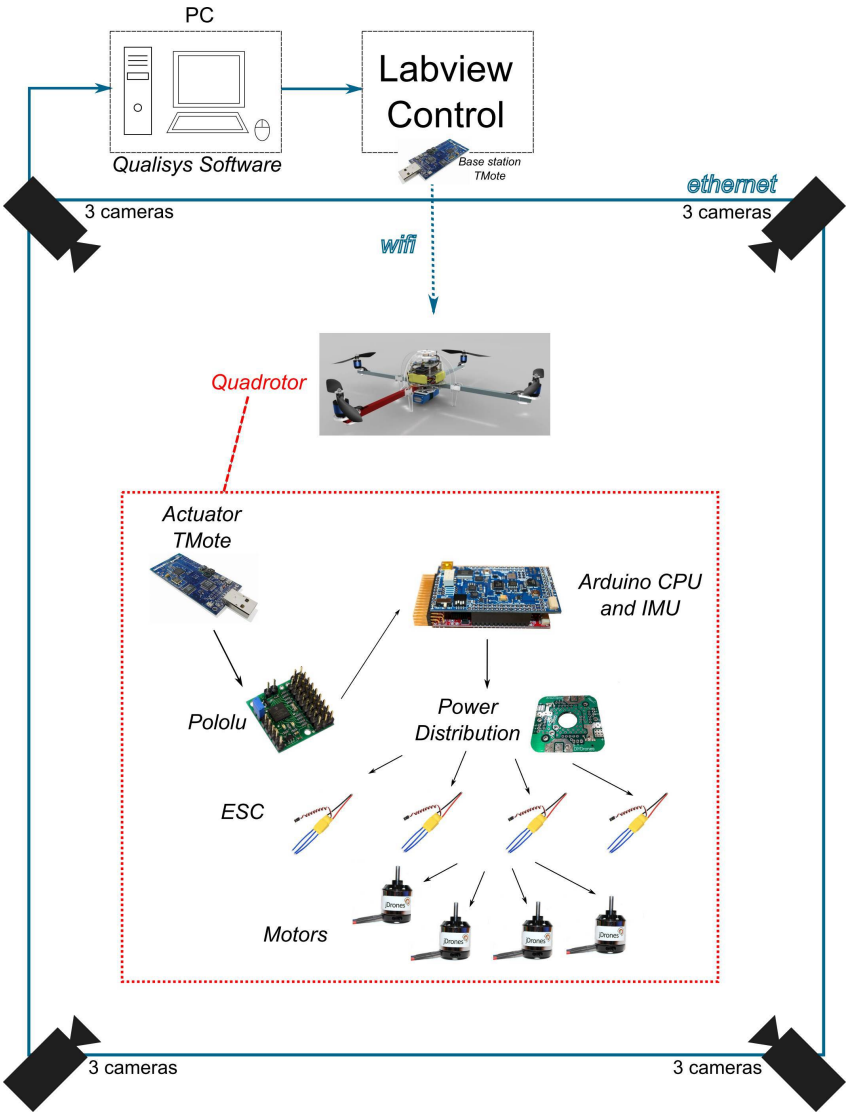


Figure 1.1: Testbed

send their measurements back to the control computer. For more details on this setting, please refer to this master thesis [4].

Chapter 2

Quadrocopter hardware

2.1 Principles of working

Here we present in an intuitive manner the basics that rule the flight of a quadrotor; these are basic concepts that do not require any further knowledge on the hardware or on the physical model, but give an appropriate understanding on how the vehicle works. We refer to figure 2.1 for a visual representation.

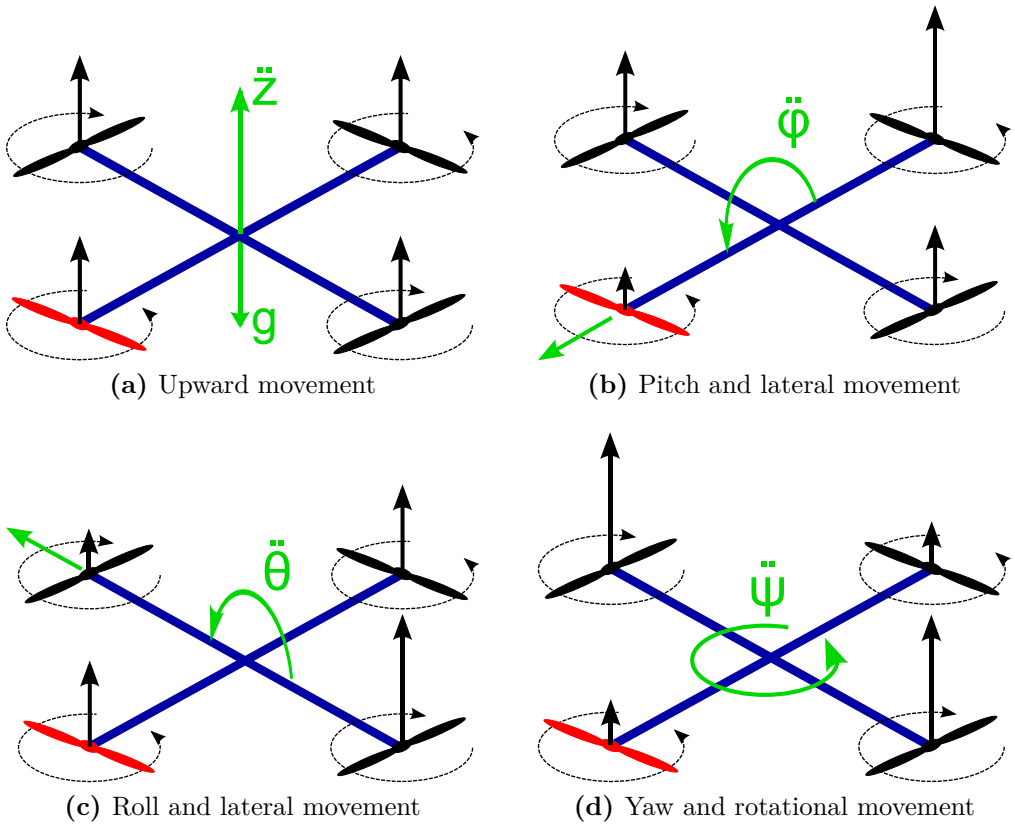
The four onboard motors can be controlled separately; each motor produces a torque that makes the rotors spin producing an upward thrust. Just like in single-rotor helicopters, the torque created by the motors would make the body of the quadrotor spin in the opposite direction of the rotors: to avoid this phenomenon two opposite rotors spin clockwise and the other two counterclockwise, so that the effect is balanced.

The upward movement is achieved when all the rotors spin at the same rate: in this case the total thrust vector is perpendicular to the ground and if its absolute value equals the gravity force acting on the quadrotor, the vehicle will hover at a fixed height.

The lateral movement of the quadrotor is performed by decreasing the rate of one rotor and increasing by the same amount the rate of the opposite one. In this scenario the quadrotor will tilt towards the direction of the slow propeller and as long as the total thrust do not variate, a lateral movement is achieved. This tilting movement is referred as *pitch* if the rotor that slows down is the front or the rear one, or *roll* if the rotor that slows down is a lateral one. However, since the vehicle is symmetrical with respect to its center of gravity, the choice of the orientation is non-influential to the dynamics.

Finally, the rotational movement[1], or yaw movement, can be obtained by slowing down the spinning rate of two opposite propellers and increasing the spinning rate

[1] around the vertical axis passing through the center of gravity of the craft.



(a) Upward movement **(b)** Pitch and lateral movement

(c) Roll and lateral movement **(d)** Yaw and rotational movement

Figure 2.1: Basic movements of a quadrotor. The front propeller is marked in red and the symbols z, ϕ, θ, ψ refer respectively to the vertical position and the pitch, roll and yaw angles.

of the others by the same amount (so that the total thrust vector is not influenced). In this way, the quadrotor will rotate in the same direction as the slow propellers, in fact the torque of the motors produces a rotation of the body in the opposite direction according to the third law of motion.

2.2 Jdrones ArduCopter

Arducopter is a platform for multicopters and helicopters that provides both manual remote control and an autonomous flight system equipped with mission planning, waypoints navigation and telemetry, managed via software from a ground station.

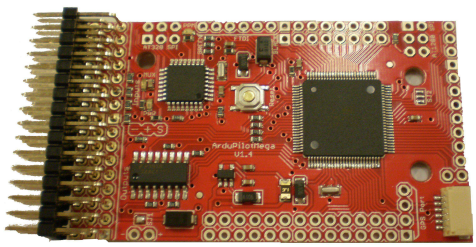
The quadrocopters in the SML have been assembled from the ArduPilot Mega kit, containing the Arduino controller board and the IMU board, and the JDrones ArduCopter kit, containing the frame, the motors, the speed controllers, the power

distribution board and the propellers. Detailed instruction on how to solder and assemble the parts of the kits are provided in the Wiki section of [1], we here present briefly how these components are connected and their role in the functioning of the quadrocopter. We suggest to follow the Wiki if you need to build a new craft.

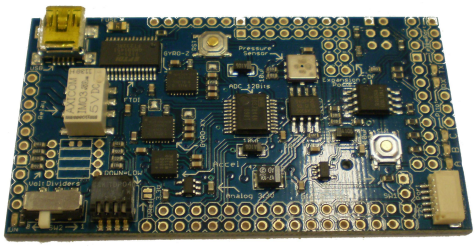
2.2.1 ArduPilot Mega

The core processing unit of the quadcopter consists in the ArduPilot Mega 1 board (figure 2.2). The firmware of this board is uploaded via usb using the `Mission Planner` software, downloadable from the Download section of [1]. See chapter 5 for details on the software.

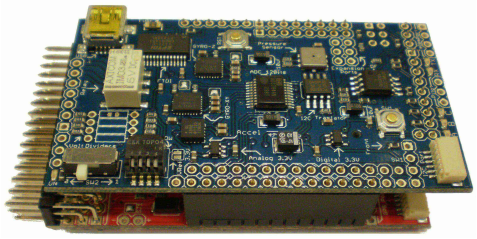
Two boards compose the ArduPilot Mega: the control board (red) and the IMU board (blue). In our testbed the control board is responsible for taking as input five remote signals in form of PWM waves (these signals correspond to throttle, pitch, roll, yaw and flight mode[2]) provided by the user and use them as references of the double cascade PID onboard controller. The control board is connected to the IMU board, that is equipped with a large number of sensors providing readings that are exploited by the controller board or can be used for other applications; in our setup the exploited sensors are an accelerometer and a gyroscope, other sensors are a GPS module (not connected), a magnetometer (soldered but not used), pressure and temperature sensors.



(a) *Control board.*



(b) *IMU board.*



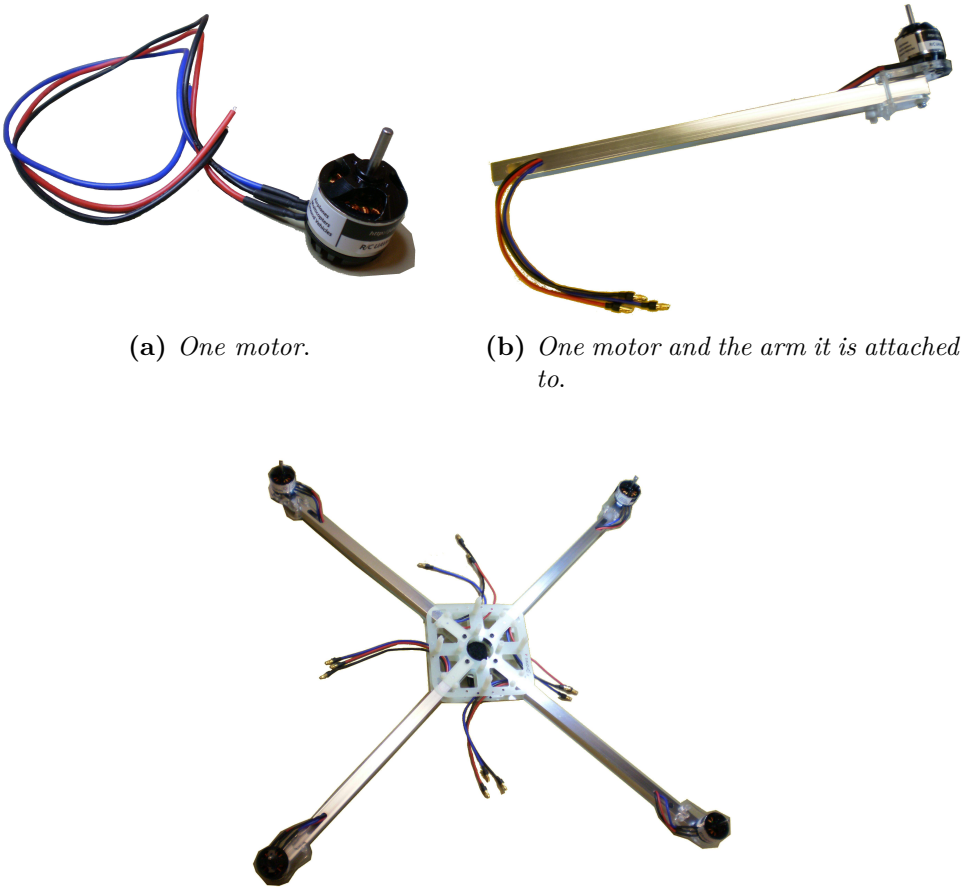
(c) *ArduPilot Mega soldered.*

Figure 2.2: ArduPilot Mega

[2]In our implementation we do not actually exploit the *flight mode* signal, so the inputs of interest are actually four, see chapter 4 for further details on this.

2.2.2 Arducopter frame, power distribution and motors

The frame of the quadrocopter is basically made of plastic pieces composing the structure and the protection for the electronics, and four aluminium bars (i.e. the arms of the quadrocopter, that are full-holes and 28 cm long) that support the four brushless DC motors and contain the wires from the motors to the speed controllers (see figure 2.3).



(a) *One motor.*

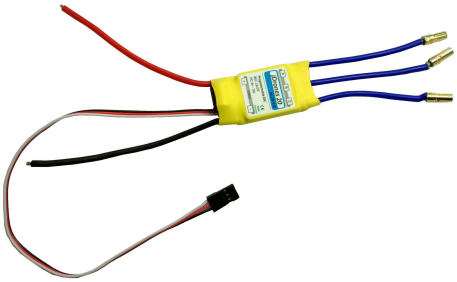
(b) *One motor and the arm it is attached to.*

(c) *The four arms with the plastic structure that keeps them together.*

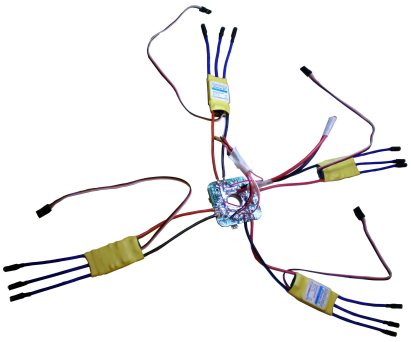
Figure 2.3: Motors and arms.

The electronic speed controllers (ESC) are soldered to the power distribution board and they get from it both the power and the signal that is generated by the ArduPilot board (figure 2.4).

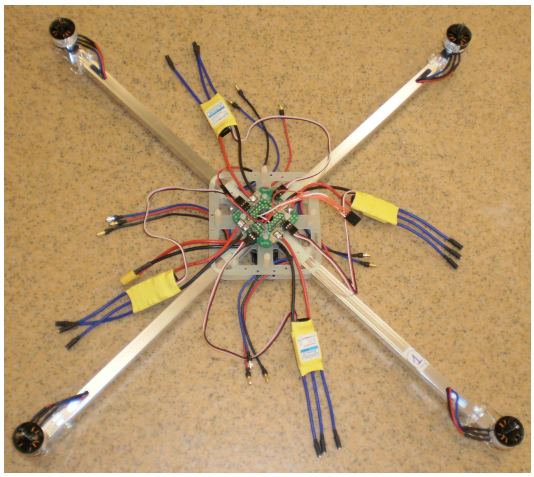
The ArduPilot lays on the top of the power distribution board with its own structure and an external frame protects the vehicle and makes it stand still. The



(a) *One ESC.*



(b) *Power distribution board attached to the ESC's.*



(c) *Quadrotor structure with motors, ESC's and power distribution board.*

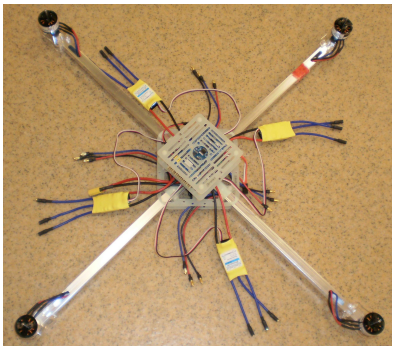
Figure 2.4: Esc and power distribution.

final configuration includes the propellers, that are fixed to the motors, and the receiver of the remote signal. A polystyrene structure has been added as base of the quadrocopter because the plastic legs are indeed very fragile and do not stand aggressive landings (see figure 2.5).

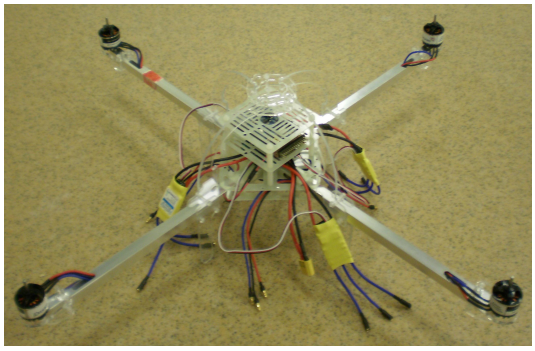
The propellers are plastic blades that are sold in pairs, the model of the ones we use is 10x4.5[3]. The front/rear propellers are different from the left/right ones, the front and back must rotate clockwise (*pusher propellers*, marked with the letter *R* after the size label) and the others counter-clockwise (*puller propellers*) (see figure 2.6).

The battery that powers the quadrocopter is a 3 cells Li-Po battery (see fig-

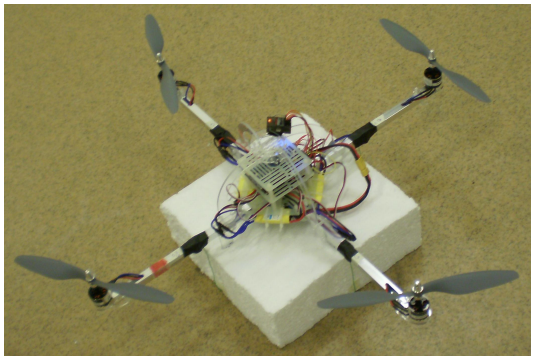
[3]10 and 4.5 represent the diameter and pitch of the propeller respectively.



(a) *Positioning of the ArduPilot board in the frame.*



(b) *Complete frame with protections and legs.*



(c) *Final structure, with propellers, RC receiver and polystyrene base.*

Figure 2.5: Frame and final configuration.

Figure 2.6: Propellers of the quadrocopter, puller on the top and pusher on the bottom.

ure 2.7) that provides 11.1V as output voltage and 2200 mAh as rated capacity. The average time of flight before the battery discharges is from 5 to 10 minutes, depending on the throttle that is applied by the motors.

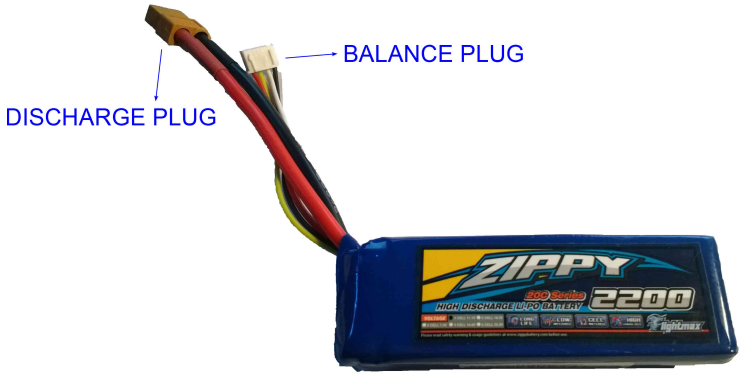


Figure 2.7: Li-Po battery of the quadrocopter.

Charging the battery

In order to charge the batteries:

1. Power the charger.
2. Connect the discharge plug and the balance plug of the battery to the charger.

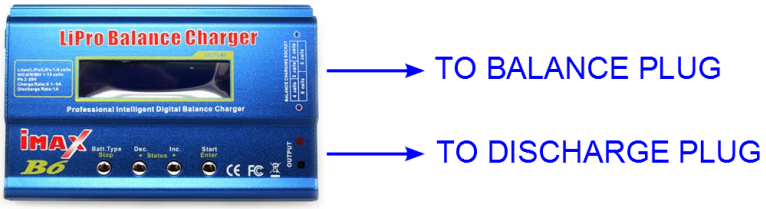


Figure 2.8: Battery charger.

3. Press `Batt.Type` until `LiPo BATT` is selected and confirm with `Start/Enter`.
4. Set the correct charging parameters, 2.2 A and 11.1 V are ok for the batteries we use. If you need to change these parameters, press `Start/Enter` to make them blink and increase/decrease (`In`/`Dec` keys). Do not charge the batteries at a higher rate than the charge rate, e.g. 2200 mAh batteries should not be charged at more than 2.2 A.
5. Insert the battery into the safety bag.

Figure 2.9: Safety bag.

6. Keep `Start/Enter` until you heard a sound. Press it again to confirm.
7. Never leave the battery unattended while it's charging. Otherwise *this* can happen.

Chapter 3

Qualysis motion capture system

(a) Qualisys logo. **(b)** QTM logo.

Figure 3.1: Qualysis and Qualisys Track Manager logos.

The motion capture (mocap) system produced by Qualisys[1] consists in twelve IR cameras, each of them equipped with an infrared flash. The light of the flashes is reflected by small reflective balls (*markers*) that are placed in advance on the objects we are interested in tracking, and the cameras capture these reflected beams and compute relative position and size of the markers. The information collected by every camera is transmitted via ethernet to a computer running the Qualisys Tracking Manager (QTM) software which in turn computes the 3D position of the markers. If a group of markers have been previously gathered by the user forming a *body*, then the software is able to track not just the position of the markers, but also the 3D position and orientation of the body when it moves in the workspace.

In order for the cameras to accurately compute the position of the markers, a calibration process has to be carried out before data collection. This is performed by placing an L-shaped tool representing the reference system on the ground, and, after starting the procedure in the QTM software, carrying around a calibration wand and moving it in all directions for a preset time (see the tools in figure 3.2).

[1] `www.qualisys.com`

Figure 3.2: Wand and reference system (`www.qualysis.com`).

The current setting of the cameras is: eight cameras are hung in groups of two on the corners of the room, two cameras are hung near the windows on opposite sides of the room, one camera lays above the door and one next to the big projector in the center of the room. Ten cameras are Oqus 4 and the two on the sides are Oqus 3+ (namely cameras nr. 2 and 10). These last have improved functions, such as light sensitivity, image and high-speed video mode; for the full specifications see table 3.1.

Figure 3.3: Group of three Oqus cameras (old placement).

Camera	Normal mode (full FOV)		High Speed mode (full FOV)		Max fps (reduced FOV)
Oqus 4	3 MP	480 fps	n/a		10000 fps
Oqus 3+	1.3 MP	500 fps	0.3 MP	1750 fps	10000 fps

Table 3.1: Specifications of the Oqus cameras (`www.qualisys.com`). `FOV` stands for Field Of View and `fps` for frames per second.

In figures 3.4 and 3.5 we show two screenshoots from the QTM software, both are images from the point of view of one of the Oqus 3+ cameras[2] framing the quadrocopter: in the first one we can see the five markers that have been placed on the quadrocopter, in the second one we overlap the video recorded by the camera and the 3D visualization of the body and the reference system.

The 3D data[3] obtained by the Qualisys software can be requested from other computers connected to the PC that runs the QTM proprietary software via a TCP/IP connection and exploiting the libraries that Qualisys provides for NI Labview and Matlab. A library[4] for Python has been created by the author and works both in Windows and in Linux. For the experiments in the testbed, the frequency of the cameras has been set to 100[Hz]; this is ten times the speed of the control loop currently used for the quadrocopter. What limits the possible control frequency is the communication link.

The performances of the Qualisys system are really good as far as the precision is concerned, so there should be no need to filter the measurements to get more reliable information on position and orientation.

[2] Not in the current placement.
[3] 3D position, pitch, roll and yaw of the body.
[4] Available at [5].

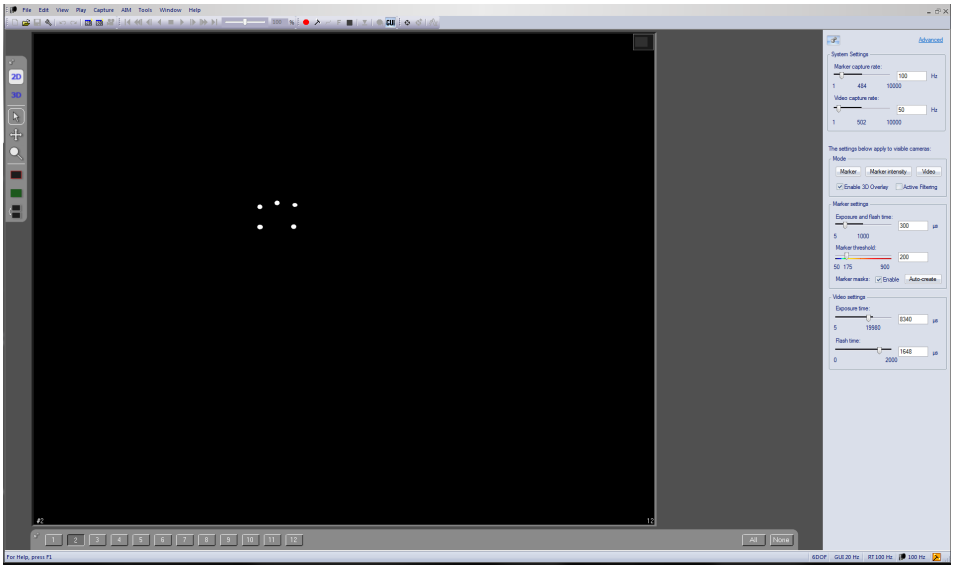


Figure 3.4: Snapshot of the markers viewed by an Oqus 3+ camera.

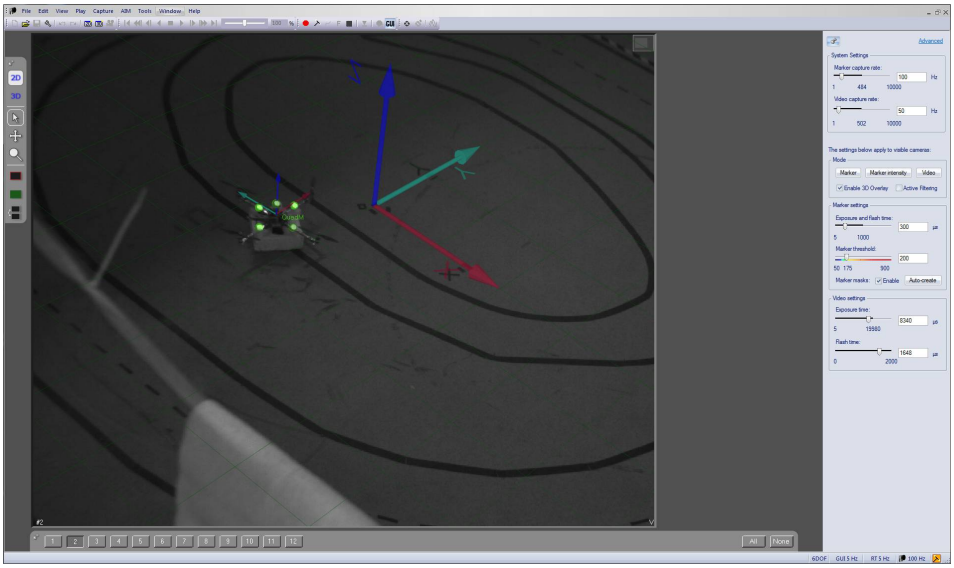


Figure 3.5: Snapshot of the video capturing of an Oqus 3+ camera, overlayed with the quadrocopter body and the reference system.

3.1 Body definition in the QTM software

Let us now describe the procedure to define a new *body* in the QTM, that is necessary if a new object must be tracked by the software. If you have not done it yet, place at least four retro-reflective markers[5] on the body in a non-symmetrical fashion and place the object inside the field of view of the cameras.

The fastest method to define a new body is:

0. *Not necessary but simplifies the final steps*: Place your body in the reference system already with the correct position and orientation. That is, place it in such a way that the point you want to be the center of the body-fixed reference system overlaps with the center of the earth-fixed reference system. Also make sure that the axis that you want to be the *x-axis* (*y-axis*) of the body is aligned with the *x-axis* (*y-axis*) of the earth-fixed reference system. The earth-fixed reference system is marked with tape on the floor and can be seen in the QTM 3D measurement window.

1. Start the QTM and open the project in which you wish the body to be saved (or a new project).

2. Start a new measurement (`CTRL+N`).

3. Start Capture (`CTRL+M`). You can set a very small capture time such as 1 s and press `Start`.

4. When the capture is over you will see in the 3D view window the markers that you placed on the body.

5. Select them by holding `SHIFT` while clicking on them .

6. Drag and drop your selection into the *Labeled trajectories* frame (or just press `L`).

7. On the *Labeled trajectories* frame you will now see your markers listed as "New" followed by four digits. You should see all of the markers you placed on the body with a 100% or anyway very high "Fill level"; if apart from these you also see markers with a very low "Fill level" just discard these last ones. Select all the good ones holding `CTRL`, right click on them and click on `Define rigid body (6DOF)`.

[5] Actually three markers would be enough to define the body, but that would not be a robust solution since if for some reason one of the markers is not seen by the cameras then the tracking is lost.

8. Enter a name for the body you are defining (e.g. "Arducopter1") and press `OK`.

9. You can now close the measurement (without saving it) and open a new one. In the 3D view window you will see your new body's reference system overlapping the fixed one.

10. Open `Tools` → `Project options` and switch to "6DOF Tracking". You will see your newly created body in the list; if you followed step 0, your body-fixed reference system should be correct and you can go directly to step 13.

11. Left-click on the body to select it and press `Translate`. You can now select "To the geometric center of the body (the average of the body points)" or set manually where the body's center will be by using "So that point ... in the fixed body has local coordinates ... " and adjusting progressively if needed.

12. In order to rotate the reference system, click on `Rotate` and use the option that suits best your needs. Note that if you use the "Align the body using its points" option, the index of a point/marker is the index of the point listed in the *Labeled trajectories* frame, being 1 the first index for each of the bodies.

13. Apply your settings and as you move the body inside the workplace you should see its reference system moving accordingly in the 3D window. Check the values of x, y, z, roll, pitch and yaw in the QTM to see if they are correct.

Chapter 4

Communication link

The wireless communication between the controller PC and the quadrocopter is made using a pair of Tmote Sky devices. Tmote Sky is an ultra low power wireless module for use in sensor networks, monitoring applications, and rapid application prototyping. Figure 4.1 shows the components of a mote. For further insights on Tmote Sky, check the SML general manual [3], the datasheet [6] and the kth-wsn webpage [7].

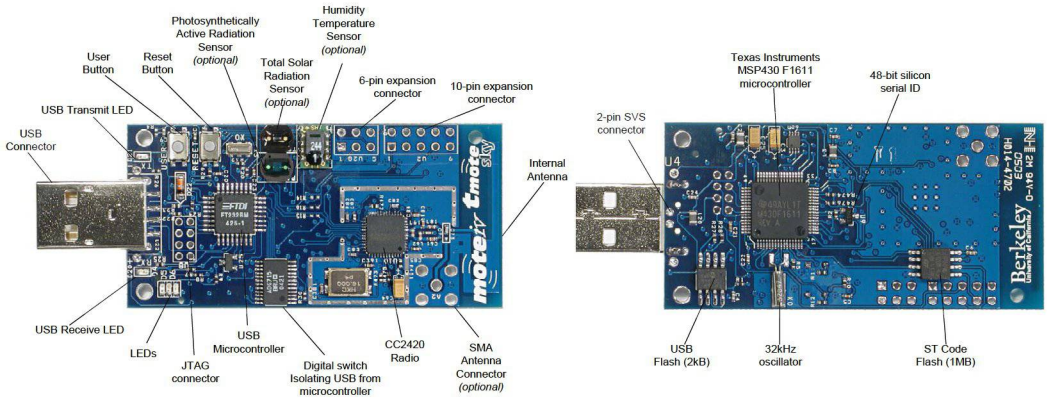


Figure 4.1: Tmote Sky components.

One mote is plugged into an USB port of the controller computer and the other one is attached to the quadrocopter in the manner we described in section 1.1. They have been programmed using TinyOS, exploiting the code available in the Smart Mobility Lab repository [5] inside `Misc/TinyOS_Code/Quad_TinyOS`; refer to the manuals in [7] for the procedure for programming the motes.

The scheme of the communication link is depicted in figure 4.2, we here summarize it and refer once again to the laboratory manual [3] for a more detailed description:

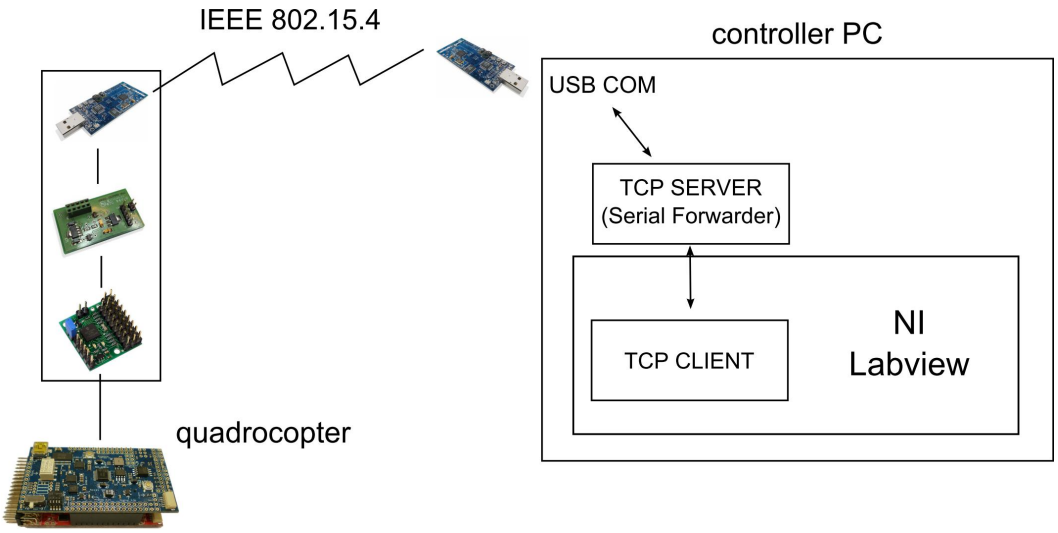


Figure 4.2: Communication link.

- A C-based serial forwarder is used to send data from the computer to the PC mote: the serial forwarder creates a TCP server process in the computer that listens to the local port we specify and forwards the received data to the PC mote.

- The Labview program creates a TCP client process that connects to the TCP server, therefore creating a TCP connection. The data we need to send are thus forwarded to the serial forwarder via this TCP connection and in turn they are received by the PC mote;

- The motes use the IEEE 802.15.4 protocol to send data wirelessly from the PC one (base) to the quadrocopter one (actuator). Please note that the pair of motes need to be programmed on the same channel and they should be the only ones using this channel in the working environment, in order to avoid packet collisions;

- The data received by the actuator mote are forwarded via serial to the serial board and the Pololu board and finally reach the Arduino board of the quadrocopter.

4.1 Sending data

In order to connect and send data to the motes in LabVIEW, we deploy the functions `OpenSF`, `WriteSF` and `CloseSF` available in `Misc/TinyOS_Code/Quad_LabVIEW` at [5].

Details on how to setup the connection can be retrieved in the SML manual [3], we here want to explain what the signals that we send to the quad through the motes are. In the current setup, the inputs of the `WriteSF` function are:

- `Connection ID`: the tcp network connection
- `Destination ID`: an unsigned word representing the ID of the mote to which we are sending
- `Servo Positions`: a cluster of 8 unsigned bytes that represent the control signals that we send to the quad

Let us focus here on the last item. The range of the 8 signals we pass as input of `WriteSF` is currently[1] from 0 to 127 and the middle value 64 represents a null input. These signals are encoded by the LabVIEW function in order to be received by the program on the sending mote. Once the information is received and decoded by the actuator mote, the serial board takes care of transmitting the eight signals in succession to the Pololu board. Here they are encoded as PWM signals and are outputted from the Pololu board as separate channels. Only the first five channels are plugged to the Arduino board in the current settings (see appendix B for details on the wirings on the quad), so it does not make any difference which values are written in the last three bytes of the cluster in LabVIEW.

The first five inputs represent the throttle, pitch, roll, yaw and flight mode. Namely, the throttle input is related to the total thrust applied by the motors. The roll and pitch inputs are in form of target angles for the onboard controller, so those inputs are directly translated to an angular reference from $-45°$ to $45°$. The yaw input is actually a rate, not an angle, and is translated to a rate reference from $-45°/s$ to $45°/s$. So giving constant non-null ($\neq 64$) yaw input, you will get a constant rotation of the arms-plane of the quad. See *this* picture for details. As far as the fifth input is concerned, we usually set the Arduino board in such a way that its value does not count either: see chapter 5 about this.

4.2 Mote troubleshooting

Here are some hints to follow if there are problems in the communication between the motes.

4.2.1 The motes do not seem to communicate

The usual procedure to create a serial forwarder process and use it in LabVIEW is

[1] This can be changed by modifying the TinyOS program running on the motes.

1. Plug the motes, one in the PC, the other on the serial board on the quad (the battery on the quad must be plugged for the mote to be powered).
2. Open a Cygwin (in Windows) or a terminal (in Unix) and type the `motelist` command.[2]
3. You will get a list of the motes connected to the PC with the name of the mote and the COM port they are connected to, e.g. MFT4LX0A, COM8.
4. In the LabVIEW program, set the `Port number` input of the `OpenSF` function, e.g to 9003.
5. The command to open the serial forwarder will be `sf 9003 COM8 telosb`[3]. If you are using Cygwin you will get a warning after entering the command, this is normal

When the motes communicate correctly between each other, you should see the red and yellow leds blinking on them each time they send and receive data. If you see both the motes blinking but still you cannot perform operations on the quad, e.g. you cannot arm, check chapter 6 for solutions.

If instead you cannot get the mote to communicate to each other at all:

- Make sure that the motes have the correct programs on them[4], they can be found in the code in [5]. One of the motes must be programmed with the code in `PC_mote`, the other one with the code in `Quad_mote`. Also note that they have to be programmed on the same channel and must be the only ones sharing that channel in the test-bed. The procedures to program the motes are explained in details in [3] and in the manuals of [7].
- Make sure that the mote with the `PC_mote` program is the one attached to the PC, and the one with the `Quad_mote` program is the one attached to the quad.
- Especially if you have more than one mote plugged in the PC, make sure you are forwarding data (e.g. from LabVIEW) to the correct mote, i.e. the one associated with the mote on the quad. Check the `Port number` input of the `OpenSF` function, it must be the same as the parameter you set in the `sf` command. Check [3] for more info.
- Check the Cygwyn/terminal window to see if you got a *"write failed"* error, if this is the case see next paragraph, 4.2.2.

[2]The programs `motelist` and `sf` must be installed before the corresponding command can work. Check the manuals in [7] about that.

[3]You can write either `telosb` or `115200`, both work.

[4]Actually, rather than "make sure", just reprogram them with the correct code, it's a really fast operation.

4.2.2 I get *"write failed"* on the Cygwin/terminal window

This means that there is some hardware problem on the connection. Either the control loop is too fast and the hardware cannot keep up with that (if this is the case, you will get *"write failed"* every time you try to send signals at that particular rate): in this case you need to slow the control loop down. Or there has been some fail on the USB connection: if this is the case, stop the program, disconnect the mote from the USB port and try again.

Unluckily, this last problem can be critical, because if your quad is flying and all of a sudden the connection breaks and causes a *"write failed"* error, you will have no possibility to stop the quad remotely and it will keep on maintaining the inputs that it had before the connection failed, leading to possible crashes.

Chapter 5

Mission Planner Software

The Mission Planner Software by Michael Oborne, available *here*, is used to upload the Arducopter firmware to the board, update it and perform some tuning operations. It has a lot of features in addition to the ones we currently use it for, above all for outdoor flight.

Figure 5.1: Screenshot of APM planner software.

Firmware upload. In order to upload a new firmware to the board, open the software and connect the board to an USB port of the PC. Don't press the `CONNECT` button on the Mission Planner window yet. Switch to the tab `INITIAL SETUP` → `Install Firmware`, click on the picture above `ArduCopter Vx.x Quad` (figure 5.2) and confirm: the software will download the latest firmware from internet and upload it to the board.

Figure 5.2: ArduCopter Vx.x Quad.

5.1 Preliminary operations

Before flight, some procedures must be carried out in order to set the quadrocopter in a proper way, exploiting the Mission Planner (MP) software. We will list them here, referring once again to [1] for further details. Make sure the battery is connected if you need to use the mote, otherwise it will not be powered, and **remove the propellers prior to these operations, as a safety measure**.

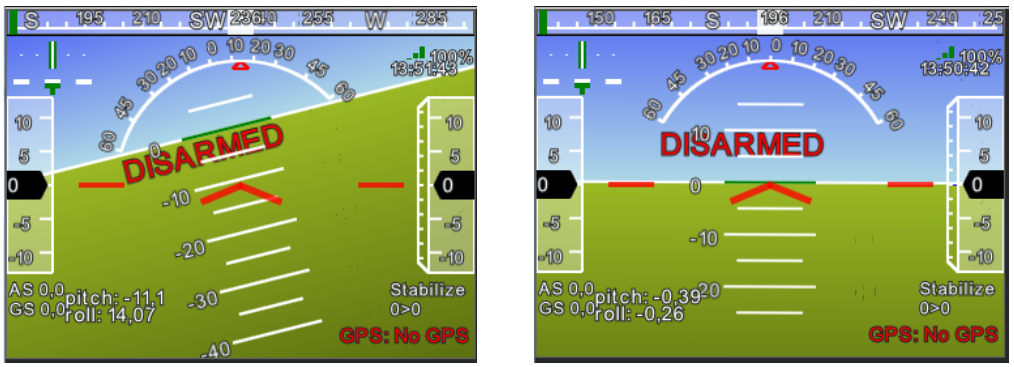
Radio range calibration. This operation has to be done before the first flight and whenever the quadrocopter seems not to respond accordingly to the input commands. It consists in setting the end points of each of the radio signals that feed the APM board and is done in the `INITIAL SETUP` tab of the APM software: select `Mandatory Hardware` → `Radio Calibration`. After you press `Calibrate Radio` you will need to send the minimum and maximum values (send 0 and 127 respectively if you are using the motes) for each of the first five channels.

Fliying mode. In the `INITIAL SETUP` tab, go to `Mandatory Hardware` → `Flight Modes`. Here you can select the flight modes associated with the range of input you send to channel 5. We usually set all the flight modes to `Stabilize`, so it actually does not matter which values we send to the fifth channel, the flight mode will always be `Stabilize`; you might as well experiment the other modes, among which the acrobatic mode `Acro`. Just remember that the motors will not arm unless the mode is set to `Stabilize` or `Simple`.

Frame type. Make sure you set the frame type ("Plus"/"X") that corresponds to your configuration in `INITIAL SETUP` → `Mandatory Hardware` → `Frame Type`.

Quadrocopter level. This operation has also to be done before the first flight and whenever the quadrocopter seems not to respond accordingly to the input commands. You can see if your board has a correct calibration from the main window of the MP software:

- Connect the board to the PC, open the MP software and press the `CONNECT` button on the top-right corner.
- On the `INITIAL SETUP` tab, right click on the ground-sky frame (the one on the top-left) and press `User Items`; you can select pitch and roll so you will visualize their values on the frame itself.
- Put the board on a flat surface, and check if the roll and pitch have close-to-zero values (see figure 5.3). If they do not, you should do the `Quadrocopter level` procedure.



(a) Bad calibration. **(b)** Good calibration.

Figure 5.3: Examples of how the ground-sky frame of the MP can appear when the board is placed on a flat surface.

The procedure consists in calibrating the accelerometer of the APM board and this is performed automatically by the software (`Mandatory Hardware` → `Accel Calibration` section of the `INITIAL SETUP` tab): the user only needs to orientate the APM accordingly to the onscreen instructions and keep it still while the software records the data it needs.

Esc calibration. This operation has to be done right after building the quadrocopter and whenever we suspect that the speed of the motors are not correctly balanced. It aims to make the speed controllers react in the same fashion to the APM and to the RC commands. Automatic calibration is the easiest way to perform it, since it is just necessary to hear the tones emitted by the APM and move the remote sticks[1] as a consequence.

Onboard PIDs. If you want to modify the parameters of the onboard PID controllers, you can do that under `CONFIG/TUNING` → `Basic Pids` and `APM:Copter Pids`

[1] or changing the input values if we are using remote Labview control.

Chapter 6

Troubleshooting

Here follows a list of possible problems that can arise when using the quad, with suggestions on how to approach them.

Problem / Question	Possible solutions / Answers
The board and the communication components (*i.e.* the mote and the Pololu board) have no lights on	Check if a charged battery is plugged into the quad.
The quad does not arm.	1. Check if a charged battery is plugged into the quad. 2. Are you using the correct arming sequence? That is, null throttle, middle value of pitch and roll, most-right value for yaw[1]. The fifth input does not matter in the current configuration[2], the sixth, seventh and eight inputs never matter (they are not connected to the Arduino board). 3. Check the Mote troubleshooting (4.2). 4. Recalibrate the radio inputs (rare; it can be necessary after a firmware update).
The propellers spin, but the quad does not lift even when the throttle input is high.	The battery is dead: recharge it.

Continues on the next page

[1] In the current configuration, these inputs correspond to the sequence 0 64 64 127.
[2] i.e., if you set all the flight modes of the fifth channel to `Stabilize`. See chapter 5 for details.

Continued from the previous page

Problem / Question	Possible solutions / Answers
The quad flips as soon as it takes off.	1. Check if the propellers have been mounted properly: • are they all well-screwed? • the front&back propellers must have an "R" mark on them (*pusher* propellers); the left&right propellers must not (*puller* propellers)[3]. All the propellers must be mounted with the marks facing upward, anyway. 2. If the wires from the ESC to the motors have been changed since the last time the quad worked, reconnect them in the proper way.[4] 3. Specially after a crash, a motor may have broken and cannot spin as fast as it should. Remove the propellers and check if this is the case: if so, replace the motor.
Which axis is related to pitch and which to roll? And which movement corresponds e.g. to a positive pitch input?	A pitch input produces an inclination of the quad towards the front or back motor; a roll input produces an inclination towards the left or right motor. The direction of the tilting depends on how the quad is configured; the easiest way to verify if a positive pitch leads to an inclination towards the front or back motor (and similarly for the roll) is to put the quad on the ground (better if with a polystyrene base), arm it and then keeping the throttle input quite low (you don't want the quad to take off) give a positive pitch input to see how it tilts.
The inputs seem to be unbalanced: i.e. the zero values of the pitch and roll inputs seem to produce a non-zero output.	1. Do the "quadrocopter level" procedure (see 5.1). 2. If after that the problem is still present (it shouldn't), add an offset to the signals produced by your controller in order to compensate.

Continues on the next page

[3]See `https://code.google.com/p/arducopter/wiki/AC2_Props`
[4]See appendix B.

Continued from the previous page

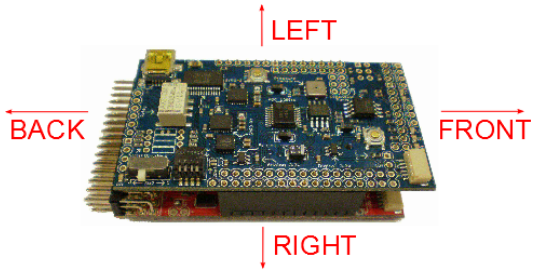
Problem / Question	Possible solutions / Answers
Even if the quad is hovering steadily, it keeps a static position offset wrt the reference I set, and nothing seems to be wrong with my controller.	This might be due to imprecisions of the onboard stabilization system. A not very neat but effective solution is to compensate this offset inserting an inverse offset on the measurements you get from the mocap system.
I do not receive the correct position/orientation of the quad from the mocap (e.g. using the Qualisys libraries in LabVIEW).	1. If you receive NaN: the quad may be outside the field of view of the cameras (place it closer to the center of the room). 2. If you receive 0: you are not using the correct body number (e.g. as input of the *Q6D Euler* block in LabVIEW), you can find the correct number by counting down to the quad's body in the *labeled trajectories* view of the QTM window. If the quad is not in the list, read the next points. 3. Are the cameras in *video mode*? If so, put them back to *markers mode*. 4. Does the quad have any marker at all on it? If not, fix at least 4 markers in a non-symmetrical fashion on the quad[5] and do the *body definition* procedure (see section 3.1). 5. Are the markers on the quad seen by the QTM system as you place the quad on the floor? If not, the software may be stuck: close and reopen the measurement window (`Esc` and then `CTRL+N`). Unless you messed up with the exposure time of the cameras[6], you should now be able to see them: go on with the *body definition* procedure (see section 3.1). 6. If the markers are on the quad and you can see them on the QTM window, but the quad is not listed as a body, then the body is not correctly defined: the markers may have moved or the quad has never been defined in the configuration file you are using. Do the *body definition* procedure (see section 3.1).

Continues on the next page

[5] e.g 4 on the arms at difference distance from the center and one on the top.

[6] We refer to the Mocap manual for further details about the working of the QTM system.

Continued from the previous page

Problem / Question	Possible solutions / Answers
Why is there polystyrene attached to the quad's bottom?	It was added to limit damages as a consequence of crashes or unsmooth landings. It is not necessary to keep it if the controller works fine (please note, though, that a good controller cannot avoid the problem presented in 4.2.2), but the controller's parameters may need to be changed after the removal, if they depend on the quad's mass and air drag.
The arms and motor all look the same. Which one is the front one, for instance?	If the quad is built in the proper way, the Ardupilot board side with the leds points forwards and identifies the front motor/arm, the board side in which the wires are plugged identifies the back motor/arm (see the figure below).[7] 
Where can you get spare pieces for the quad?	The parts that need to be replaced more often are definitely the propellers: they break quite easily even after minor crashes. The arms, plastic frame and motors follow, but from a distance. Drawer L2.1 in the SML contains many spare pieces. New ones can be bought from `www.buildyourowndrone.co.uk/`, check appendix A for the specific links.
The quad appears upside-down in the QTM, and so I get values close to 180° for the pitch (or the roll).	Probably the configuration of the markers on the quad is "too symmetrical", and the software has troubles to state if the z-axis fixed to the quad is pointing downwards or upwards. Re-place the markers trying to have as less symmetry as possible (wrt to center of the reference system) and do the *body definition* procedure (see section 3.1).

Concluded from the previous page

For more FAQs, check the troubleshooting in the arducopter project page *here*.

[7] The board is oriented in this fashion in a "Plus" configuration: if you are building a new quad and wish to use a "X" configuration instead, follow the instructions in [1].

Appendix A

Where to buy parts

Here you find some links to the online shops where quad parts can been bought. Double check that the piece you order is what you actually need.

Item	**Link to online shop**
Propellers	`www.buildyourowndrone.co.uk/Propeller-set-10x45-EPP-Style-Composite-p/ac-psc10x45.htm`
Arms	`http://www.buildyourowndrone.co.uk/ArduCopter-Quad-Arm-28cm-Full-Holes-p/ac-arm-28fh.htm`
Motors	`http://www.buildyourowndrone.co.uk/850Kv-Motor-AC2830-358-p/mot-85kv.htm`

If you need to replace other parts, you can find them at `http://www.buildyourowndrone.co.uk/`.

Appendix B

Quick reference for wire connections

In figure B.1 we illustrate how the parts of the quad are connected to each other; red arrows indicate a power flow, and blue arrows a signal flow.

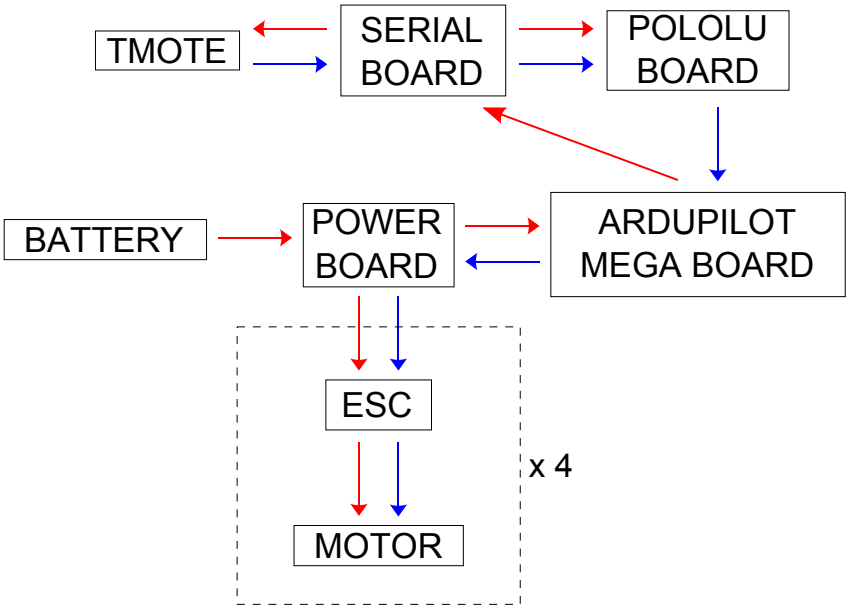


Figure B.1: Wire connections on the quad. Red arrows indicate a power flow, and blue arrows a signal flow.

In the following we will report pictures and explanations for each of the connections between the parts of the quad.

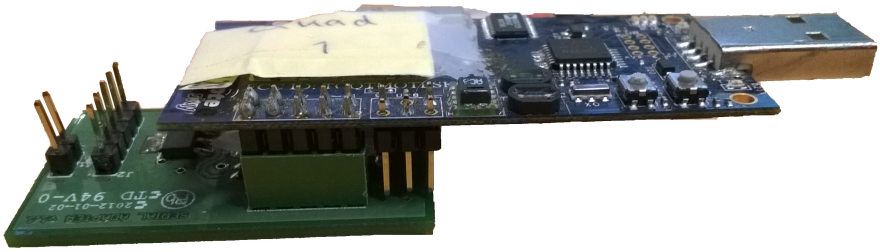
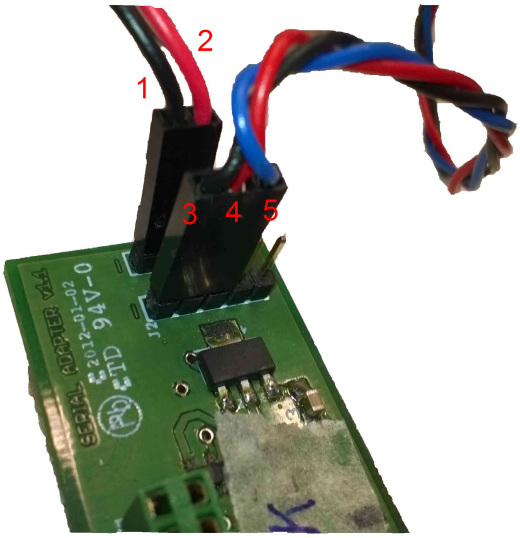
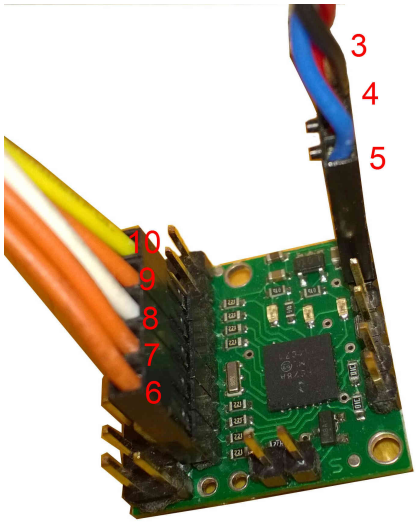


Figure B.2: Connection TMote - Serial board

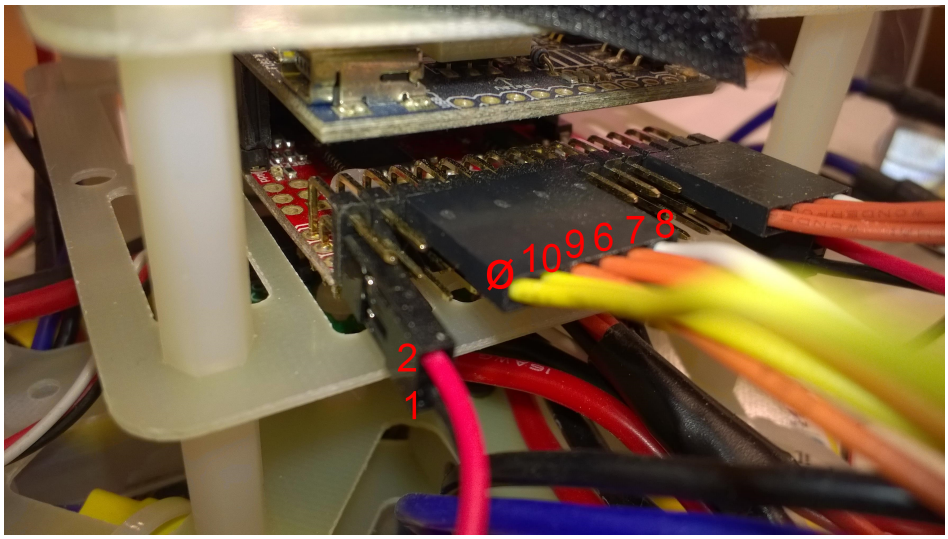


(a) Serial board connections.

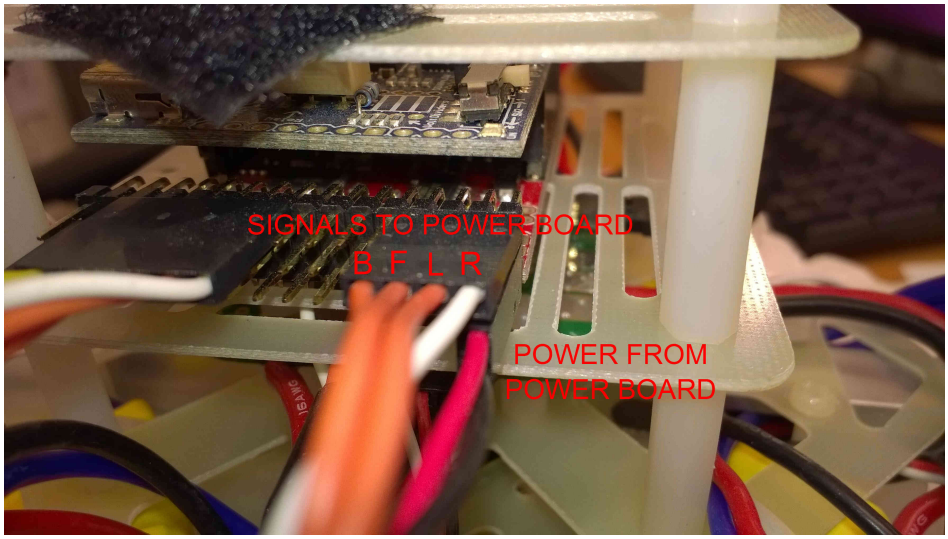


(b) Pololu board connections.

Figure B.3: Connections on the Serial board and Pololu board.



(a) Arducopter board connections S- left view.



(b) Arducopter board connections - right view.

Figure B.4: Connections on the Arducopter board.

For connections from - to the power board we refer to `https://code.google.com/p/arducopter/wiki/AC2PowerDistro`: a very clear visual explanation is provided there.

As far as the ESC-to-motor connection is concerned, there are three cables coming out of each ESC and three of each motor[1]. So, there are six possible ways of connecting the cables: three of these will make the motor spin clockwise, the others counter-clockwise[2]. Basically you can plug them randomly, and if the rotation is not in the right sense just switch two of them.

A configuration that should work is:

- For clockwise rotation (front and back propellers), plug red-motor to black-esc and black-motor to red-esc.
- For counter-clockwise rotation (left and right propellers), plug red-motor to red-esc and black-motor to black-esc.

[1] If you need to install a new motor, you will need to solder the motor connectors. There are several online guides on how to do this, for example `http://www.youtube.com/watch?v=B9yY9Kk4bEA`.

[2] See `http://www.helifreak.com/archive/index.php/t-145841.html` for a clarification about this.

Bibliography

[1] *Google code page for the arducopter project.* [Online]. Available: https://code.google.com/p/arducopter/.

[2] M. Vanin, "Modeling, identification and navigation of autonomous air vehicles", Master's thesis, KTH, Automatic Control, 2013. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-125752.

[3] M. Amoozadeh, *Smart mobility lab manual*, KTH Stockholm, 2013. [Online]. Available: https://code.google.com/p/kth-smart-mobility-lab/downloads/list.

[4] A. Marzinotto, "Cooperative control of ground and aerial vehicles", Master's thesis, KTH, Automatic Control, 2012. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?searchId=1&pid=diva2:547617.

[5] *Google code page for KTH Smart Mobility Lab.* [Online]. Available: https://code.google.com/p/kth-smart-mobility-lab/.

[6] "Tmote sky datasheet", Moteiv corporation, Data sheet, 2006. [Online]. Available: http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf.

[7] *Google code page for KTH Wireless Sensor Networks.* [Online]. Available: https://code.google.com/p/kth-wsn/.