User manual

Drönarprojekt Visionen

Editor: Emil Gustafsson

Version 1.0

Status

Reviewed	PL	2018-12-14
Approved	Client	2018-12-14

PROJECT IDENTITY

2018/HT,
Linköping University, Dept. of Electrical Engineering (ISY)

Group members

Name	**Responsibility**	**Phone**	**Liu-ID**
Otto Bergdahl	Project manager (PM)	070 57 28 442	ottbe754
Ali Murtatha Shuman	Project manager Stunt (PMS)	076 26 55 339	alish183
Jakob Palm	Project manager Hollywood (PMH)	076 01 75 783	jakpa844
Emil Gustafsson	Documentation (DOC)	072 55 63 156	emigu569
Morten Eriksson	Hardware Infrastructure (HRI)	070 35 15 263	morer916
Christian Jonsson	Design Infrastructure (DRI)	073 02 82 738	chrjo915
Thomas Liersch	Test Infrastructure (TRI)	076 16 74 114	tholi468
Roushan Rezvani	Software Infrastructure (SRI)	073 63 50 090	karre337
Toivo Rodriguez	Information manager (INF)	070 72 88 457	toiro677
Irman Svraka	Test Hollywood (TRH)	076 39 00 266	irmsv753
Jonas Vedin	Hardware Hollywood (HRH)	073 82 29 811	jonve231
Albin Vestin	Software Hollywood (SRH)	079 34 77 055	albve286
Robert Widén	Test Stunt (TRS)	076 82 90 315	robvi186
Markus Åstrand	Design Hollywood (DRH)	076 25 28 886	maras553
Erik Östberg	Git-Master (GM), (DRS)	076 80 61 570	erios967

Email list for the whole group: dronarprojekt@gmail.com
Web site: *http://www.isy.liu.se/edu/projekt/tsrt10/2018/dronarprojektvisionen*

Customer: ISY, Linköpings universitet, 581 83 Linköping
Customer contact: Gustaf Hendeby, 013-285815, gustaf.hendeby@liu.se
Client: Christian A. Naesseth, 013-281087, christian.a.naesseth@liu.se
Examiner: Daniel Axehill, 013-284042, daniel.axehill@liu.se
Supervisors: Kristoffer Bergman, kristoffer.bergman@liu.se
Per Boström-Rost, per.bostrom-rost@liu.se

Contents

Dokumenthistorik **5**

1 System overview **6**
1.1 Hardware 6
1.1.1 Qualisys positioning system 6
1.1.2 Projector system 6
1.1.3 Hollywood drone 6
1.1.4 Stunt drone 7
1.2 Software 8
1.2.1 Gazebo 8
1.2.2 Qualisys positioning system 8
1.2.3 PX4 firmware 8

2 Software setup **8**
2.1 Dependencies 8
2.1.1 Hollywood specific dependencies 9
2.1.2 Stunt specific dependencies 9
2.2 Download software 9
2.3 Configure computer 9
2.4 Changing parameters 10
2.5 Hollywood software setup 10
2.5.1 ROS-setup 10
2.5.2 Raspberry Pi setup 11
2.5.3 ACADO toolkit code generation 11
2.5.4 Pixhawk setup 11
2.6 Stunt software setup 12
2.6.1 cflib 12
2.6.2 Tkinter 12

3 System startup **12**
3.1 Projection 13
3.2 Calibration 13
3.3 Infrastructure 13
3.4 Hollywood 13
3.5 Stunt 14

4 User interfaces **14**

4.1 Hollywood 14

4.2 Stunt 15

Referenser **16**

Document history

Version	Date	Changes	Sign	Reviewed
0.1	2018-12-06	Revised according to comments from client	All	PL
0.1	2018-12-03	First draft.	All	PL

1 System overview

This system is composed of three main parts, two autonomous drone-platforms and an infrastructure system.

The infrasctructure system integrates the above mentioned, and similar, platforms with the positioning and projector systems available in Arena Visionen. The infrastructure also provides a simulation and visualisation environment.

The Hollywood Drone is an autonomous drone platform intended for trajectory following and target tracking. Specific controllers such as MPC- and p-controllers are implemented to improve reference- and attitude-following.

The Stunt Drone consists of a Crazyflie 2.0 drone and a Crazyradio PA USB dongle antenna for pc to platform communication. The stunt drone is to be able to perform various acrobatics in the form of rolls, flips and trajectory following and more.

1.1 Hardware

This section describes the hardware used in all subprojects.

1.1.1 Qualisys positioning system

The Qualisys positioning system is used to track bodies in a 3D space within Arena Visionen. This system consists of twelve OQUS infrared cameras [?] that emit infrared flashes with a frequency of 0-300 Hz. The cameras can then detect reflections and triangulate the position of the reflective surface. This system is pre-installed by Qualisys AB.

1.1.2 Projector system

Arena Visionen has three projectors, two in the ceiling projecting on the floor and one on a wall projecting on the opposite wall. The two projectors in the ceiling function as one. These projectors will be used to visualise the environment that is simulated for the drone to interact with and project the simulated view from a virtual camera. The projectors are connected to a dedicated PC located in the Arena Visionen control room.

1.1.3 Hollywood drone

The Hollywood drone is an autonomous drone platform that is intended to demonstrate target tracking and trajectory following using an MPC-controller. The following hardware is used:

Pixhawk 4 The Pixhawk 4 is an all-in-one flight controller running either PX4 or ardupilot firmware. In this project PX4 is used.

Raspberry Pi 3 model b+ A Raspberry Pi is used as a companion computer. It runs the MPC-controller and a route planner.

S500 airframe An S500 airframe is used together with motors and rotor blades.

3-cell LiPo battery A LiPo battery is used to power the drone in flight.

1.1.4 Stunt drone

The stunt drone is a demonstration platform which is able to perform aerobatics and follow trajectories. The drone used is the small scale quadcopter Crazyflie 2.0. It communicates with a PC via a Crazyradio PA USB dongle. An overhead sketch of the drone can be seen in Figure 1.
The following hardware is available:

Crazyflie 2.0 The Crazyflie is an open source, software development platform produced by Bitcraze AB. This quadcopter is small and lightweight but still contains a 10-DOF IMU with accelerometer, gyro, magnetometer and a high precision pressure sensor.

Crazyradio PA The Crazyradio PA is made by Bitcraze and it enables radio communication over distances up to 1000 meters. It is an open source project based on the “nRF24LU1” chip from Nordic Semiconductor.

3D-printed reflector holder for Crazyflie A reflector holder, see Figure 2, for the Crazyflie has been designed and 3D-printed to facilitate proper reflector position of the reflectors on the Crazyflie.

Drone batteries The drone uses a small scale 240mAh LiPo battery, and is charged with a 500mA USB charger. One fully charged battery gives approximately 7 minutes of flight time, and it takes approximately 40 minutes to fully charge a depleted battery.

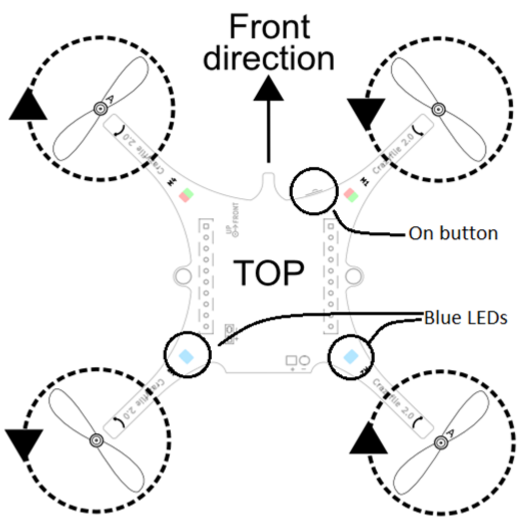

Figure 1: Sketch of the Crazyflie 2.0 [3]

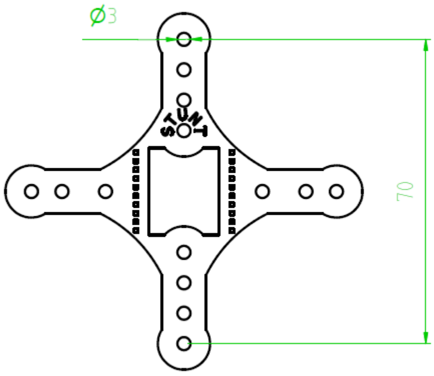

Figure 2: 3D-printed reflector holder for the Crazyflie 2.0.

1.2 Software

The following software are used in the project.

1.2.1 Gazebo

Gazebo is a robot simulation and visualisation program used in this project primarily as a visualisation tool. [4]

1.2.2 Qualisys positioning system

The software for the Qualisys postitioning system, QTM (Qualisys Track Manager), is located on a desktop computer in the Visionen control room. To learn how to use this system, please see the user manual which can be found at `http://content.qualisys.com/2017/03/Getting-started.pdf`

1.2.3 PX4 firmware

PX4 is a firmware for the Pixhawk 4 autopilot. Some parameters must be changed in the firmware before flight, see Section 2.5.4.

2 Software setup

This section goes through how to setup the software for every platform.

2.1 Dependencies

The required operating system for running the software is Ubuntu 16.04 (Xenial) (or another Linux distribution based on it). The following components are required for running the system software:

- ROS Kinetic Kame
- Catkin (Included in the full desktop installation of ROS Kinetic Kame)
- Gazebo version 7.x (Included in the full desktop installation of ROS Kinetic Kame)
- CatkinTools (Not necessary but recommended for easier development of catkin packages. Paths later in the document assume a catkin tools setup.)

2.1.1 Hollywood specific dependencies

Two ROS packages are needed to run the Hollywood platform. These are:

- Mavros
- catkin_simple

A more thorough description on how to install these can be seen in Section 2.5.1.

2.1.2 Stunt specific dependencies

The following python 2.7 packages and libraries are needed to be able to run the Stunt project:

- cflib
- TkInter
- tkFileDialog

2.2 Download software

The software necessary for using the system is contained in the project git repository. The required directory for the repository is in the `src` folder of a catkin workspace. The git repository can be cloned by typing the following command in the terminal:

```
$ git clone git@gitlab.ida.liu.se:tsrt10_2018/visionen.git
```

2.3 Configure computer

Besides downloading the proper software, some changes will need to be made to the computer system in order for the software to function. Insert the following lines at the end of the file `~/.bashrc`

```
source /opt/ros/kinetic/setup.bash
source {PATH_TO_CATKIN_REPO}/devel/setup.bash

export GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:$HOME/git/
    gazebo_ros_pkgs/gazebo_plugins:$HOME/{PATH_TO_CATKIN_REPO}/
    build/visionen_camera_description:{PATH_TO_CATKIN_REPO}/build
    /arena_visionen_gazebo
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:{PATH_TO_CATKIN_REPO
    }/src/visionen/arena_visionen_gazebo/models
```

Restart the terminal after these additions.

2.4 Changing parameters

The file `ui_params.yaml` located in the directory `/visionen/arena_visionen_gazebo/config` contains the configuration parameters for the system. The following parameters can be specified by the user:

zone_outer_boundaries The outer boundaries of the green and yellow zones. They are defined in the following way:
`[green.x, green.y, green.z, yellow.x, yellow.y, yellow.z]`.

gazebo_odom_sub Selects the odometry message that Gazebo subscribes to. Choose S for simulation, V for Visionen, or B for both.

To change the image on the Gazebo ground plane (the "floor" in the virtual environment), change the file name in the file `/visionen/arena_visionen_gazebo/models/my_ground_plane/materi`
`my_ground_plane.material`
at row 14.
The image file must be located in the directory
`/visionen/arena_visionen_gazebo/models/my_ground_plane/materials/textures`

2.5 Hollywood software setup

Following sections goes through how to setup the ROS-packages, the Raspberry Pi as well as how to redo the code generation for the MPC-controller.

2.5.1 ROS-setup

Two extra packages are required to build the Hollywoood drone ROS-package (also known as *holly*). These are the *mavros* and *catkin_simple* packages.

The *holly* package should be built using the *catkin build* command which is a part of catkin tools. However, the *catkin_simple* package must be built using the *catkin_make* command. To fix this issue, build the *catkin_simple* package with *catkin_make*, then remove the build and devel directories from the catkin workspace and rebuild the entire workspace using *catkin build*.

The *mavros* package can be installed by running the commands:

```
$ sudo apt-get install ros-kinetic-mavros
$ sudo apt-get install ros-kinetic-mavros-extras
```

2.5.2 Raspberry Pi setup

The Raspberry Pi companion computer needs to run an operating system that supports ROS Kinetic Kame and the required extra packages mentioned in Section 2.5.1. It is recommended to run Ubuntu MATE 16.04 as this is the only operating system that has been verified to work. **Note:** Ubuntu MATE 16.04 has a problem with the bootmanager for Raspberry Pi 3 model b+. A fix for this is to use a modified image with a fixed bootloader. This image file is available in the projects GIT repository.

2.5.3 ACADO toolkit code generation

The use of ACADO toolkit varies whether it is used for simulation or code generation. Simulation requires definition of references and *Gnuplot* is used to plot results. Observe that it is necessary to define references and including them in to the solvers *minimizeLSQ* and *minimizeLSQEndTerm* in order to run. Make sure that *OCPexport* is not included in the code when simulating.

The MPC-controller is modified using the *new_mpc.cpp* file in the */holly/ACADO_Code* directory. This file must be moved to your Acado workspace for simulations and code generation. To generate C-code from the actual code used in the ACADO toolkit for the MPC-controller, the function *OCPexport* should be used. To generate C-code, the reference vector must be removed from the solvers *minimizeLSQ* and *minimizeLSQEndTerm*. Also, the *Gnuplot*-function must be removed. Finally *OCPexportmpc* is used to generate the final files containing the finished MPC-controller. Finally a target directory must be defined and thereafter the code is simply run in ACADO. Implementing the code in to the controller node requires adding the genereated *qpoases*-folder as well as all of the files located in the */examples/getting_started/getting_started_export* map into the *solver* map.

2.5.4 Pixhawk setup

The Pixhawk flight controller runs the PX4 autopilot firmware. This can be installed from QGroundcontrol (ground station software which is compatible with Pixhawk) but it is recommended to build it from source instead because some modules might be missing in the QGroundcontrol installation. Before flight, the flight controller needs to be calibrated. This is only needed once and can be done via QGroundcontrol.

Parameters that need to be set in QGroundcontrol are listed in Table 1.

Table 1: PX4 parameters

Parameter	Value	Description
CBRK_GPSFAIL	240024	Allows flight without GPS-data.
CBRK_IOSAFETY	22027	Allows flight with I/O connections.
CBRK_USB_CHK	197848	Allows flight with USB connected.
SYS_MC_EST_GROUP	local_position_estimator OR ekf2	Position estimator, lpe required for offboard control.
ATT_EXT_HDG_M	Motion Capture	Allows the Pixhawk to recive mocap position. Only available if SYS_MC_EST_GROUP is set to lpe.
MAV_1_CONFIG	TELEM2	Sets mavlink to use the TELEM2 port
MAV_1_MODE	Onboard	Required for mavlink.
MAV_1_RATE	921600	Baudrate for mavlink port.

2.6 Stunt software setup

To be able to run the stunt project you must have the correct python libraries installed.

2.6.1 cflib

To ensure that the user installs the latest version of the Crazyflie Python API, which contains all the python modules for radio communication, we refer you to Bitcrazes github repository [2].

2.6.2 Tkinter

Tkinter is a standard package in many python distributions but can be installed from a terminal using.

```
$ apt-get install python-tk
```

3 System startup

This section describes how to run the launch files for each subproject.

3.1 Projection

To start the projector system, boot the dedicated projector computer in the arena Visionen control room and start the projector program. Select the projectors you wish to use and switch them on. To change to the Visionen network, switch the network cable that is plugged in to the computer. Any video stream created by the infrastructure can be accessed by typing

```
{IP_OF_COMPUTER_RUNNING_GAZEBO}:8080
```

in any web browser. On the web page you reach, click on the topic you wish to project. This will take you to a preview of the stream. To view the stream in real time, remove "_viewer" from the url. Now the window can be dragged to the desired position. Use fullscreen (F11) to guarantee proper scaling.

3.2 Calibration

Before any measurements can be made, QTM needs to be calibrated. For the physical origin to coincide with the simulated origin in Gazebo, type the following command in a terminal:

```
$ roslaunch arena_visionen_gazebo world_calibration.launch
```

This will launch a Gazebo environment with the L-frame, used in QTM calibration, located in the virtual origin. Project the video stream of this environment onto the floor of arena Visionen and place the real L-frame on top of the projected one.

3.3 Infrastructure

When planning to use physical measurements from arena Visionen, make sure that the computer running the infrastructure software is connected to the "Visionen" network and that the QTM program is currently measuring on its dedicated computer in the control room. In order to start the infrastructure, type the following commands in a terminal:

```
$ roslaunch arena_visionen_gazebo arena_visionen.launch
```

This command will launch the virtual environment in Gazebo and set up a video server for streaming video of the simulated environment.

3.4 Hollywood

To run the hollywood drone project simply type:

```
$ roslaunch holly holly.launch
```

and then in a new terminal run:

```
$ rosrun holly user_interface
```

By default this runs a simulation. To start the real drone platform project the *holly.launch* file must be modified to instead of launching the simulation, launch mavros and the offboard node. This can be done by removing the simulation node and uncommenting the mavros node and the offboard node.

3.5 Stunt

To run the stunt drone project open a terminal and enter the following:

```
$ roslaunch stunt basic.launch
```

This will set up the stunt specific nodes together with the stunt GUI application further explained in Section 4.2.

4 User interfaces

This section describes how the user can control each platform.

4.1 Hollywood

The user interface for the hollywood drone is based purely on keyboard input in a Linux terminal.

Table 2: Hollywood user commands.

Keyboard input	Functionality
a	Sets the mode to autonomous flight
s	Sets the mode to simulated flight
m	Sets the mode to manual flight
t	Take off, sets the drone to 'AUTO.LAND' mode
l	Land, sets the drone to 'AUTO.TAKEOFF' mode
o	Set the drone to 'OFFBOARD' mode
space	Abort signal
r	Reset signal
enter	Start/stop mission
0-7	Initiate mission 0-9, as defined in Table 3
p	Get current position of the drone

Table 3: Missions of the Hollywood drone.

Mission	Description
0	Fly to specific point
1	Rotate to face stationary target
2	Fly to a sequence of specific points
3	Fly to specific point while facing a stationary target
4	Fly to a sequence of specific points while facing a stationary target
5	Hover while facing a moving target
6	Follow a moving target with specific distance
7	Follow a moving target with specific distance while facing the target

To run an example of flying to a specific point in simulation, start up the user interface and type as following:

- Press s to set the mode "simulated flight".
- Press 0 (zero) to initiate the "fly to a specific point"-mission.
- Press Enter to start the mission.

Now you can choose to stop the current mission and start another, for example flying to a sequence of points, by:

- Press Enter to stop the mission.
- Press 2 to initiate the "fly to a sequence of points"-mission.
- Press Enter to start the selected mission.

4.2 Stunt

The graphical user interface for the stunt drone is shown in Figure 3 and is used both for simulation and flying. The figure shows the GUI as it is directly after startup, i.e., with the drone not connected. To connect, click the red connect-button. When a connection has been established this button will turn green, and the drone is ready to receive commands. When the connection is established the drone will ramp the thrust to indicate that a proper connection was made, otherwise the drone connection may have failed. The commands can be chosen from the drop-down menu to the left. The parameters used for each command are given in meters, except for the roll and flip (pitch) commands, which needs a rotation direction specified. 1 for a clockwise rotation, and -1 for a counter-clockwise rotation. To send the command to the drone, click the send command-button.

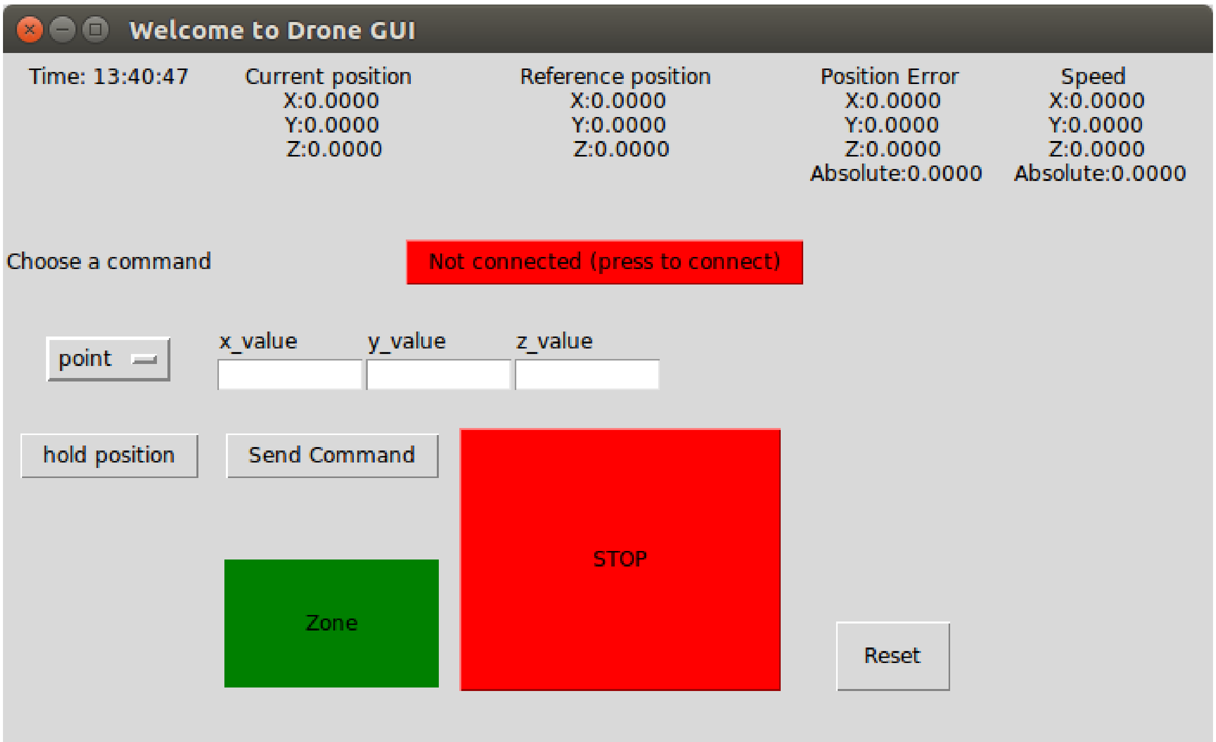

Figure 3: Stunt GUI in an unconnected state

The green zone field in Figure 3 indicates in which zone the drone is located by changing its colour. The red stop button sends an emergency stop signal when pressed, which freezes all motors. After the stop button has been pressed, the reset button has to be pressed in order to be able to send further commands. At the top row, odometry data is shown in meters.

To run a example of flying to a specific point in simulation, choose point in the drop down menu in the GUI. Type in a desired position in the coordinate boxes and click the send command button to execute the command.

References

[1] *LIPS – nivå 1. Version 1.0.* Tomas Svensson och Christian Krysander. Kompendium, LiTH, 2002.

[2] *Bitcraze Github/cflib* Bitcraze AB
Available at: *https://github.com/bitcraze/crazyflie-lib-python*

[3] Read me file from Qualisys Github. *https://github.com/qualisys/crazyflie-resources*

[4] The Gazebo homepage *http://gazebosim.org*