A.R Drone Navigation – Application Design Document

Authors: Yuri Bakulin, Maxim Kirilov & Yuval Kovler

Academic Advisor: Dr. Yuval Elovici
Technical Advisor: Rami Puzis

http://www.cs.bgu.ac.il/~maximkir/ardrone/
Table of Contents

1. CHAPTER 1 – USE CASES ........................................................................................................ 6
   1.1. Usage Scenario Number 1 - The AR Drone Takes-Off from the ground .................. 6
   1.2. Usage Scenario Number 2 - Drone moves along to the marked path .................. 7
   1.3. Usage Scenario Number 3 - Checkpoint handling .................................................. 9
   1.4. Usage Scenario Number 4 - Searching for a path ................................................... 10
   1.5. Usage Scenario Number 5 - Stop Button pushed - the Drone lands ...................... 12

2. CHAPTER 2 – SYSTEM ARCHITECTURE ........................................................................... 13
   2.1. Global Architecture ...................................................................................................... 13
   2.2. System Architecture ...................................................................................................... 14

3. CHAPTER 3 – DATA MODEL ............................................................................................. 16

4. CHAPTER 4 BEHAVIORAL ANALYSIS ............................................................................. 17
   4.1. Sequence Diagrams ...................................................................................................... 17
       4.1.1. Taking off .............................................................................................................. 17
       4.1.2. Landing ............................................................................................................... 17
       4.1.3. Video frame process .......................................................................................... 18
   4.2. Events .......................................................................................................................... 19
   4.3. States ........................................................................................................................... 19
   4.4. Flow Diagram ............................................................................................................. 20

5. CHAPTER 5 - OBJECT ORIENTED ANALYSIS ............................................................... 21
   5.1. Class Diagrams ............................................................................................................ 21
       5.1.1. Control ............................................................................................................... 21
       5.1.2. Image Processing .............................................................................................. 22
       5.1.3. Navigation ......................................................................................................... 23
       5.1.4. Utility ................................................................................................................. 24
   5.2. Class Descriptions Class Descriptions ....................................................................... 25
       5.2.1. Communication ................................................................................................. 25
           5.2.1.1. Autopilot .................................................................................................... 25
           5.2.1.2. Drone ....................................................................................................... 25
5.2.1.3. UdpWorker ................................................................................................................. 26
5.2.1.4. VideoDataRetriever ................................................................................................. 27
5.2.1.5. NavigationDataRetriever ......................................................................................... 27
5.2.1.6. CommandSender ........................................................................................................ 27

5.2.2. Image Processing ........................................................................................................ 28
5.2.2.1. SignDetector .............................................................................................................. 28
5.2.2.2. QRcodeDetector ......................................................................................................... 28
5.2.2.3. GroundLineDetector ................................................................................................. 28
5.2.2.4. SignSelector ................................................................................................................ 29
5.2.2.5. BiggestSignSelector .................................................................................................. 29
5.2.2.6. FeatureExtractor ....................................................................................................... 29

5.2.3. Navigation .................................................................................................................... 30
5.2.3.1. CourseAdvisor .......................................................................................................... 30

5.2.4. Utility ............................................................................................................................ 30
5.2.4.1. CheckPointHandler .................................................................................................... 30
5.2.4.2. ImageFormatter ......................................................................................................... 31

5.3. Packages .......................................................................................................................... 31
5.3.1. Control Package ............................................................................................................ 31
5.3.2. Utilities Package ............................................................................................................ 31
5.3.3. Navigation Package ..................................................................................................... 31
5.3.4. Image Processing Package ............................................................................................ 32

5.4. Unit Testing ...................................................................................................................... 32
5.4.1. Communication .............................................................................................................. 32
5.4.1.1. AutoPilot .................................................................................................................. 32
5.4.1.2. Drone ......................................................................................................................... 33
5.4.2. Image Processing .......................................................................................................... 35
5.4.2.1. SignDetector .............................................................................................................. 35
5.4.2.2. BiggestSignSelector .................................................................................................. 35
5.4.2.3. QRCodeDetector ...................................................................................................... 36
5.4.2.4. GroundLineDetector ................................................................. 36
5.4.2.5. AngleExtractor ........................................................................ 36
5.4.3. Navigation ..................................................................................... 37
  5.4.3.1. CourseAdvisor ......................................................................... 37
5.4.4. Utility ............................................................................................. 37
  5.4.4.1. CheckPointHandler .................................................................. 37

6. CHAPTER 6 - USER INTERFACE DRAFT .............................................. 38

7. CHAPTER 7 - TESTING ........................................................................ 39
  7.1. General Schema ............................................................................... 39
  7.2. Testing Functional Requirements ..................................................... 39
  7.3. Testing Non-Functional Requirements ............................................. 39

8. CHAPTER 8 - TASK LIST ................................................................... 41
  8.1. Control task list ............................................................................... 41
  8.2. Utility task list ................................................................................ 41
  8.3. Image processing task list ............................................................... 41
  8.4. Navigation task list ......................................................................... 42
  8.5. User Interface .................................................................................. 42
  8.6. Auto Pilot ........................................................................................ 42
  8.7. Documents and Projects Day task list ............................................ 43
Terms commonly used:

1. NavData – Navigation Data that is send approximately 30 times per second from the drone and includes the following data:
   a. Angular position : Psi, Phi, Theta.
   b. Altitude.
   c. Speed Gaz(Vertical), Yaw (Angular)
   d. Battery level
2. Path – Marked Line on the floor.
4. AT command – The set of commands that are the communication protocol of the AR Drone.
5. openCV (Open Source Computer Vision) - is a library of programming functions for real time computer vision.
6. Emgu CV is a cross platform .Net wrapper to OpenCV.
7. The Battery is charged – means the battery is charge more than 20%.

Important note:
There are a few usages that seem abstract in the Use Cases; those exist because we have a few “Black Boxes”.
All of the black boxes are those from the side of the AR Drone and its firmware and SDK – including the video recording and sending, sensors data and performing the AT Commands.
1. **Chapter 1 – Use Cases**

1.1. **Usage Scenario Number 1 - The AR Drone Takes-Off from the ground**

**ID:** 1

**Primary actors:** AR Drone, Computer, Operator

**Trigger:** The Operator clicks on the “Start” button.

**Pre-conditions:**

1. The Drone is sitting on a horizontal ground with the bottom camera facing down.
2. The Drone is placed without any obstacles on top of him.
3. The drone is placed at the starting point.
4. A Path has been marked.
5. The Auto-pilot Software on the computer is initialized.
6. The Drone and the computer have established a Wi-Fi connection (network working correctly).
7. The Battery is fully charged.

**Post-conditions:**

1. The drone is up in the air

**Flow of events:**

1. The AT Command “Take-Off” is sent by the computer and received by the Drone.
2. The Drone Starts its Engines and all his sensors.

**Alternative flows:**

2. a **The Drone is not able to starts its Engines Action:**

   2. a. 1. The Build in firmware on the Drone will stop the drone’s actions.
   2. a. 2. The drone sends the Navdata to the computer.
   2. a. 3. The computer interprets the Navdata and displays a failure message on the screen.
   2. a. 4. End of use case
1.2. **Usage Scenario Number 2 - Drone moves along to the marked path.**

**ID:** 2

**Primary actors:** AR Drone, Computer.

**Trigger:** Video frame received.

**Pre-conditions:**

1. The Drone’s position is horizontal (moving is correct - with the bottom camera facing down).
2. A Path has been marked.
3. The Auto-pilot Software on the computer is initialized.
4. The Drone and the computer have established a Wi-Fi connection (network working correctly).
5. The Battery is charged.
6. Use Case number 1 has finished successfully.
7. Use Case number 3¼ isn’t running.

**Post-conditions:**

1. The drone moves according to the commands sent to him.

**Flow of events:**

1. The computer formats the video frame from “source bitmap” to “image” (EMGU format) and shows it on the screen.
2. The computer checks there isn’t a checkpoint by running the Image process algorithm for identifying a checkpoint on the formatted video frame.
3. The computer runs the Image process algorithm for identifying a path on the formatted video frame and decides on a direction for the drone to continue.
4. The computer shows the processed video frame on the screen.
5. The computer calculates the set of AT commands that are equal to the calculated direction.
6. The computer sends the set of AT commands to the drone.
7. The drone receives the AT commands and performs them.

**Alternative flows:**

1.b **The computer is unable to format the video frame.**

**Action:**
1.b 1. The computer displays an appropriate message on the screen.
1.b 2. End of use case.

2.a The computer isn’t able to run the Image process algorithm

Action:

2.a 1. The computer displays an appropriate message on the screen.
2.a 2. End of use case.

2.b The computer runs the Image process algorithm and detects a checkpoint.

Action:

2.b 1. Use Case number 3 - Checkpoint handling is started.
2.b 2. End of use case.

3.a The computer runs the Image process algorithm for identifying a path and isn’t able to identify a path.

Action:

3.b 1. The computer displays an appropriate message on the screen and adds to the counter of continuous failures.
3.b 2. If the counter of continuous failures equals to predefined value then use case 4 – “searching for a path” starts.
3.b 2. End of use case.
1.3. **Usage Scenario Number 3 - Checkpoint handling**

**ID:** 3

**Primary actors:** AR Drone, Computer

**Trigger:** Use case 2 - alternative step 2.a 1. has been made.

**Pre-conditions:**

1. The Drone’s position is horizontal (moving is correctly with the bottom camera facing down).
2. A Checkpoint has been identified.
3. The Auto-pilot Software on the computer is initialized.
4. The Drone and the computer have established a Wi-Fi connection (network working correctly).
5. The Battery is charged.
6. Use Case number 1 has finished successfully.

**Post-conditions:**

1. The Drone stays in the same position as it was before.

**Flow of events:**

1. The computer sends the drone the “Hover” AT Command.
2. The drone receives the “Hover” AT Command and hovers.
3. After a predefined period of time the reverse of “Hover” AT Command is been sent by the computer to the drone.
4. The drone receives the command and recovers from the “Hover” mode.
1.4. **Usage Scenario Number 4 - Searching for a path**

**ID:** 4

**Primary actors:** AR Drone, Computer

**Trigger:** Use case 2 - alternative step 5.b. 2 have been made.

**Pre-conditions:**

1. The Drone’s position is horizontal (moving is correctly with the bottom camera facing down).
2. A path has been marked.
3. The Auto-pilot Software on the computer is initialized.
4. The Drone and the computer have established a Wi-Fi connection (network working correctly).
5. The Battery is charged.
6. Use Case number 1 has finished successfully.

**Post-conditions:**

1. Use case 2 has been activated
   
   **Or**
   
   The drone lands and an appropriate message is displayed to the user.

**Flow of events:**

1. The computer sends a command to the drone to go left a predefined distance.
2. The Drone receives and does the command.
3. The computer receives each frame and runs the Image process algorithm for identifying a path and fails to recognize a path.
4. The computer sends a command to the drone to go right (to return to the same place) a predefined distance.
5. The computer sends a command to the drone to go right a predefined distance.
6. The Drone receives and does the command.
7. The computer receives each frame and runs the Image process algorithm for identifying a path and fails to recognize a path.
8. The computer sends a command to the drone to go left (to return to the same place) a predefined distance.
9. The computer sends a command to the drone to go toward a predefined distance.
10. The Drone receives and does the command.
11. The computer receives each frame and runs the Image process algorithm for identifying a path and fails to recognize a path.
12. The computer sends a “Land” AT command and an appropriate message is displayed to the computer.

Alternative flows:

3.a\7.a\11.a Image process algorithm for identifying a path successfully recognizes a path.
   3.a\7.a\11.a 1 An appropriate message is displayed to the user.
   3.a\7.a\11.a 2 End of use case.
1.5. **Usage Scenario Number 5 - Stop Button pushed - the Drone lands.**

**ID:** 5

**Primary actors:** AR Drone, Computer, Operator.

**Trigger:** The operator clicks on the stop Button.

**Pre-conditions:**

1. The Auto-pilot Software on the computer is initialized.
2. The Drone and the computer have established a Wi-Fi connection (network working correctly).
3. The Battery is charged.
4. Use Case number 1 has finished successfully.

**Post-conditions:**

1. The drone is on the ground.
2. The drone’s engines are off.

**Flow of events:**

1. The Computer sends a land AT Command to the drone.
2. The drone receives the land command and starts landing.
3. The drone sends Navdata.
4. The computer receives the Navdata and recognizes when the drone landed and displays an appropriate message on the screen.
2. **Chapter 2 – System Architecture**

2.1. **Global Architecture**

Our system has four main parts (Figure 1):

1. **ARDrone-Control-.NET** –
   
   An application for flying the Parrot AR drone in Windows
   
   This framework is used to abstract the whole Aviation control.
   
   We are using it to receive all the info that is sent from the drone (Video Stream and NavData), and easily send commands to the drone.

2. **AR Drone** –
   
   The AR Drone operates on "BusyBox" which is a small embedded Linux distro.
   
   We are using the AR Drone’s built in firmware to communicate with the drone via Wi-Fi to do the following:

   a. Send commands.
   
   b. Receive video stream & navigation data so we could calculate the next move.

3. **EMGU** –
   
   We are using the .Net wrapper to the Intel OpenCV image processing library
   
   To process the video frames received from ARDrone-Control-.NET framework.
   
   We will run various image processing algorithms for identifying a path/checkpoint using the library’s functions.

4. **Autopilot** –
   
   The last and most important part of the system - the part that is developed by us.
   
   In its main responsibilities is coordinating between all the parts of the system – running the image processing algorithms after receiving the video frame and then calculating the drones next command, this part will be more detailed in the next section.
### 2.2. **System Architecture**

1. The Drone which acts like a server sending flight information and video while also receiving flight directions.

2. A desktop client program which analyzes the data received from the drone and generates flight directions according to the images.
The desktop client is written in C# using Microsoft's .NET framework. The system consists of three main modules:

**Control:**
This module contains an implementation of the Drone's communication protocol (AT Commands) and an easy to work with interface for communicating with the drone. It also contains dedicated worker threads responsible for retrieving updates video and flight information data from the drone in real time.

**Image Processing:**
This module consists of two parts:

1. A Path Detection component responsible of identifying the path marker in the image and return the direction of the target.

2. A Checkpoint Detection component responsible of identifying checkpoint markers and performing the required action.

The image processing is designed to be interchangeable in order to enable easy testing and use of different path marking and detection techniques as well as different ways to mark, detect and respond to checkpoints.

**Navigation:**
This module is responsible of maintaining the drone's flight path. It incorporates data from the image processing module to compute 3D navigation directions for the drone.
3. **Chapter 3 – Data Model**

In fact that the project is dealing with live video processing and it is important to state that there isn’t any data storing the whole issue of developing a data model and storing data is irrelevant to our project. – meaning it is **Not Applicable**.
4. Chapter 4 Behavioral Analysis

4.1. Sequence Diagrams

4.1.1. Taking off

![Taking off sequence diagram]

Figure 3 – Taking off sequence diagram

4.1.2. Landing

![Landing sequence diagram]

Figure 4 – landing sequence diagram
4.1.3. Video frame process

Figure 5 – Video Frame Process
4.2. **Events**

Our system is an autonomous system, therefore we think that our software system isn’t best described and understood in terms of the events but in terms of states.

4.3. **States**

Figure 4 describes the behavior of the system in state-machine form. We leave out the take-off and landing phases and describe the main operation loops of the program and the Drone.

![State-machine diagram](image)

*Figure 6 - System state-machine representation.*
4.4. Flow Diagram

The following is a flow diagram describing the main control flow of our program:

Figure 7 Flowchart of system operation.
5. **Chapter 5 - Object Oriented Analysis**

5.1. **Class Diagrams**

5.1.1. **Control**

![Class Diagram of the “Control” Package](image_url)

Figure 8 – Class Diagram of the “Control” Package
5.1.2. Image Processing

Figure 9 - Class Diagram of the “Image Processing” Package
5.1.3. Navigation

Figure 10 - Class Diagram of the “Navigation” Package
5.1.4. Utility

Figure 11 - Class Diagram of Utilities
5.2. **Class Descriptions**

5.2.1. **Communication**

5.2.1.1. **Autopilot**

Main class – starts the AR Drone application.

**Init:**
- Establishing connection to the AR Drone.
- **Pre-Condition:** the AR Drone is on & in proper condition and the drone is actually sits on a horizontal ground.
- **Post-Condition:** return true if and only if all workers have been started.

**StartAutoPilot:**
- Starts the AR Drone autopilot.
- **Pre-Condition:** initialization was finished successfully (return true).
- **Post-Condition:** the AR Drone starts to look for a path.

**Dispose:**
- Shuts down the system.
- **Pre-Condition:** none.
- **Post-Condition:** the system releases all resources and disconnects from the AR Drone.

5.2.1.2. **Drone**

Represents the AR Drone.

**Take-off:**
- Sends take off command to the AR Drone.
- **Pre-Condition:** the drone was initialized; the AR Drone sits on a horizontal ground.
- **Post-Conditions:** the AR Drone is hovering 1 meter above the ground.

**Land:**
- Sends land command to the AR Drone.
- **Pre-Condition:** take off command was invoked.
- **Post-Condition:** the drone is on ground.
Connect:
- Connects to the AR Drone.
- **Pre-Condition**: the drone Wi-Fi is on.
- **Post-Condition**: the workers start to receive data from the drone.

Disconnect:
- Disconnects from the drone.
- **Pre-Condition**: connection to drone is established.
- **Post-Condition**: all open connections were closed.

SendCommand:
- Sends command to the AR Drone.
- **Pre-Condition**: connection was established and take-off was invoked.
- **Post-Condition**: none (no grantees that the command was received).

SwitchCamera:
- Sends a special command to the drone to switch the video camera feed.
- **Pre-Condition**: connection to the AR Drone was established.
- **Post-Condition**: none (no grantees that the command was received).

GetInstance:
- Returns Drone instance (drone is singleton).
- **Pre-Condition**: none.
- **Post-Condition**: none.

5.2.1.3. **UdpWorker**

Responsible for the communication with different AR Drone components (sensors).

CreateUdpSocket *(address, port-number, time-out)*:
- Creates UDP socket to specific port at given address within given timeout.
- **Pre-Condition**: the AR Drone is running on the address and listening on the port number given.
- **Post-Condition**: return true if and only if a socket has been established successfully between the AR Drone and the Worker.
SendMessage (message):

- Sending message to the AR Drone.
- **Pre-Condition**: a socket is open and the AR Drone is listening on the input stream.
- **Post-Condition**: none.

DisconnectFromSocket:

- Closing connection with the AR Drone component.
- **Pre-condition**: a socket is open and the AR Drone is listening on the input stream.
- **Post-Condition**: the socket is closed.

5.2.1.4. **VideoDataRetriever**

Responsible for process (format) received images from the AR Drone.

**GetCurrentImage**:

- Getter of the last received image from the AR Drone.
- **Pre-Condition**: connection to the AR Drone camera was established.
- **Post-Condition**: Return the source in Image format.

5.2.1.5. **NavigationDataRetriever**

Responsible for parse the received NavData from the AR Drone.

**GetCurrentNavigationData**:

- Getter of the last received NavData from the AR Drone.
- **Pre-Condition**: connection to the AR Drone NavData port was established
- **Post-Condition**: Return AR Drone NavData.

5.2.1.6. **CommandSender**

Responsible for commands delivery to the AR Drone. In addition this worker keeps active commands connection to the AR Drone by sending empty commands while the commands queue is empty.

**SendQueuedCommand (command)**:

- Puts the command into commands queue.
• Pre-Condition: Connection to the AR Drone was established.
• Post-Condition: none.

**SendUnqueuedCommand (command):**
• Sends the command immediately.
• Pre-Condition: Connection to the AR Drone was established.
• Post-Condition: none.

### 5.2.2. Image Processing

#### 5.2.2.1. SignDetector
Responsible for signs detection. This is a base class for all algorithms for signs recognition.

**DetectSign (image):**
• Detect sign on input image.
• Pre-Condition: image is valid.
• Post-Condition: Return list of sign detections if and only if signs exist in the image.

#### 5.2.2.2. QRcodeDetector
Responsible for detecting QR code signs.

**DetectSing (image):**
• Detects QR code signs.
• Pre-condition: image is valid.
• Post-condition: Return list of sign detections if and only if QR code are exist in the image.

#### 5.2.2.3. GroundLineDetector
Responsible for detecting pre-defined colored strips.

**DetectSign (image):**
• Detect colored strip.
• Pre-condition: stripe width should be at least 5 cm.
• Post-condition: Return list of sign detections if and only if colored stripe exist in the image.
5.2.2.4. **SignSelector**

Base class for various algorithms for selecting one sign from list of detected signs.

**ExtractSign** (*image, list of signs)*:
- Extracts one sign from list.
- **Pre-Condition**: legal image and legal signs.
- **Post-Condition**: one sign returned.

5.2.2.5. **BiggestSignSelector**

Selects the biggest (with the biggest volume) sign from list of detected signs.

**ExtractSign** (*image, list of signs)*:
- Selects one sign from signs list.
- **Pre-condition**: legal image and legal signs.
- **Post-Condition**: Return the biggest detected sign if and only if this sign has the biggest volume.

5.2.2.6. **FeatureExtractor**

Interface for different features that can be extracted from detected signs.

**ExtractFeature** (*image, sign-area)*:
- Extracts feature from image.
- **Pre-Condition**: image and area sign are legal values.
- **Post-Condition**: some object that implements the Feature interface is return.

**AngleExtractor** (*image, sign-area)*:
- Extracts an angle from linear approximation to the sign. The angle is relative to line intersection with x-axis.
Pre-Condition: image and area sign are legal values.
Post-Condition: some object that implements the Feature interface is return.

HorizontalDistanceExtractor (image, sign-area);
- Extracts distance estimation from the sign.
- Pre-Condition: image and area sign are legal values.
- Post-Condition: some object that implements the Feature interface is return.

5.2.3. **Navigation**

5.2.3.1. **CourseAdvisor**
Base class for course advisors, responsible for directions extraction from detected signs in image.

GetNavigationAdvice (image, sign-area):
- Process the signs according to specific designation.
- Pre-Condition: none.
- Post-Condition: Return directions if and only if area of sign isn’t empty.

5.2.4. **Utility**

5.2.4.1. **CheckPointHandler**
Responsible for handle check points.

HandleCheckPoint (checkpoint):
• Handles check point according to specific implementation.
• Pre-Condition: valid check point.
• Post-Condition: Return true if and only if the received check point wasn’t handled before.

5.2.4.2. **ImageFormatter**

Responsible for convert an image from a raw type (bitmap source) to Image type.

**BitmapSourceToImage (bitmap):**

• Converts from bitmap source to image.
• Pre-Condition: valid image.
• Post-Condition: image representation of the bitmap source.

5.3. **Packages**

The classes are divided into 4 packages as follow:

5.3.1. **Control Package**

Control- This package contains classes that are responsible of communicating with the AR Drone.

Control.Workers – This package contains classes that are responsible for handle different connections to AR Drone.

Control.Tests – Stores all Control package tests.

5.3.2. **Utilities Package**

Utilities – This package contains different utilities for video, network and configuration.

Utilities.Commands – This package contains AT commands extensions.

Utilities.Configuration – This package contains different configuration files.

Utilities.Tests – Stores all Utilities package tests.

5.3.3. **Navigation Package**

Navigation – This package contains classes for computation the next flight commands.

5.3.4. **Image Processing Package**

ImageProcessing – This package contains designated classes for different algorithms for path and sings recognition.

ImageProcessing.Tests – Stores all Image processing package tests.

5.4. **Unit Testing**

5.4.1. **Communication**

5.4.1.1. **AutoPilot**

**Init:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The AR Drone is on and no other connection connected to him</td>
<td>Connection to the AR Drone is established, all workers started. True value returned.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>The AR Drone is off.</td>
<td>Exception is thrown with message: “Cannot establish connection to the AR Drone”.</td>
</tr>
<tr>
<td>3</td>
<td>Failure</td>
<td>The AR Drone is on and other client is connected to him.</td>
<td>Exception is thrown (same message as above).</td>
</tr>
</tbody>
</table>

**Dispose:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Connection to the AR Drone was established, AR Drone on ground.</td>
<td>All active connections are closed.</td>
</tr>
<tr>
<td>2</td>
<td>Success</td>
<td>Connection to the AR Drone established and AR Drone is flying.</td>
<td>AR Drone lands, all active connections are closed.</td>
</tr>
<tr>
<td>3</td>
<td>Failure</td>
<td>No active connection to the AR Drone.</td>
<td>Exception is thrown.</td>
</tr>
</tbody>
</table>
5.4.1.2. **Drone**

**GetFrame:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Connection to the AR Drone was established, front camera set to default. Stop sign place before the AR Drone camera.</td>
<td>The stop sign should image should be returned.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>No active connection to the AR Drone.</td>
<td>Exception thrown.</td>
</tr>
</tbody>
</table>

**GetNavData:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Connection to the AR Drone was established. The AR Drone is placed on horizontal ground.</td>
<td>Drone data should be return with all values set to 0.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>No active connection to the AR Drone.</td>
<td>Exception thrown.</td>
</tr>
<tr>
<td>3</td>
<td>Success</td>
<td>Connection to the AR Drone was established, the AR Drone held 1 meter above the ground.</td>
<td>Drone date returned with altitude value 1.</td>
</tr>
</tbody>
</table>

**TakeOff:**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Connection to AR Drone was established, the AR Drone sits on a horizontal ground.</td>
<td>The AR Drone takes off.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>No active connection to the AR Drone.</td>
<td>Exception thrown.</td>
</tr>
<tr>
<td>3</td>
<td>Success</td>
<td>The AR Drone is flying.</td>
<td>Nothing happens.</td>
</tr>
<tr>
<td>4</td>
<td>Failure</td>
<td>Connection to AR Drone was established, the AR Drone sits on a vertical ground.</td>
<td>The AR Drone enters to emergency mode.</td>
</tr>
</tbody>
</table>
slope ground.

Land:

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The AR Drone is flying.</td>
<td>The AR Drone lands,</td>
</tr>
<tr>
<td>2</td>
<td>Success</td>
<td>The AR Drone is on ground</td>
<td>Nothing happens.</td>
</tr>
</tbody>
</table>

SendCommand:

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The AR Drone is flying. Play led animation command is send each second during one minute.</td>
<td>The AR Drone led lamps change their color at least once.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>No active connection to the AR Drone.</td>
<td>Exception thrown.</td>
</tr>
<tr>
<td>3</td>
<td>Success</td>
<td>The AR Drone is flying. 3 flight move forward commands are sent to the AR Drone.</td>
<td>The AR Drone moves forward.</td>
</tr>
</tbody>
</table>
### 5.4.2. Image Processing

#### 5.4.2.1. SignDetector

**DetectSign (image):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image object with one sign.</td>
<td>Sign Result object is returned with sign detection.</td>
</tr>
<tr>
<td>2</td>
<td>Success</td>
<td>The method is invoked with legal image object without signs.</td>
<td>Null object is returned.</td>
</tr>
</tbody>
</table>

#### 5.4.2.2. BiggetsSignSelector

**ExtractSign (image, list of signs):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image object and 2 detected signs: first sign has volume of 10 units and the other has volume of 2 units.</td>
<td>Sign Result object is returned with the first sign.</td>
</tr>
<tr>
<td>2</td>
<td>Success</td>
<td>The method is invoked with legal image object with one detected sign.</td>
<td>Sign Result object is returned with the received sign.</td>
</tr>
<tr>
<td>3</td>
<td>Success</td>
<td>The method is invoked with legal image object with two identical detected signs.</td>
<td>Sign result object is returned with the sign that his center of mass is closer to the image center.</td>
</tr>
</tbody>
</table>
5.4.2.3. **QRCodeDetector**

**DetectSign (image):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image object with one QR code sign.</td>
<td>Sign Result object is returned with sign detection around the QR code.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>The method is invoked with legal image object without QR code sign.</td>
<td>Null object is returned.</td>
</tr>
</tbody>
</table>

5.4.2.4. **GroundLineDetector**

**DetectSign (image):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image object with one stripe yellow.</td>
<td>Sign Result object is returned with sign detection around the yellow stripe.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>The method is invoked with legal image object without strip sign.</td>
<td>Null object is returned.</td>
</tr>
</tbody>
</table>

5.4.2.5. **AngleExtractor**

**ExtractFeature (image, sign-area):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image with detected sign. The approximation line intersects the x axis at 45 degree.</td>
<td>Angle object is returned with angle value: $40 \leq angle \leq 50$</td>
</tr>
</tbody>
</table>
**HorizontalDistanceExtractor (image, sign-area):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image with detected sign. The sign approximate distance from the AR Drone is 10 meters.</td>
<td>Distance object is returned with distance value: $8 \leq distance \leq 12$</td>
</tr>
</tbody>
</table>

**5.4.3. Navigation**

5.4.3.1. **CourseAdvisor**

**GetNavigationAdvice (image, sign-area):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>The method is invoked with legal image and sign area. The image has sign that his linear approximation is intersects with the x axis in 60 degrees.</td>
<td>The navigation adviser will return 10 flight move commands. These commands will turn the AR Drone 30 degrees against clockwise.</td>
</tr>
</tbody>
</table>

**5.4.4. Utility**

5.4.4.1. **CheckPointHandler**

**HandleCheckPoint (checkPoint):**

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
<td>Valid check point that wasn’t handled before.</td>
<td>Return true value.</td>
</tr>
<tr>
<td>2</td>
<td>Failure</td>
<td>Valid check point that was handled before.</td>
<td>Return false value.</td>
</tr>
</tbody>
</table>
6. **Chapter 6 - User Interface Draft**

The user interface is a simple GUI as shown in the next figure:

![User Interface Diagram]

**Figure 13 – user interface example**

**The GUI includes:**

2. Status – information on the status of the drone.
3. The original video stream that we receive from the AR DRONE along with the processed video stream with identified path/checkpoints.
4. User Buttons – the only interaction that the user has are two buttons:
   - A. The start button – used to take-off the drone and start the autopilot navigation.
   - B. The stop button (will be used only in emergencies) – used to land the drone wherever it is flying.
5. An informative shell in which all the important messages will be displayed to view (e.g. failure to detect a path).
7. Chapter 7 - Testing

7.1. General Schema

The project’s testing is divided to 3 levels of tests:

**Unit Tests:**
Testing each class as a separate unit.

**Components Testing:**
We have 3 main components in our system – control, navigation and image processing. Each of the components are tested separately. Then a connection of each of the components with components connected to him is being tested using a Stub of all connected components to him.

**End-to-End Testing:**
A full flow testing for each user-case scenario. The End to End test contain a full scenario that is being executed and they allow us to check the properties of the system and all requirements.

7.2. Testing Functional Requirements

In order to test all functional requirements we will combine the whole system. We will manually test all functional requirements.

7.3. Testing Non-Functional Requirements

We will test the nonfunctional requirements in the following manner:

1. **AR Drone speed** – the drone will advance at a speed of at least 0.25 meters in second. We will measure the path length and the flight time. From this data we will calculate the average speed.

2. **Checkpoint detection** – the drone will detect at least 90% of check points marks. We will calculate the hit ratio \( \frac{\text{detected checkpoint}}{\text{total checkpoints}} \times 100 \).

3. **Clear and Easy for use Tutorial** – we will publish our code and documentation at the official AR Drone community; from the community feedbacks we can estimate the quality of our tutorial.
4. **Path installation space** – the path could be set in any open space. During project development we will test our drone auto pilot in different places like: university classes, public parks, dorms corridors etc.
8. **Chapter 8 - Task List**

### 8.1. **Control task list**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control Implementation</td>
<td>5 days</td>
<td>Wed 25/04/12</td>
<td>Tue 01/05/12</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Implement UdpWorkers</td>
<td>3 days</td>
<td>Wed 25/04/12</td>
<td>Fri 27/04/12</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Implement Drone</td>
<td>1 day</td>
<td>Mon 30/04/12</td>
<td>Mon 30/04/12</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Testing Drone Functionality</td>
<td>1 day</td>
<td>Tue 01/05/12</td>
<td>Tue 01/05/12</td>
<td>2</td>
</tr>
</tbody>
</table>

### 8.2. **Utility task list**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Utilities Implementation</td>
<td>4 days</td>
<td>Wed 02/05/12</td>
<td>Mon 07/05/12</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Implement CheckPoint Handler</td>
<td>1 day</td>
<td>Wed 02/05/12</td>
<td>Wed 02/05/12</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Implement Image formatter</td>
<td>1 day</td>
<td>Wed 02/05/12</td>
<td>Wed 02/05/12</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Unit Tests for Image Formatter</td>
<td>1 day</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Implement Commands</td>
<td>1 day</td>
<td>Fri 04/05/12</td>
<td>Fri 04/05/12</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Unit Tests for commands</td>
<td>1 day</td>
<td>Mon 07/05/12</td>
<td>Mon 07/05/12</td>
<td>8</td>
</tr>
</tbody>
</table>

### 8.3. **Image processing task list**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Image Processing Implementation</td>
<td>8 days</td>
<td>Thu 03/05/12</td>
<td>Mon 14/05/12</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>Implement Sign Detector</td>
<td>1 day</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>Implement Path Detector</td>
<td>0 days</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>Implement Ground Stripe Detector</td>
<td>1 day</td>
<td>Fri 04/05/12</td>
<td>Fri 04/05/12</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>Unit Tests for Ground Stripe Detector</td>
<td>1 day</td>
<td>Mon 07/05/12</td>
<td>Mon 07/05/12</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>Implement Check Point Detector</td>
<td>0 days</td>
<td>Mon 07/05/12</td>
<td>Mon 07/05/12</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>Implement QR Code Detector</td>
<td>1 day</td>
<td>Tue 08/05/12</td>
<td>Tue 08/05/12</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>Unit Tests for QR Code Detector</td>
<td>1 day</td>
<td>Wed 09/05/12</td>
<td>Wed 09/05/12</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>Implement Sign Selector</td>
<td>0 days</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td>17</td>
</tr>
<tr>
<td>Task ID</td>
<td>Task Name</td>
<td>Duration</td>
<td>Start</td>
<td>Finish</td>
<td>Predecessors</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------</td>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>20</td>
<td>Implement Biggest Sign Selector</td>
<td>1 day</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Unit Tests for Biggest sign selector</td>
<td>1 day</td>
<td>Fri 04/05/12</td>
<td>Fri 04/05/12</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Implement Feature Extractor</td>
<td>0 days</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Implement Angle Extractor</td>
<td>1 day</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Unit Tests for Angle extractor</td>
<td>1 day</td>
<td>Fri 04/05/12</td>
<td>Fri 04/05/12</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Implement Horizontal Distance Extractor</td>
<td>1 day</td>
<td>Thu 03/05/12</td>
<td>Thu 03/05/12</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Unit Tests for horizontal distance extraction</td>
<td>1 day</td>
<td>Fri 04/05/12</td>
<td>Fri 04/05/12</td>
<td></td>
</tr>
</tbody>
</table>

8.4. **Navigation task list**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Course Advisor</td>
<td>4 days</td>
<td>Tue 15/05/12</td>
<td>Fri 18/05/12</td>
<td>1,5,11</td>
</tr>
<tr>
<td>28</td>
<td>Implement CourseAdvisor</td>
<td>3 days</td>
<td>Tue 15/05/12</td>
<td>Thu 17/05/12</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Implement Ground Stripe Course Advisor</td>
<td>1 day</td>
<td>Fri 18/05/12</td>
<td>Fri 18/05/12</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>Unit Tests for ground stripe Advisor</td>
<td>2 days</td>
<td>Tue 15/05/12</td>
<td>Wed 16/05/12</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Implement Horizontal Placard Course Advisor</td>
<td>1 day</td>
<td>Tue 15/05/12</td>
<td>Thu 17/05/12</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Unit tests for horizontal placard advisor</td>
<td>2 days</td>
<td>Wed 16/05/12</td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

8.5. **User Interface**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>UI</td>
<td>2 days</td>
<td>Thu 24/05/12</td>
<td>Fri 25/05/12</td>
<td>36</td>
</tr>
<tr>
<td>34</td>
<td>UI Implementation</td>
<td>2 days</td>
<td>Thu 24/05/12</td>
<td>Fri 25/05/12</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Unit Tests for UI</td>
<td>1 day</td>
<td>Thu 24/05/12</td>
<td>Thu 24/05/12</td>
<td></td>
</tr>
</tbody>
</table>

8.6. **Auto Pilot**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>AutoPilot Implementation</td>
<td>3 days?</td>
<td>Mon 21/05/12</td>
<td>Wed 23/05/12</td>
<td>27</td>
</tr>
<tr>
<td>37</td>
<td>System Integration</td>
<td>1 day?</td>
<td>Tue 22/05/12</td>
<td>Wed 23/05/12</td>
<td>39</td>
</tr>
<tr>
<td>38</td>
<td>System testing</td>
<td>1 day?</td>
<td>Wed 23/05/12</td>
<td></td>
<td>37</td>
</tr>
<tr>
<td>39</td>
<td>Implement AutoPilot</td>
<td>1 day</td>
<td>Mon 21/05/12</td>
<td></td>
<td>37</td>
</tr>
</tbody>
</table>
### 8.7. Documents and Projects Day task list

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Documents and Projects Day</td>
<td>219 days</td>
<td>Fri 30/09/11</td>
<td>Wed 01/08/12</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Working on Testing document</td>
<td>12 days</td>
<td>Fri 30/03/12</td>
<td>Mon 16/04/12</td>
<td>44</td>
</tr>
<tr>
<td>42</td>
<td>Working on Projects Day Poster</td>
<td>7 days</td>
<td>Thu 24/05/12</td>
<td>Fri 01/06/12</td>
<td>36</td>
</tr>
<tr>
<td>43</td>
<td>Working on User manual</td>
<td>7 days</td>
<td>Thu 24/05/12</td>
<td>Fri 01/06/12</td>
<td>36</td>
</tr>
<tr>
<td>44</td>
<td>Working on ADD Document</td>
<td>70 days</td>
<td>Fri 23/12/11</td>
<td>Thu 29/03/12</td>
<td>46</td>
</tr>
<tr>
<td>45</td>
<td>Working on Prototype Presentation</td>
<td>7 days</td>
<td>Fri 23/12/11</td>
<td>Mon 02/01/12</td>
<td>46</td>
</tr>
<tr>
<td>46</td>
<td>Working on ARD Document</td>
<td>60 days</td>
<td>Fri 30/09/11</td>
<td>Thu 22/12/11</td>
<td></td>
</tr>
</tbody>
</table>