Parrot Mini Drone



**Date:** 8 September 2022

**Year:** 4 **Semester:** 7 **Period:** Q1

**Location:** Helmond

**Authors:** Demian Verheesen, Giel Jansen, Pim Vos & Zohair Lasfar **Group:** 5

Project contribution

|  |  |
| --- | --- |
| **Demian Verheesen** | |
| Report | Template of the report and filled in chapters “Problem description”, “Requirements”, “Test plan” & “Used hardware and software”. Updated “Design” chapter and added “State Machine Diagram”. |
| Program | Helping the others with their modules, made the blue waypoint module and updated the path planning module. |
| **Giel Jansen** | |
| Report | Filled in chapters “Block Definition Diagram” & “Internal Block Diagram” |
| Program | Made the constant height module and helped with the blue waypoint module. |
| **Pim Vos** | |
| Report | Filled in chapter “Use Case diagram”. |
| Program | Made/updated the waypoint follower module. |
| **Zohair Lasfar** | |
| Report | Filled in chapter “Introduction”. |
| Program | Made the blue dominance module. |

Table of Contents

[1 Introduction 4](#_Toc113627822)

[2 Project description 4](#_Toc113627823)

[2.1 Project goal 4](#_Toc113627824)

[2.2 Project approach 4](#_Toc113627825)

[2.3 Project background 4](#_Toc113627826)

[3 Requirements 5](#_Toc113627827)

[4 Design 6](#_Toc113627828)

[4.1 Use Case Diagram 6](#_Toc113627829)

[4.2 Block Definition Diagram 6](#_Toc113627830)

[4.3 Internal Block Diagram 7](#_Toc113627831)

[4.4 State Machine Diagrams 7](#_Toc113627832)

[5 Test plan 8](#_Toc113627833)

[5.1 Simulation testing 8](#_Toc113627834)

[5.2 Drone testing 9](#_Toc113627835)

[6 Used hardware & software 10](#_Toc113627836)

[6.1 Hardware 10](#_Toc113627837)

[6.2 Software 10](#_Toc113627838)

[7 References 10](#_Toc113627839)

# Introduction

This document describes the requirements and design for the Flight Control System of the Parrot MiniDrone. This system has to be built as part of the 4th year grade module “Model-Based Systems Engineering”. It describes both the user requirements, the system specification and the testing.

# Project description

## Project goal

The goal of this project is to create a program using MATLAB Simulink for a Parrot mini drone to fly. This program should allow the drone to take-off, land and fly a predetermined path in the shape of the figure eight. It also should be able to use the distance sensor to always stay 1 meter from the ground. The camera should be used to take pictures or video during the flight and to maybe detect and land near a blue object. This application could be used for filming a movie or as a following drone during a race and the object detection could be used for tracking an object during filming or a race.

## Project approach

The project will be approached using the V-model development process and the program will be split up into multiple modules to allow for parallel development. The project will start by defining the requirements for the project and the division of the tasks and modules. Then the different modules will be developed and tested. Finally, the modules will be integrated and the full program tested to the requirements in a simulated environment and real-life.

## Project background

The project is part of the 7th semester “Model-Based Systems Engineering” (MBSE) course for the 4th year of the electrical engineering study. MBSE is a modeling system used to support the complete project from the requirements to the design, verification and validation. Using these models allows people to have a better overview of the project than by transferring the project information using many documents. This is especially true when it’s a complex system involving multiple teams.

This course includes learning and using the MATLAB system composer and stateflow add-ons to make models for different systems. This includes the models required for this project.

# Requirements

The drone project has a few requirements and these can be seen in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Requirement** | **Rational** | **Priority** | **Status** |
| R1 | The drone can take-off safely. | Don’t want a drone crash during take-off. | Must | Accepted |
| R2 | The drone can land safely. | Don’t want a drone crash or hard landing during the landing. | Must | Accepted |
| R3 | The drone can follow a predetermined path in the shape of the figure eight. | During a movie scene the drone should follow its predetermined path. It can also be used to follow a race track using the predetermined path. | Must | Accepted |
| R4 | The drone stays 1 meter from the ground using the distance sensor. | If required for a movie scene or when following a race track, it should be able to stay the same distance from the ground. | Must | Accepted |
| R5 | The drone can take video during its flight. | A movie scene requires video to be taken and when following the cars on the race track it should also show video. | Must | Accepted |
| R6 | The drone can use the camera to find a blue object. | This can be useful to track objects or for finding objects on its path. | Should | Accepted |
| R7 | The drone can land at the location of the blue object. | This can be used to land or start doing something else during the filming of a scene. | Could | Accepted |

Table 3‑1 Requirements

# Design

## Use Case Diagram

Diagram

Description automatically generated

Figure 4‑1 Use case diagram

## Block Definition Diagram

A picture containing diagram

Description automatically generated

Figure 4‑2 Block Definition Diagram

## Internal Block Diagram

Chart, box and whisker chart

Description automatically generated

Figure 4‑3 Internal Block Diagram

## State Machine Diagrams

Diagram

Description automatically generated

Figure 4‑4 Stateflow diagram of Blue waypoint module

# Test plan

The testing of the drone program is divided into two parts. The first being the simulation testing and the second being the real-life drone testing.

## Simulation testing

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Test** | **Result** |
| R1 | The drone can take-off safely. | Run the simulation and see if the drone takes off correctly. | Pass |
| R2 | The drone can land safely. | Run the simulation and see if the drone lands correctly. | Pass |
| R3 | The drone can follow a predetermined path in the shape of the figure eight. | Run the simulation and see if the drone flies the predetermined path correctly. | Pass |
| R4 | The drone stays 1 meter from the ground using the distance sensor. | Run the simulation and see if the drone keeps a height of 1 meter from the ground correctly. | Pass |
| R5 | The drone can take video during its flight. | Run the simulation and see if the drone takes video correctly. | Pass |
| R6 | The drone can use the camera to find a blue object. | Run the simulation and see if the camera detects the blue object (using for example a scope). | Pass |
| R7 | The drone can land at the location of the blue object. | Run the simulation and see if the drone lands at the blue objects location when it has finished its predetermined path. | Pass |

Table 5‑1 Simulation test plan

## Drone testing

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Test** | **Result** |
| R1 | The drone can take-off safely. | Run the program and see if the drone takes off correctly. | Pass |
| R2 | The drone can land safely. | Run the program and see if the drone lands correctly. | Pass |
| R3 | The drone can follow a predetermined path in the shape of the figure eight. | Run the program and see if the drone flies the predetermined path correctly. | Pass |
| R4 | The drone stays 1 meter from the ground using the distance sensor. | Run the program and see if the drone keeps a height of 1 meter from the ground correctly. | Pass/Fail, when there is an abrupt change of height the drone throws an error and fails. |
| R5 | The drone can take video during its flight. | Run the program and see if the drone takes video correctly. | Pass |
| R6 | The drone can use the camera to find a blue object. | Run the program and see if the camera detects the blue object on the blue dominance video viewer. | Pass |
| R7 | The drone can land at the location of the blue object. | Run the program and see if the drone lands at the blue objects location when it has finished its predetermined path. | Not tested due to camera not working until end of the project and due to optical flow errors. |

Table 5‑2 Drone test plan

# Used hardware & software

## Hardware

The hardware used for this project is the Parrot Mambo mini drone.

## Software

The software used for this project is MATLAB from MathWorks. The version for MATLAB is R2022a. This project also required the usage of multiple MATLAB add-ons, which can be seen in the table below.

|  |  |
| --- | --- |
| **MATLAB add-on** | **Version number** |
| Simulink | 10.5 |
| Image Processing Toolbox | 11.5 |
| Signal Processing Toolbox | 9.0 |
| Control System Toolbox | 10.11.1 |
| Curve Fitting Toolbox | 3.7 |
| Statistics and Machine Learning Toolbox | 12.3 |
| Symbolic Math Toolbox | 9.1 |
| System Composer | 2.2 |
| Stateflow | 10.6 |
| Requirements Toolbox | 2.0 |
| Simulink Support Package for Parrot Minidrones | 22.1.1 |
| MATLAB Support Package for Parrot Drones | 22.1.1 |
| MATLAB Coder | 5.4 |
| Simulink Coder | 9.7 |
| Embedded Coder | 7.8 |
| Aerospace Toolbox | 4.2 |
| Aerospace Blockset | 5.2 |
| Computer Vision Toolbox | 10.2 |
| Simulink 3D Animation | 9.4 |
| UAV Toolbox | 1.3 |
| ROS Toolbox | 1.5 |

Table 6‑1 Required MATLAB add-ons

# References

The main reference for this project is the GitHub, which stores the project and can be accessed using the link below. The most up-to-date project is in the development branch.

<https://github.com/DemianVerheesen/DronePatternMBSE>