Package delivering
Mini Drone

Parrot

Authors:

- Joppe Cutsem, 3866661
- Tobias van Diepen, 3945588
- Jochem van Oosteroom, 3880990

Course: MBSE

Semester: S7, 2022/2023

Group: 2

Creation Date: Sep 5, 2022

Contribution

Name	Contribution
Joppe Van Cutsem	Architectures, Stateflow diagram, Testing
Tobias van Diepen	Architectures, Documentation, Organizing specification
Jochem van Oosteroom	Programming, Simulation, Testing

Table 1: Contribution

Content

List of tables

List of figures

Abstract

To complete the course MBSE we designed a minidrone project. The goal of this minidrone project is to fly a predetermined path to deliver a package to a particular delivery address. This project is made with the usage of MATLAB R2022 and specific addons needed to control the Parrot Mambo minidrone.

Introduction

This document will describe the package delivering mini drone project. The project is developed using model based system engineering to give a better understanding of the system and a faster result with less debugging time. The application of the delivering drone is to fly packages from distribution center to clients homes at remote locations where no conventional way is possible. The goal will be to developed an architecture using MBSE (Model Based System Engineering) after this is done it will be tested with simulation and afterwards it will be deployed on the mini drone. By making use of the MBSE model, errors will be faster noticed and therefor will reduce developing time.

Figure 1: Package Delivering Drone

Assignment

Application

The application for the mini drone is to deliver packages form place A to B. Think of small parcels that needs to be delivered from a distribution center to a client's home. In remote places were no basic delivery is possible using a drone instead of a conventional way will be faster and better for the environment. So in densely populated countries such as the Netherlands, this method of package delivery will not be profitable. But there are many countries that have less dense population and will benefit from fast package delivery with a drone.

Assignment description

To demonstrate the use of the packages delivering drone a small demo will be made to test the proof of concept. This demo will consist of a pre-defined path that the drone must fly on the basis of a structurally based program with MathWorks. The goal is to fly a path which proposes the delivery of packages:

- The drone takes off at the distribution center after the package is attached and secured.
- The drone then takes a picture of the current status of the package, to make sure the package has not been defected in any earlier production phases.
- To make sure the drone is also able to elevate, we decided the drone must elevate to a different heights in certain points in it's path.
- At the end of the path the drone must safely hover above the delivery spot, to make sure at this state the drone could safely deliver the package.
- At the hover-state at the end the drone must take another picture, to make sure the package hasn't been defected during delivering of the package. After this the drone will fly back.

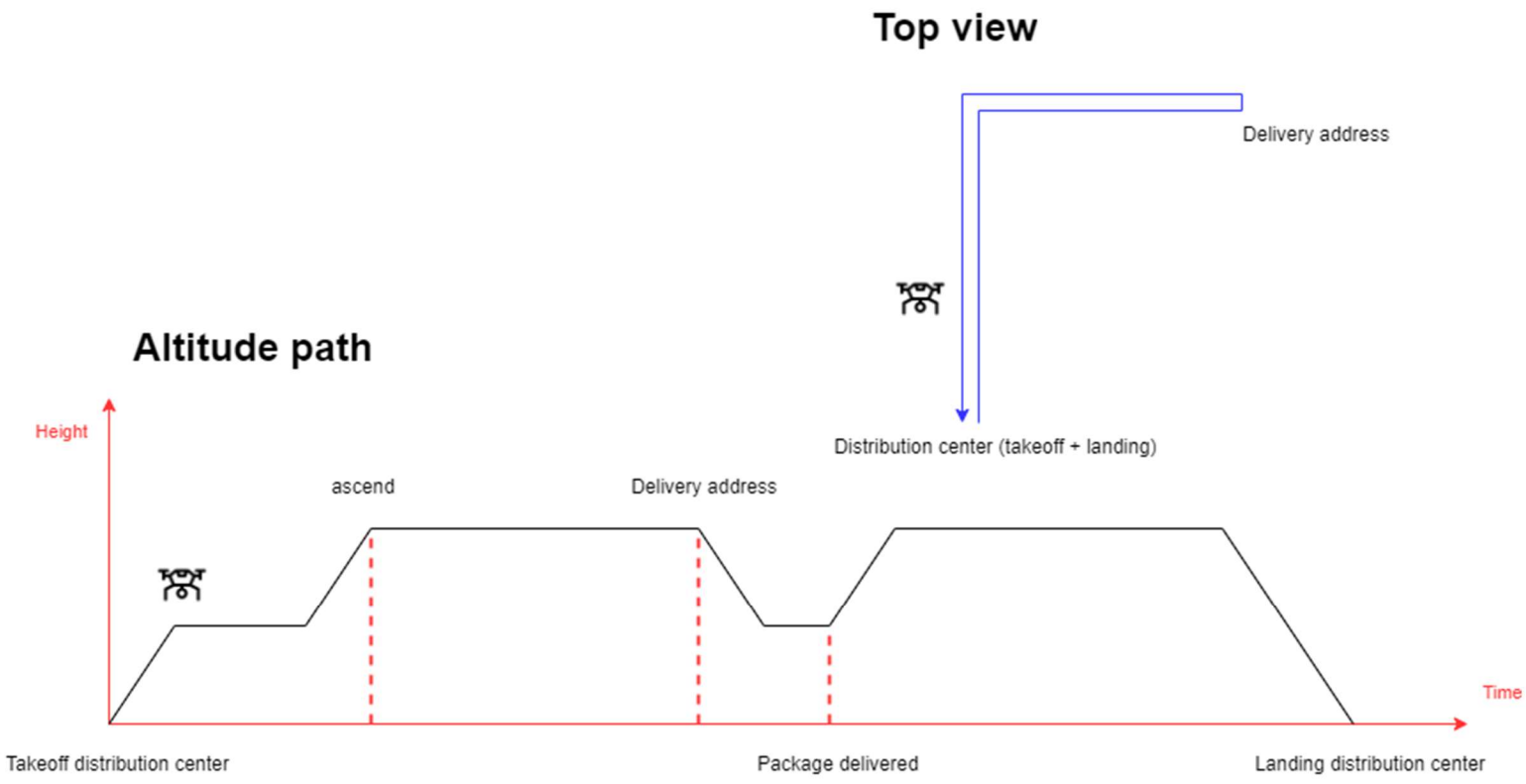

Figure 2: Example Path

Approach

The starting point of this project is the MathWorks template called “mini drone hover”. This template will cause the mini drone to hover at steady point and after a specific time the drone will land. The project started with a thinking phase in which the functionality of the template was agreed on the basis of making a functional architecture with system composer. Based on this functionality, research has been done into the functionalities of the template projects to determine which functionalities we need internally of the template and to determine which subsystems need to be adjust to get the desired result. After modifying the template project, the project is simulated into a desired simulation to ensure that the drone has to perform as few failing test runs as possible and to prevent drone failure.

MBSE

MBSE brings together three concepts: model, systems thinking, and systems engineering:

A **model** is a simplified version of something--a graphical, mathematical, or physical representation that abstracts reality to eliminate some complexity. This definition implies formality or rules in simplifying, representing, or abstracting. To model a system, a systems architect must represent the system with less detail so that its structure and behavior are apparent and its complexity is manageable. In other words, models should sufficiently represent the system, and the system should confirm the models.

Systems thinking is a way of looking at a system under consideration not as a self-sufficient entity, but as part of a larger system. Systems thinking is not the same as a systematic adherence to following good plans, collecting statistics, or being methodical. The systems engineer observes the system from a distance; explores its boundaries, context, and lifecycle; notes its behavior; and identifies patterns. This method can help the engineer to identify issues (e.g., missing interaction, a missing step in a process, duplication of effort, missed opportunity for automation) and manage a system's complexity. Although systems engineers must break down and analyze the system in the beginning--identify parts and describe connections between them--with systems thinking, they later synthesize the parts back into a coherent whole. Parts are not just connected to other parts, they depend on each other to work properly. Systems thinking emphasizes this interconnectedness. The behavior of the system emerges from the activities of the system's subparts. Observing the system's interconnections, the systems engineer identifies feedback loops and causality patterns that may not be apparent at first. Systems thinking can help make issues more apparent and easier to identify, balance the system, and manage the system's complexity.

Systems engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods. It brings together a number of techniques to make sure that all requirements are satisfied by the designed system. It concentrates on architecture, implementation, integration, analysis, and management of a system during its lifecycle. It also considers software, hardware, personnel, processes, and procedural aspects of the system.

Background

The drone used in this project is the Parrot Mambo minidrone. The Parrot Mambo is a quadcopter that allows the user to fly it with a mobile app or program it with for example Mathworks. The drone can be programmed using the Parrot Minidrones Support from Simulink and multiple specific addons needed for the example projects. These Matlab extensions give the user access to the data of a particular drone when connected via Bluetooth.

Requirements

Takeoff and Land

The minidrone must have Vertical takeoff and landing (VTOL) capabilities. This will enable the minidrone to have:

- efficient maneuverability
- ability to hover steadily
- minimal space needed to launch

Perform Controlled Flight

The minidrone should have a 6 DOF movement capability during flight to achieve its desired applications.

Fly range

To avoid expected obstacles in its operation area, the drone must be able to achieve and maintain a constant altitude of 2 meters.

Altitude

The flight is managed from a control system, and it can operate autonomously or based on guidance from central location.

Flight Control

The flight is managed from a control system, and it can operate autonomously or based on guidance from central location.

Communicate with Ground base

The mini drone should be able to receive commands and send information to a ground base that controls its operation.

Camera

The minidrone must be able to snap a picture of the package.

Carry a load

The mini drone should be able to carry a load to deliver packages.

Max size

Max size of 5x5x5 cm.

Max weight

Max weight of 500 grams.

Pick up controller

In order to take the packages the drone should have a pick up system for delivering the package.

Stakeholder

During designing the total system, we took two main stakeholders into account for the purposes of the system. These two stakeholders are: Distribution Center and the Client.

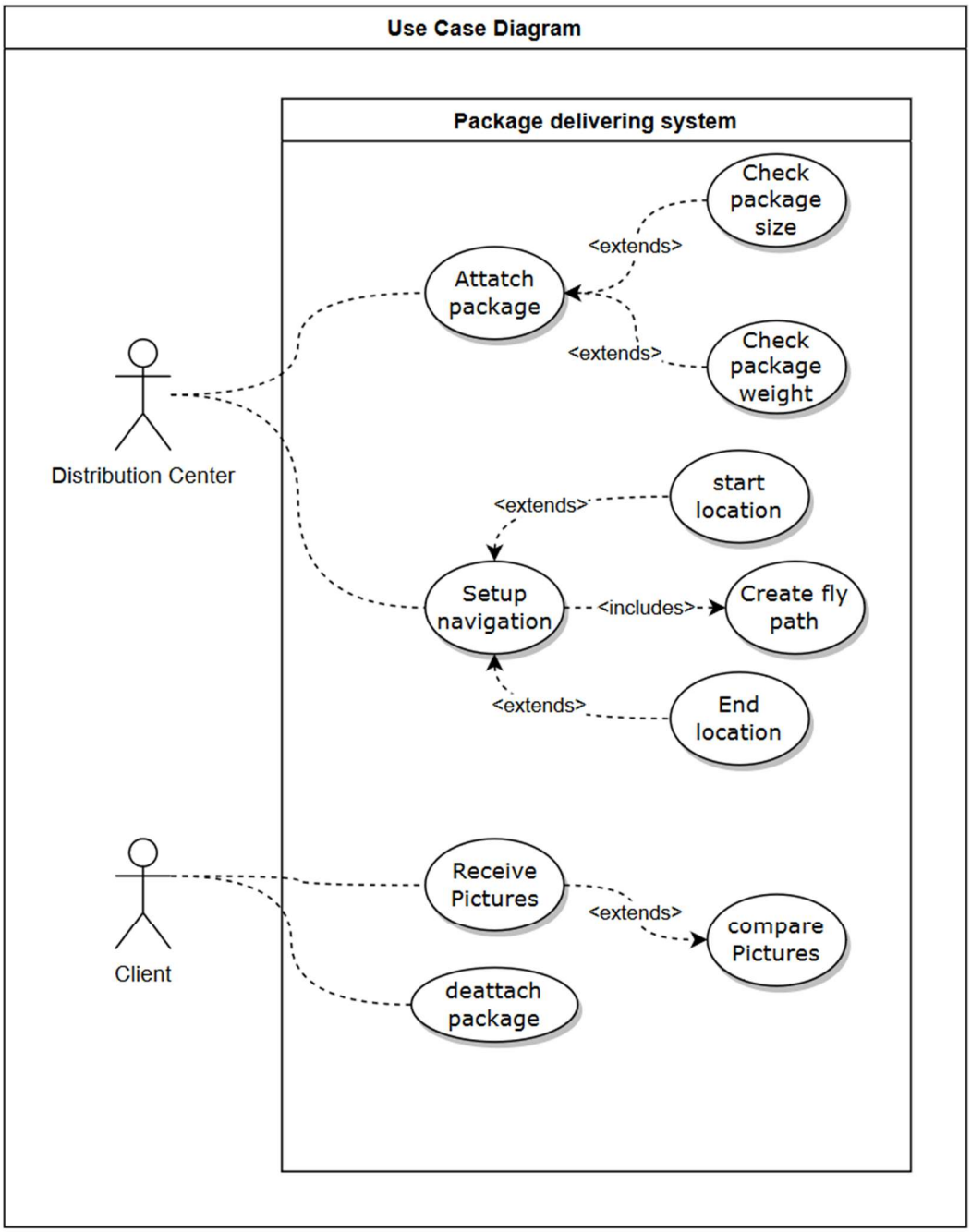

Figure 3: Use Case Diagram

Developed architecture

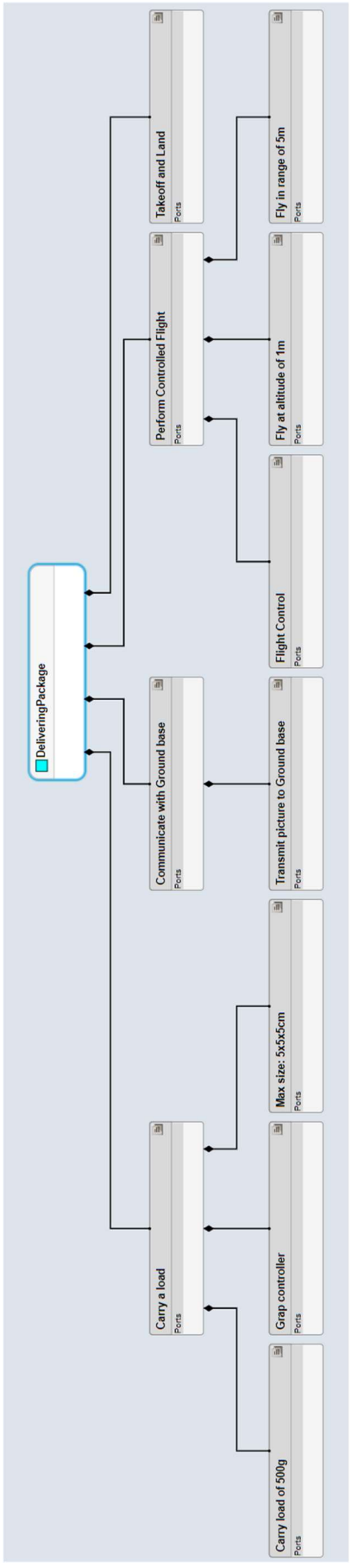

Figure 4: Functional Architecture

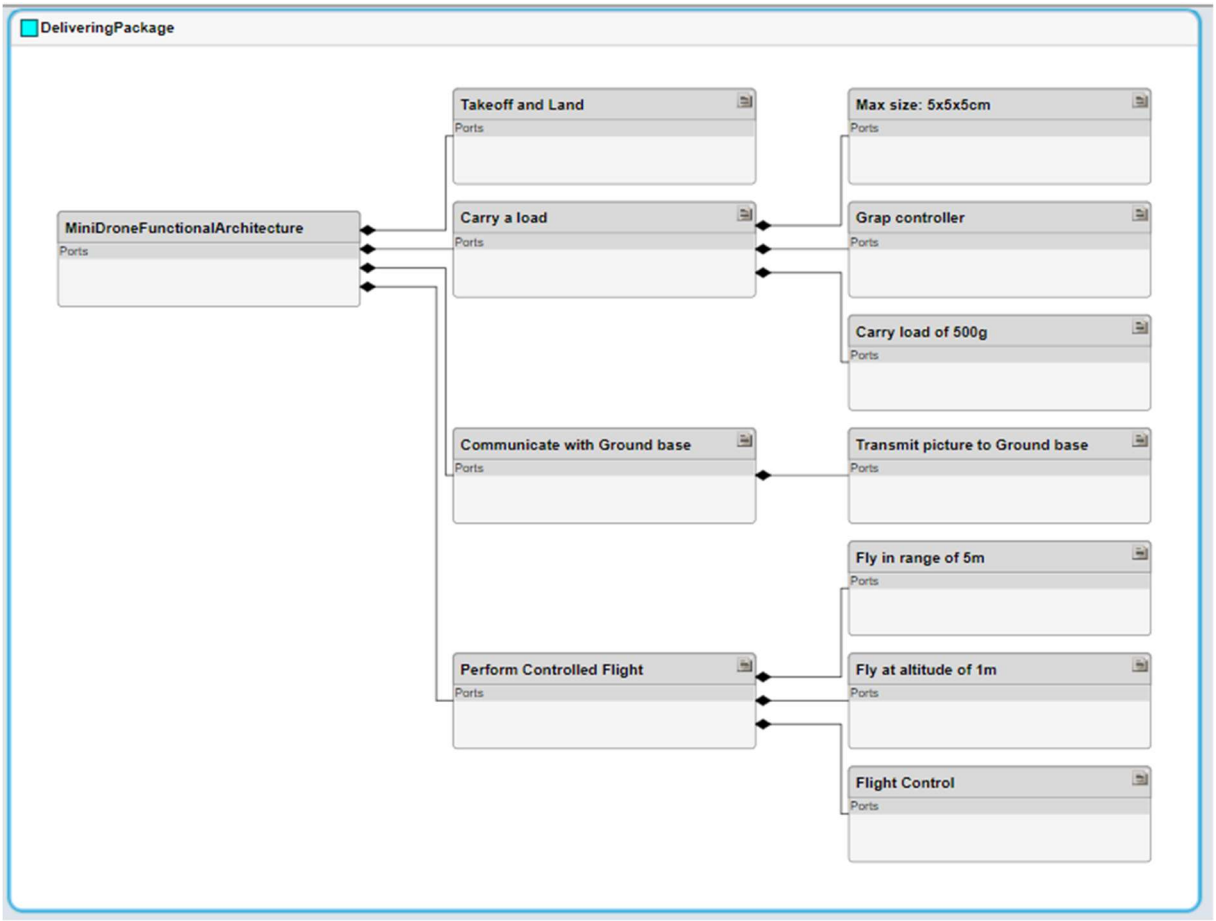

Figure 5: Funcional Architecture

Logical architecture

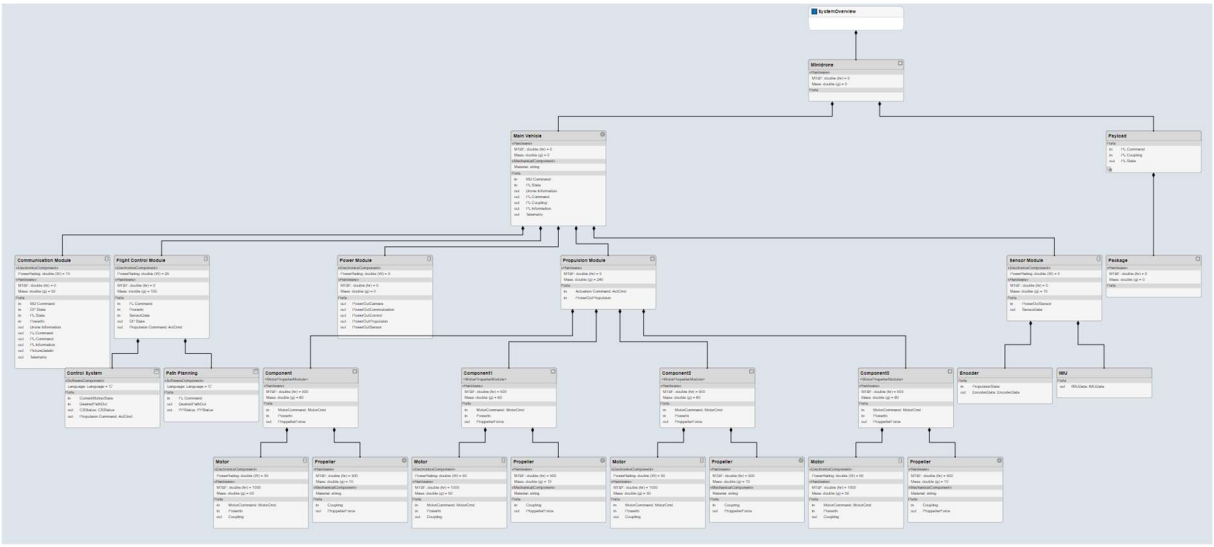

Figure 6: Logical Architecture

Matlab program

To alter the coordinates of the drone there is made a stateflow chart in order to adjust the x,y and z positions of the drone. The following states have been made so that the drone is able to fly every kind of path:

- Takeoff
- Ascend
- Descend
- Turn right
- Turn left
- Land

We used a stateflow chart because the path is already predefined before the drone takes off and therefore is not able to detect any changes in its path when flying.

Inputs	Outputs
xtemp	X
ytemp	Y
ztemp	Z
	yaw

Table 2: I/O list

Yaw has been added to change the degrees of the drone in order to make the drone also go 90 degrees in the direction it is going to be sure the drone always flies forward.

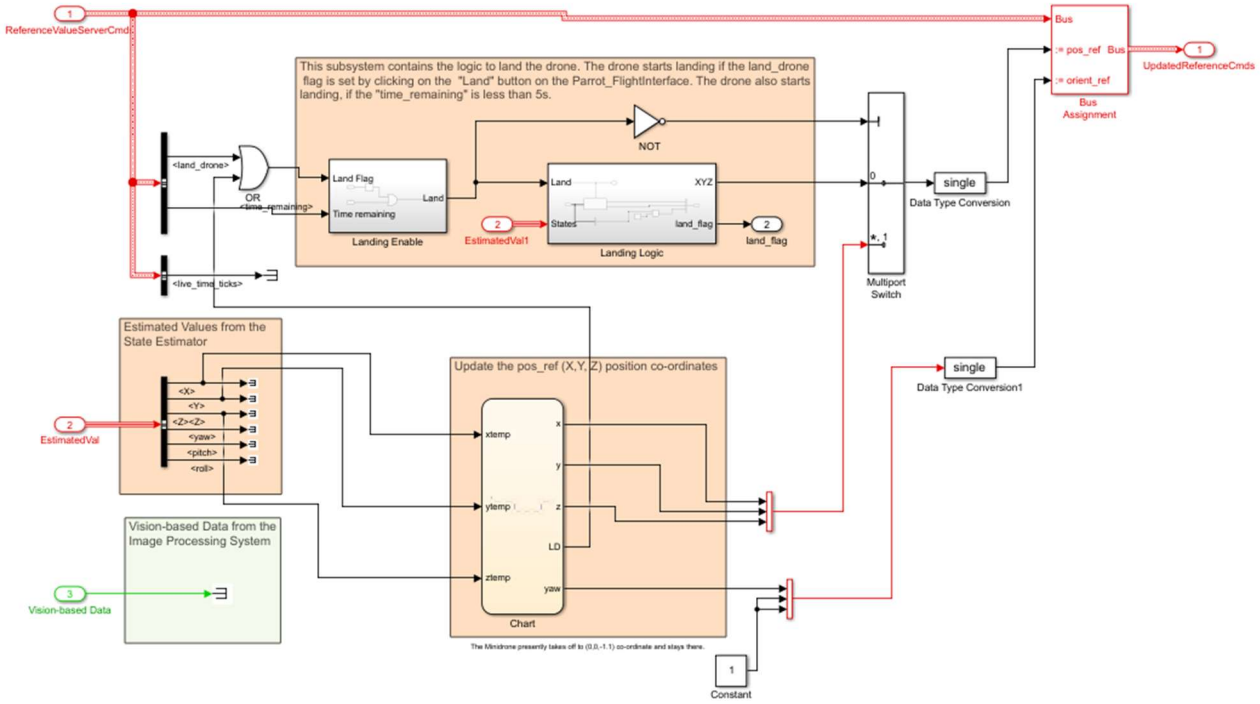

Figure 7: Path Planning Subsystem

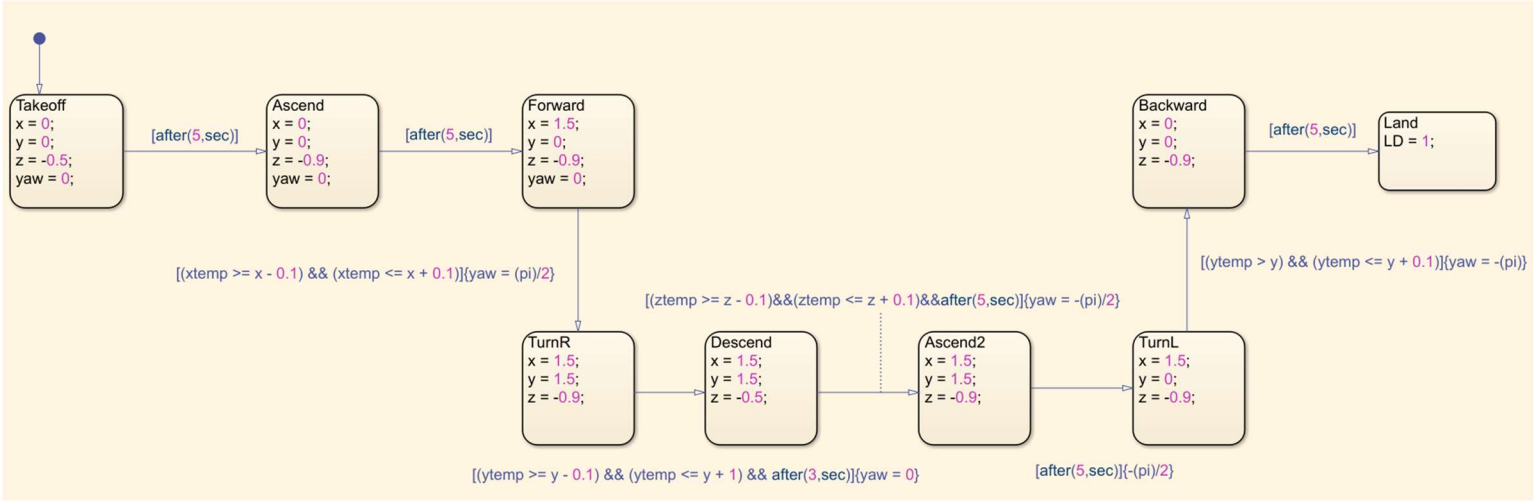

Figure 8: Stateflow

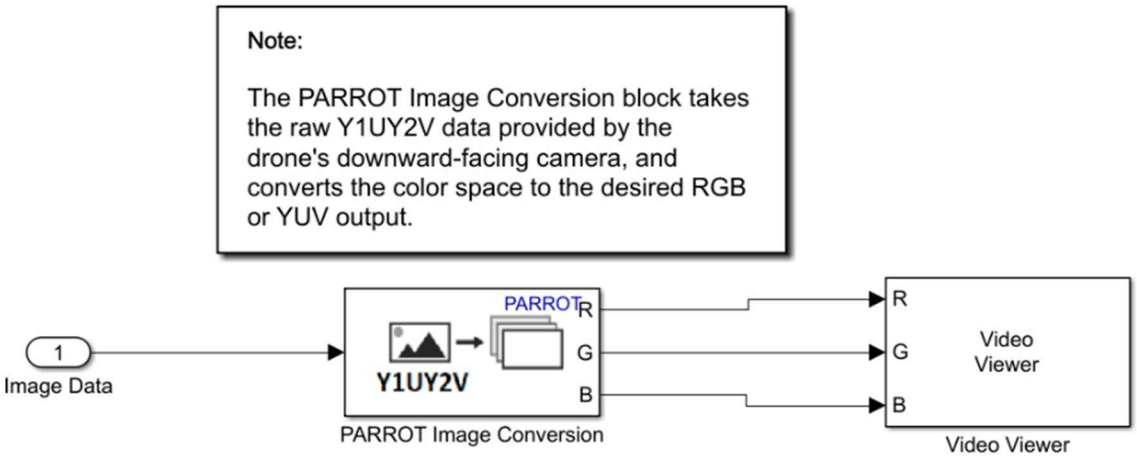

Figure 9: Video Viewer

Simulation

In the appendix you can find a OneDrive-link directly towards 2 video's of the simulations of the package delivering mini drone project.

Software packages

During the project a program from MathWorks called MATLAB is used to develop, model and test the created design. As describe the starting point was a template “mini drone hover” in order to get up and running with this template several addons needed to be downloaded. At table 3 a list of used packages is placed with version number.

Software Add-Ons	Software version
Aerospace Blockset	5.2
Aerospace Toolbox	4.2
Embedded Coder	7.8
ROS Toolbox	1.5
Simulink 3D Animation	9.4
Simulink control design	6.1
System composer	2.2
Stateflow	10.6
Support package for Parrot drone	22.1.1
Simulink Coder	9.7
Image processing toolbox	11.5
Signal processing toolbox	9.0
Requirements toolbox	2.0
Simulink	10.5

Table 3: Software Packages

Troubleshooting

During the project, several errors were corrected and hypotheses were established about the functioning of the parrot mambo mini drone.

During the actual tests with the mini drone, it was discovered that the drone functions less well on a surface without any patterns. Due to this solid surface, it can happen that the drone does not detect movement. This can then result in the drone starting to drift and therefore giving an error. This problem was solved by using white tape on a dark background to create relief, which allowed the drone to better detect motion.

In addition, we have found out that as soon as the drone starts to descend from a higher height to a height below half a meter, drift occurs. We think that by a too low altitude close to the ground the drone creates an airflow and drives itself out of the pre-programmed path. Besides this error in height we found out that the drone can't increase to a height greater than 1m. Having a height greater then 1m results in the drone flying less steady.

Conclusions and recommendations

With the help of the adjustments in the mini drone hover project, we managed to follow a certain path in which the drone makes a turn and descends to a lower altitude where the drone waits to actually deliver the package. In addition, there is a viewer where you have access to the image generated by the drone camera that is assembled at the bottom of the drone. Unfortunately, it was not possible to take a snapshot of the images shot by this camera.

For a newer version of this project, we have prepared several recommendations that can ensure that the next version is a vastly better version than the current result:

- The state flow could be adjusted where the x, y and z values are also inputs so that they can be adjusted more easily with new addresses. As a result, it becomes more clear what the drone does at x, y and z.
- Taking snapshots at certain times. As a result, it is not necessary to film the entire path, with the result that these files require less memory. Functionally, it is best to have a photo of the packages at the beginning of the path and a photo of the package at the relevant delivery address.
- The use of extensions for the drone:
    - The FPG camera: With this camera, the drone can also perceive the image in front of it. This can help, for example, with some unexpected obstacles in the path. And in addition, you can, for example, check the house number to be sure to prevent problems.
    - Grabber: With this expansion, the packages can be picked up and disassembled by the mini drone itself.

Figure 7: Drone Extensions

The complete project folder can be found on GitHub:

https://github.com/tobios123/MBSE_Gr2/

Reference list

https://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/

https://nl.mathworks.com/help/supportpkg/parrot/ref/follow-waypoints-parrot-drone.html

https://nl.mathworks.com/help/supportpkg/parrot/ug/optical-flow-with-parrot-minidrones.html

https://nl.mathworks.com/help/supportpkg/parrot/ref/getting-started-with-parrot-minidrone-vision.html

https://github.com/koraykzly/parrotMinidroneCompetition

https://www.youtube.com/watch?v=qPVnweXMguE

https://canvas.fontys.nl

Appendix

https://stichtingfontys-my.sharepoint.com/:f:/r/personal/433092_student_fontys_nl/Documents/EMBSE/Simulation?csf=1&web=1&e=2HWe0q