

---

---

# Testing Board for Quadcopter Parameters Visualization and Stability Analysis using Laser Detection and Ranging

---

---

SUPERVISED BY

DR. YASSINE SALEH-ALJ

EDITED BY

IDRISS SEFRIOUI  
NADIME LHASSANI  
HOUSSAM SABER

SPRING 2023

AL AKHAWAYN UNIVERSITY IN IFRANE

## **Abstract**

Whatever the abstract might be.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>System Components</b>	<b>4</b>
<b>3</b>	<b>System Architecture</b>	<b>5</b>
<b>4</b>	<b>System Implementation</b>	<b>5</b>
4.1	User Interface . . . . .	5
4.2	Calculations and Measurements . . . . .	6
4.3	Flowchart . . . . .	7
<b>5</b>	<b>Kalman Filter</b>	<b>7</b>
5.1	Prediction Step . . . . .	8
5.2	Update Step . . . . .	8

## List of Figures

3.1	System Architecture . . . . .	5
-----	-------------------------------	---

# 1 Introduction

Quadcopter and Aircraft applications in general require a lot of precision to ensure that the desired position and velocity are met. For even with very advanced implementations of control systems and accounting for nonlinear dynamics and noise through various control approaches such as estimations through Linear or Gaussian Quadratic Estimations. Which is why it is necessary to include an independent testing mechanism where we can analyze the various parameters that a portable aircraft designer is interested in, such as:

- Positioning
- Velocity
- Acceleration
- Orientation

To test some of these parameters, a testing bench capable of measuring the parameters using sensors and displaying them in a user-friendly interface, for the aircraft designer would be a good starting point.

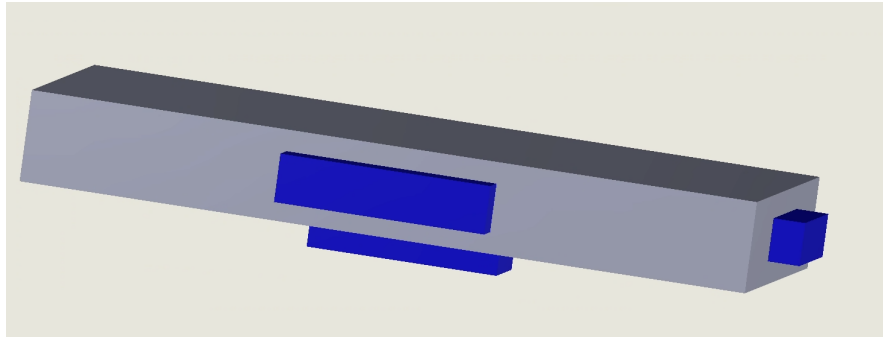
## 2 System Components

The system will be composed of the following:

- **Microcontroller:** The microcontroller will be responsible for taking the measurements from the sensors and performing the necessary calculations and sending the results to the user interface.
- **Ultrasound Sensor:** The ultrasound sensor will be used to measure the distance between the sensor and the Various objects in the environment.
- **A Bluetooth Module:** The Bluetooth module will be used to send the data from the microcontroller to the user interface.
- **User Interface:** A Linux-based system capable of displaying the data received from the microcontroller and displaying it in a user-friendly interface via a programmed GUI.
- **Power Supply:** A power supply capable of providing the necessary power to the system.
- **Gyroscope:** A gyroscope will be used to measure the orientation of the system.

### 3 System Architecture

The system bench will be composed as follows:



**Figure 3.1:** System Architecture

Where the microcontroller will be stored inside the box of the system and linked to the bluetooth module and the ultrasound sensors, which appear as blue boxes in Figure 3.1.

The mounting of the circuit will be done as follows:

## 4 System Implementation

### 4.1 User Interface

The user interface will be implemented using Python and the PyQt5 Library, which is a Python binding for the Qt5 GUI Library. The user interface will be composed of the following:

- **Main Window:** The main window will be the first window that the user will see when the program is launched. It will contain the following:
  - **Connect Button:** The connect button will be used to connect to the microcontroller via Bluetooth.
  - **Disconnect Button:** The disconnect button will be used to disconnect from the microcontroller.
  - **Start Button:** The start button will be used to start the data acquisition from the microcontroller.
  - **Stop Button:** The stop button will be used to stop the data acquisition from the microcontroller.
  - **Exit Button:** The exit button will be used to exit the program.
- **Settings Window:** The settings window will be used to set the parameters for the data acquisition from the microcontroller. It will contain the following:

- **Start Button:** The start button will be used to start the data acquisition from the microcontroller.
- **Stop Button:** The stop button will be used to stop the data acquisition from the microcontroller.
- **Exit Button:** The exit button will be used to exit the program.
- **Data Acquisition Window:** The data acquisition window will be used to display the data acquired from the microcontroller. It will contain the following:
  - **Start Button:** The start button will be used to start the data acquisition from the microcontroller.
  - **Stop Button:** The stop button will be used to stop the data acquisition from the microcontroller.
  - **Exit Button:** The exit button will be used to exit the program.

The user will be input a range for each parameter, or chose to ignore some of them, or even all of them in case of simple visualization of the data. If any system computed variable during any instant  $t$  is not within the range specific by the user, it will trigger a response of LEDs and Sound Emitters which will signal that the quadcopter is not functioning as desired.

## 4.2 Calculations and Measurements

The microcontroller will be responsible for taking the measurements from the sensors and performing the necessary calculations and sending the results to the user interface. The microcontroller will be responsible for the following:

- **Distance Measurement:** The distance measurement will be straightforward, the microcontroller will be automatically given the distance between the sensor and the object in front of it with the time of flight of the ultrasound signal.
- **Velocity Measurement:** The velocity measurement will be done point by point to be able to draw a graph of the velocity across each axis. The velocity will be calculated straightforwardly by the taking the distance  $\Delta d$  at a time  $t_i$  between points  $i - 1$  and  $i$  and dividing it by the time interval between two sent iterations.

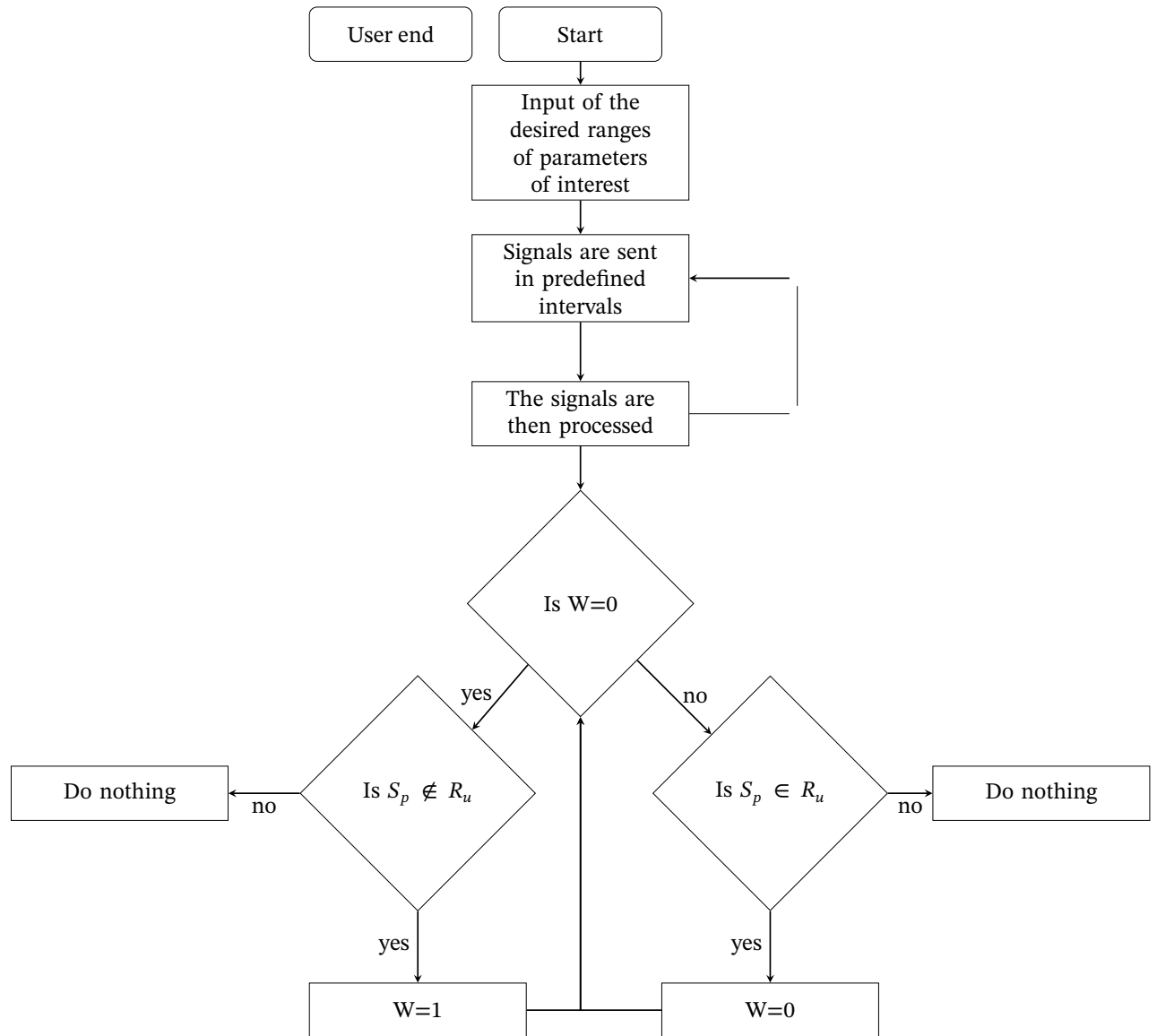
$$v_i = \frac{\Delta d}{\Delta t} \quad (4.1)$$

- **Acceleration Measurement:** The acceleration measurement will be done point by point to be able to draw a graph of the acceleration across each axis. The acceleration will be calculated straightforwardly by the taking the velocity  $\Delta v$  at a time  $t_i$  between points  $i - 1$  and  $i$  and dividing it by the time interval between two sent iterations.

$$a_i = \frac{\Delta v}{\Delta t} \quad (4.2)$$

- **Orientation Measurement:** The measurement of the orientation is a bit more complex, but by using a Gyroscope, we could be comparing the inclination of the quadcopter with a base value then calculate the error and send it to the user interface.

### 4.3 Flowchart



## 5 Kalman Filter

The Kalman Filter is a state estimation algorithm that is used to estimate the state of a system given a set of measurements. It is used in a wide variety of applications such as robotics, navigation, and control. The Kalman Filter is a recursive algorithm that uses the previous state and the current measurement to estimate the



current state. The Kalman Filter is composed of two parts, the prediction step and the update step.

## 5.1 Prediction Step

The prediction step is used to predict the state of the system at the next time step. It is composed of the following:

- **State Prediction:** The state prediction is used to predict the state of the system at the next time step. It is calculated by using the following equation:

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k \quad (5.1)$$

- **Covariance Prediction:** The covariance prediction is used to predict the covariance of the system at the next time step. It is calculated by using the following equation:

$$P_{k+1} = AP_kA^T + Q \quad (5.2)$$

## 5.2 Update Step

The update step is used to update the state of the system at the current time step. It is composed of the following:

- **Kalman Gain:** The Kalman Gain is used to calculate the optimal weighting between the prediction and the measurement. It is calculated by using the following equation:

$$K_k = P_{k+1}H^T(HP_{k+1}H^T + R)^{-1} \quad (5.3)$$

- **State Update:** The state update is used to update the state of the system at the current time step. It is calculated by using the following equation:

$$\hat{x}_k = \hat{x}_{k+1} + K_k(z_k - H\hat{x}_{k+1}) \quad (5.4)$$

- **Covariance Update:** The covariance update is used to update the covariance of the system at the current time step. It is calculated by using the following equation:

$$P_k = (I - K_kH)P_{k+1} \quad (5.5)$$

We will be using the Kalman Filter to minimize the error in the measurements of the sensors. The Kalman Filter will be used to estimate the state of the system at each time step. The state of the system will be the position of the quadcopter in the 3D space. The measurements will be the distance measurements from the sensors and the orientation of the quadcopter.