Testing Board for Quadcoptor Parameters Visualization and Stability Analysis using Ultrasonic and Gyroscope Detection

SUPERVISED BY

DR. YASSINE SALIH-ALJ

EDITED BY

IDRISS SEFRIOUI
NADIME LHASSANI
HOUSSAM SABER

Preface

As of this phase of the project, we have decided to drop the following features of the project:

- The third axis of rotation which is rotation around Z axis.

- The control system of the ultrasonic sensor, which was supposed to be done using a Proportional-Integral controller, but would be redundant with the implementation of the Kalman filter.

This is due to the fact that axis fusion will be extremely difficult and a lot more advanced and complex than what time allows us, by requiring a more advanced number system (quaternions) and more advanced concepts from Topology. [1]
The interface has been completed and is ready to be used, but the code is not yet operational because of delays in the shipment of hardware due to being stuck in customs, please refer to appendix A for details.
The code will be completed and tested as soon as the hardware arrives. As of now, the code can be found in the following link, which includes all of our progress and the code for the interface and our advances for the System implementation in C.
https://github.com/Riyll/Quadcopter-Testing-Machine

Contents

List of Figures

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 dyanamics 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 (Roll, Pitch)

To test these parameters, a testing bench capable of measuring them using sensors and displaying them in a user-friendly interface,for the aircraft designer would be a good starting point. The user will also be able to input ranges of the desired performance of the aircraft, and the system will be able to warn them if the aircraft is not performing as expected through LEDs and a Buzzer in addition to the graphical user interface.

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.

The following Table will specify the choice of each component.

Table 1: System Components

Component	Choice	Purchase Link
Microcontroller	STM32F401RE	Here
Ultrasound Sensor	VL53L1X Time-of-Flight	Here
Bluetooth Module	HM-10	Here
Power Supply	5V 2A Power Supply	Already Available
Gyroscope	MPU-6050	Here
LED	-	Already Available
Buzzer	-	Already Available

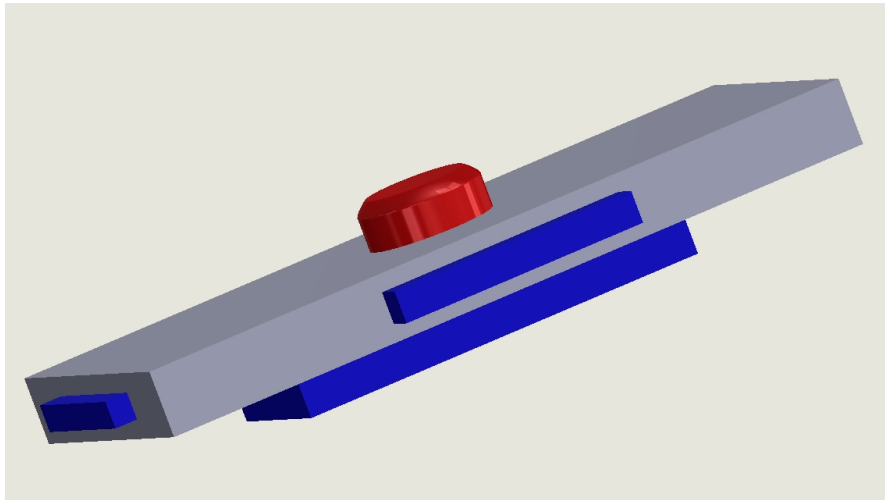# 3   System Architecture

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 3.2, with the Gyroscope appearing with a Red Color.
The mounting of the circuit will be done as follows:
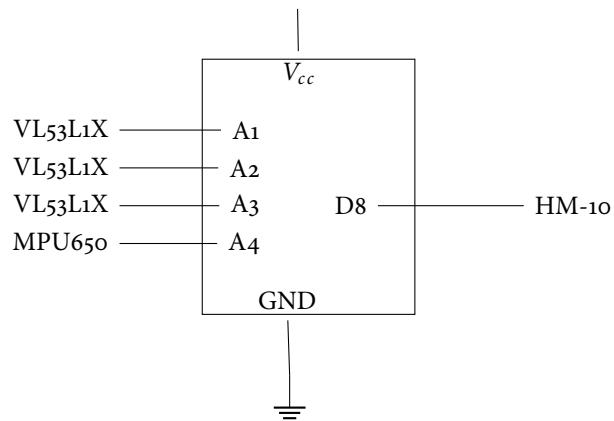
Figure 3.2: Microcontroller Circuit

Only the pins that are used in the circuit have been shown in the Figure. Unfortunately, it is not possible to simulate the circuit using popular circuit simulation software such as Proteus or Multisim, as either the MPU6050 or the Ultrasound sensors are not supported with the available libraries.
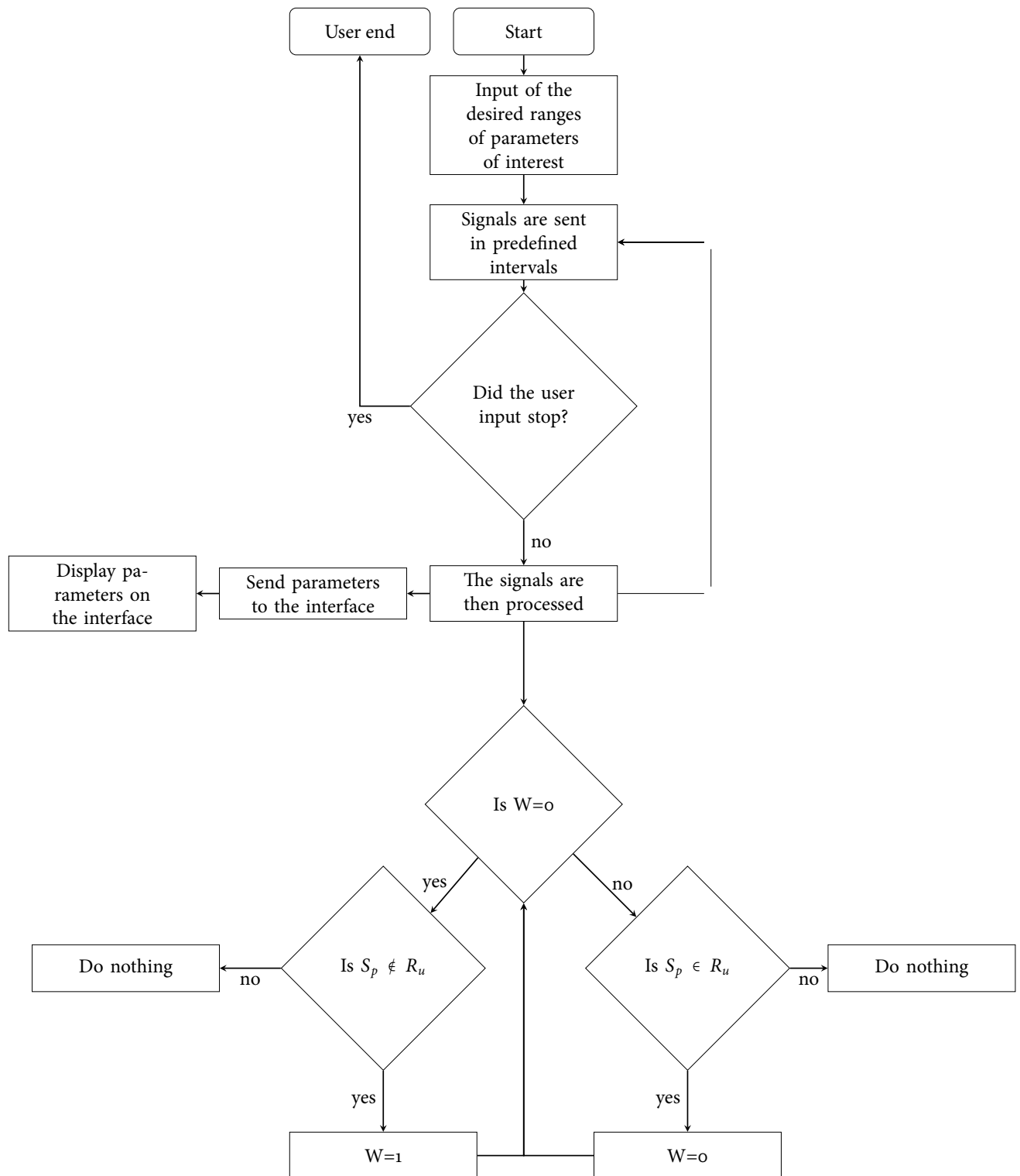
3.1 Flowchart

Figure 3.3: Algorithm of the system

4 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 straightforwadly by the taking the distance Δd at a time t_i between points $i-1$ and i and dividing it by the time interval between two sent iterations.

$$v_i = \frac{\Delta d}{\Delta t} \tag{4.1}$$

- **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 straightforwadly by the taking the velocity Δv at a time t_i between points $i-1$ and i and dividing it by the time interval between two sent iterations.

$$a_i = \frac{\Delta v}{\Delta t} \tag{4.2}$$

- **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.

To ensure the maximization of the accuracy of the results, we will be computing the Position, Velocity, and Acceleration of the quadcopter in another way using the Accelerometer, using the following equations:

$$\Delta x = \frac{1}{2} a_x \Delta t^2 + v_x \Delta t$$
$$\Delta v = a_x \Delta t \tag{4.3}$$

These will be the combined with the results of the Ultrasonic Sensor to get the most accurate results.

5 Graphical User Interface

The graphical user interface will be responsible for displaying the graphs of the parameters of the quadcopter, the user interface is separated into two windows:

- **Main Plot Window:** The main plot window will be responsible for displaying the graphs of the parameters of the quadcopter.
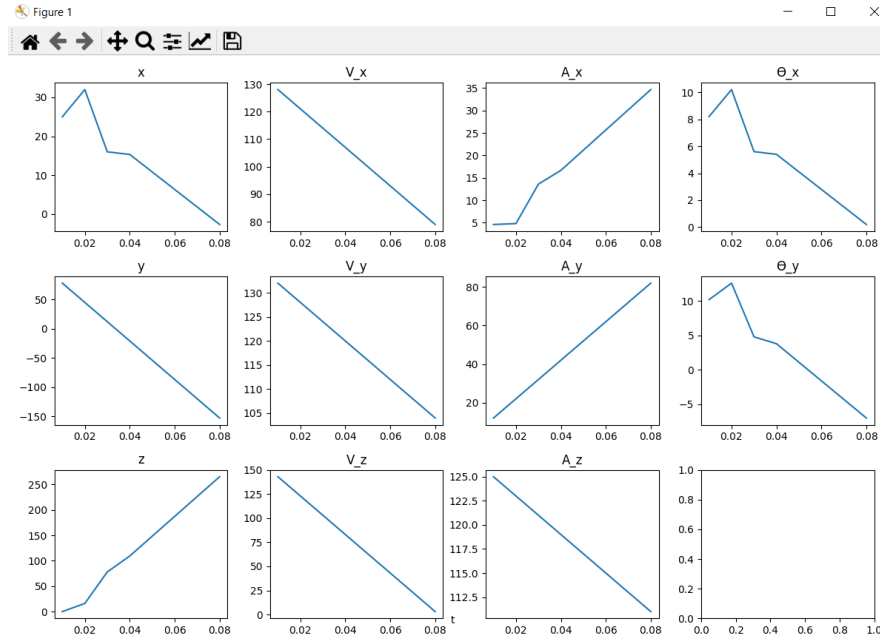

Figure 5.1: Main Plot Window

These graphs are updated in real time as the quadcopter travels.

- **Parameter Specification Window:** This window will be where the user inputs the minimum and maximum values of the parameters (ranges) for which they think the Quadcopter will be working, to be then alerted using LEDs and buzzers about the performance of the quadcopter not being within the specified ranges. Clicking on each parameter prompts a new window for which the user will be able to input their desired ranges.
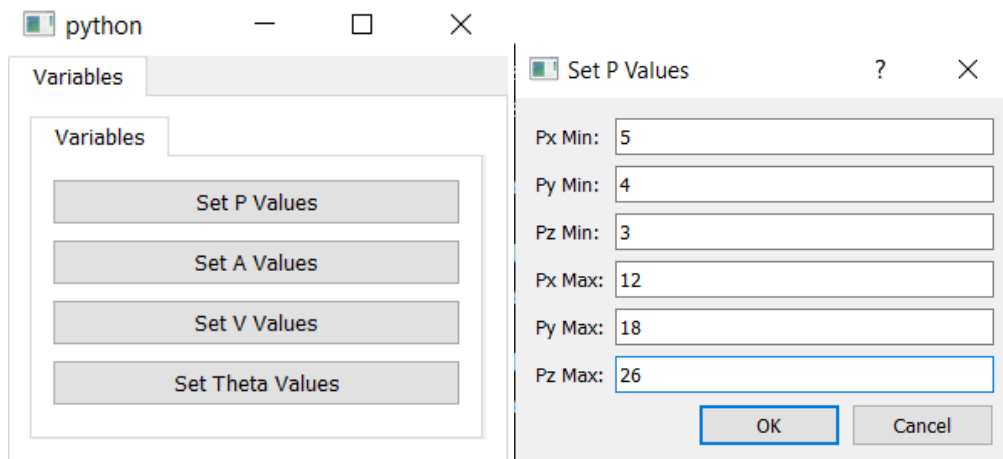
Figure 5.2: Parameter Specification Window

P refers to position, A to acceleration, and V to velocity, Theta is the angle of rotation (Pitch and Roll).

The inputed ranges are stored in the local memory and communicated to the microcontroller using the serial port. The microcontroller will then be responsible for comparing the values of the parameters with the ranges and alerting the user using LEDs and buzzers.


```
P Values: Px=('5', '12'), Py=('4', '18'), Pz=('3', '26')
```

Figure 5.3: Terminal Output Results after the input

Note: All the used values are Dummy Data as we cannot synchronize the data without the hardware.

6 Sensor Fusion using Kalman Filters

Since accelerometer and gyroscope sensors are both used together to determine the parameters (attitude) of the aircraft, we want to ensure that both of these sensors can have acceptable accuracy since the information required by the tester must be very precise. Combining both of them leads to deviation and the noise in the environment perturbates the sensing. To overcome this problem, we will be using a Kalman Filter to fuse the data from both sensors to ensure that the data is as accurate as possible.
The Kalman Filter is a recursive algorithm that uses the previous state of the system to predict the next state of the system. It uses the measurements of the system to correct the prediction and ensure that the prediction is as accurate as possible.
We will be using a Kalman Filter library given by [2] to implement the Kalman Filter in our system.
The Libraries have been modified accordingly and are integrated into the system as can be found in the GitHub repository linked in the Preface.

7 Conclusion and Future Work

The graphical user interface, which makes up more than a quarter of the project implementation is already completed, and communication with the Microcontroller is already configured. In addition, the theoretical study of Kalman Filter which is the principal concept that is behind this project is also completed, with a good start in implementation in the code for the STM32 Microcontroller which will hopefully be completed once the parts are received. The progress is definitely not as successful as we had hoped, but a lot of work was done from our part these last 2 weeks, which will hopefully smoothen the final part of the implementation to deliver an impeccable build.

A   Shipment Delay proofs

The Wednesday presented in the picture was supposed to be the 8th of March.

References

[1] Álvaro Deibe, José Augusto Antón Nacimiento, Jesús Cardenal, and Fernando López Peña. A kalman filter for nonlinear attitude estimation using time variable matrices and quaternions. *Sensors*, 20(23):6731, Nov 2020.

[2] Yuchao Huang. Kalmanfilter. `https://github.com/ycHepth/KalmanFilter%7D%7D`, 2019.

[3] Phil Kim and Lynn Huh. *Kalman filter for beginners: With MATLAB examples.* CreateSpace, 2011.

[4] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* jul 23 2013.

[5] STMicroelectronics. STM32 MCU Wiki Solutions. `https://wiki.st.com/stm32mcu/wiki/Solutions%7D%7D`.